

Lifelong Learning for Visual Recognition Systems

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von Dipl.-Inf. Alexander Freytag
geboren am 20. November 1988 in Erfurt

Gutachter:

1. Prof. Dr.-Ing. Joachim Denzler,
Friedrich-Schiller-Universität Jena, Germany
2. Prof. Dr.-Ing. Joachim Hornegger,
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
3. Prof. Trevor Darrell, Ph.D.,
University of California, Berkeley, United States of America

Tag der Einreichung: 21. Januar 2016

Tag der öffentlichen Verteidigung: 16. September 2016

Acknowledgments

This thesis would not have been possible without the support from a variety of great persons.

Dear Advisers: First of all, my deepest thanks go to my Ph.D. adviser Prof. Dr. Joachim Denzler for his never-ending support, his trust, and his intellectual input. Joachim made it possible that I was able to look into a variety of topics and responsibilities, to explore the beauty of science, to travel to conferences, and to grow personally. Of similar importance was Dr. Erik Rodner, who nursed my interest in various aspects of machine learning and who showed me how to back up new ideas with sophisticated and convincing mathematical formulations. Over the past years, Erik was a fountain of inspiration and – most of all – a friend. Finally, I am thankful for the support of Prof. Trevor Darrell, Ph.D. for the unique opportunity to join his vision group in Berkeley and to benefit from his inspiring atmosphere.

Dear reviewers: Without any question, I am truly indebted to the three reviewers of this thesis, Prof. Dr. Joachim Denzler, Prof. Dr. Joachim Hornegger, and Prof. Dr. Trevor Darrell, for their tremendous voluntary effort in reviewing this (quite comprehensive) book.

Dear co-authors: During my time in the Computer Vision Group Jena (CVGJ), I was happy to work in a group of great people with whom I published in a variety of topics. Special thanks go to Paul Bodesheim, Dr. Erik Rodner, Dr. Björn Fröhlich, Johannes Rühle, Dr. Michael Kemmler, Christoph Käding, Christoph Göring, and Marcel Simon. More than with anyone else, this thesis is connected with Paul Bodesheim with whom I shared a long way starting from our first semesters in Jena. He contributed as tandem partner to many aspects in our lifelong learning projects, and we loved to dive into the challenging area of novelty detection. Björn introduced me to the interesting topic of semantic segmentation and led my first baby steps in climbing. Johannes taught me the art of software design as well as the benefits of cmake, challenged me multiple times at the dart board, and started great jam sessions in the RoboLab. Micha guided my first steps in machine learning and frequently surprised us with unknown and stunning mathematical details. Christoph joint our attempts in discovering the mysteries of lifelong learning and became the father of WALL. Together with Chris G. and Marcel, I spent hours in watching birds, thereby pretending to do “fine-grained recognition research”. Many thanks to all of you for our great time together!

Dear colleagues: The CVGJ has always been and still is a great and inspiring place for researchers. This is mainly caused by a fantastic group which is build from fantastic people. Although I had not the pleasure of joint projects with all group members, everybody contributed in his or her own way to the success of

this Ph.D. thesis (*e.g.*, with endless scientific discussions as well as enjoyable social events). Special thanks go to Dr. Marco Körner and Sven Sickert (we have been a great teaching team!), to Micha Koch and Dr. Marcel Brückner (thanks for your inspiring humor), to Dr. Ferid Bajramovic (what a great bouldering partner), to Dr. Daniel Haase (what happens in Gatlinburg stays in Gatlinburg, and Cincinnati Reds!) to Dr. Yanira Guanche (I truly enjoyed our endless dart sessions, and wishing you all the best for the upcoming years with MSCJ), to Cornelia Müsse (“it’s not magic, it’s a secretary”), and to my supervised students (most importantly Alena Schadt, Martin Thümmel, Anastasia Afthakova, and Alex Schultheiss).

Dear supporters: I have been extremely lucky that I received support from various sides. One important aspect is funding, and I am thankful for the support by the German Academic Exchange Service (DAAD), the Computer Vision Foundation (CVF), and the Volkswagenstiftung. On the social level, my thanks go to all colleagues from the CVGJ for more than five exciting years. Furthermore, I am thankful for Jena’s Mensa filling my hungry stomach without waste of time. Regarding intellectual exchange and input from people outside of Jena, I am especially thankful to colleagues at Carl-Zeiss (most importantly Christian Wojek), colleagues at Daimler (David Pfeiffer, Timo Rehfeld, Markus Enzweiler, and Uwe Franke), colleagues at Desko (most importantly Tobias Langhammer), as well as Christoph Lampert and his team in Vienna at IST Austria. Finally, I am thankful for my time as research coordinator at the Michael Stifel Center Jena for Data-driven and Simulation Science (MSCJ) and the stimulating atmosphere which was created by a group of great people from highly diverse research areas.

Dear family: Besides all the previously mentioned aspects, my deepest thanks go to my family for their guidance during the early stages of my life, their support during my Ph.D., and their love and warmth. This especially applies to my parents (you build the basis for making me able to get a Ph.D.) as well as to my parents in law (thanks for becoming a second family and for showing me the Alps as the perfect place for recovering a great work-life-balance).

Dear love: Last but not least, I want to say “Thank you, thank you, thank you” to my wife and friend Stefanie. Thank you for your never-ending love, for your support especially after paper rejections and before submission deadlines, for cheering me up, for the interest and trust in my work, and for the multitude of inspiring discussions.

This thesis would not have been possible without you!

Abstract

Recent breakthroughs of machine learning and visual recognition are fundamentally impacting our today's society. Two impressive show-cases are autonomous vehicles in automotive industry or smart diagnosis systems in modern health care. However, current solutions are static by design, *i.e.*, once they are trained, they heavily rely on non-changing environments and tasks. In contrast, humans are incredibly successful in adapting to new environments by asking appropriate questions, in recognizing previously unseen objects, and in interactively expanding their knowledge over time. In this thesis, we draw inspiration from human learning by transferring individual aspects to continuously learning systems and introduce the lifelong learning cycle as a combination of involved topics.

Replacing static with continuously learning systems poses a multitude of challenges and interesting questions. As a first step towards lifelong learning systems, we provide solutions to four involved areas: large-scale learning, incremental learning, active learning, and feature learning. In the area of large-scale learning, we show how to learn Gaussian process models from large labeled datasets by exploiting inherent properties of histogram intersection kernels. Our techniques overcome complexity limitations of common kernel-based methods with respect to memory demand and computation time and are applicable to all stages of inference, including training, optimization of hyperparameters, classification, and uncertainty prediction. To also incorporate new data over time, we additionally present efficient incremental learning techniques for different aspects of inference with Gaussian process models.

We further contribute to the field of active learning, which is similar to the dialog of a child's questions and its parent's answers. Using active learning, a lifelong learning system can decide for which image with unknown content it shall request a label such that the information gained by the system is maximized. Based on existing selection criteria, we derive efficient realizations for the specific choice of Gaussian process models. In addition, we introduce the new concept of estimated model output changes which approximates the estimated reduction of empirical risk. Finally, we extend our techniques to multi-class scenarios with unnameable instances to handle objects which are even unknown to the expert-in-the-loop.

To represent new object categories appropriately, we finally introduce techniques for mining exemplar-specific patch features. These representations can be learned in a supervised or unsupervised manner and correspond to mid-level features like object parts. Furthermore, we show how to exploit expensive expert annotations, *e.g.*, manually labeled part positions in animal images. The presented

non-parametric technique allows transferring part annotations to previously unseen objects. Thereby, object representations can be focused on regions which are known to be relevant according to application experts.

Our presented techniques are extensively evaluated on several established benchmarks, including applications in scene categorization, object recognition, face identification, digit recognition, and bird identification. The obtained results clearly demonstrate the benefit of our techniques compared to established approaches. Furthermore, they give insights for realizing the next generation of lifelong learning systems.

Zusammenfassung

Der technologische Fortschritt in den Forschungsbereichen des Maschinellen Lernens und der Maschinellen Visuellen Wahrnehmung beeinflusst maßgeblich unsere heutige Gesellschaft. Zwei beeindruckende Beispiele sind hoch- oder gar voll-automatisiert fahrende Fahrzeuge sowie intelligente Diagnosetools zur Unterstützung von Ärzten im modernen Gesundheitswesen. Aktuelle Lösungen sind jedoch statisch konzeptioniert. Dies impliziert, dass sie nach abgeschlossener Konfigurationsphase von einer konstanten Umgebung und gleichbleibender Aufgabenstellung abhängen. Im Gegensatz zu diesem statischen Design steht unsere menschliche Fähigkeit, kontinuierlich Wissen zu akkumulieren. Zuverlässig können wir unbekannte Objekte identifizieren, wir können uns durch das Stellen geeigneter Fragen in neuen Umgebungen zurechtfinden und wir können interaktiv unser bereits erworbenes Wissen erweitern und geeignet repräsentieren. In der vorliegenden Arbeit übertragen wir Aspekte dieser beeindruckenden Fähigkeit auf kontinuierlich lernende Systeme und fassen diese im Zyklus des lebenslangen Lernens zusammen.

Das erfolgreiche Ersetzen von statischen durch kontinuierlich lernende Systeme birgt eine Reihe von Herausforderungen und Fragestellungen. Als einen ersten Schritt zu lebenslang lernenden Systemen stellen wir in dieser Arbeit Beiträge in vier betroffenen Forschungsbereichen vor: Lernen mit großen Datenmengen, inkrementelles Lernen, aktives Lernen, und Lernen von Objektrepräsentationen. Im ersten Bereich erarbeiten wir Techniken, um Gauss-Prozess-Modelle auch auf sehr großen Datensätzen trainieren zu können. Unsere Verfahren nutzen Eigenschaften des Histogramm-Intersection-Kernels und überwinden die üblichen Limitierungen dieser Modelle mit Blick auf Rechenzeit und Speicherbedarf. Insbesondere erlauben die Techniken Fortschritte für alle Anwendungsphasen: initiales Training, Optimierung von Hyperparametern, Klassifikation neuer Beispiele, und Unsicherheitsschätzung. Um auch neues Wissen über die Zeit hinweg adäquat zu integrieren, präsentieren wir zudem effiziente Verfahren zum inkrementellen Lernen von Gauss-Prozess-Modellen in verschiedenen Formulierungen.

Wir stellen zudem Beiträge im Bereich des aktiven Lernens vor, welches dem Frage-Antwort-Dialog von Kind und Eltern ähnelt. Aktives Lernen versetzt lebenslang lernende Systeme in die Lage, gezielt aus einer Menge von Bildern unbekannten Inhalts einzelne Bilder so auszuwählen, dass das Erfragen von deren Inhalt den Wissenszuwachs des Systems maximiert. Zunächst leiten wir dazu effiziente Verfahren aus existierenden Kriterien mit der Verwendung von Gauss-Prozess-Modellen her. Zudem führen wir das neue Konzept der erwarteten Modell-Ausgabe-

Änderungen ein, welches die erwartete Reduktion empirischer Fehler approximiert. Schließlich erweitern wir die Verfahren, sodass diese auch in Multi-Klassen-Szenarien mit nicht benennbaren Objekten anwendbar sind.

Um neue Objektkategorien angemessen zu repräsentieren, stellen wir eine Technik zum Lernen von Exemplar-spezifischen Patch-Merkmalen vor. Diese Repräsentationen können sowohl überwacht als auch unüberwacht gelernt werden und korrespondieren zu Merkmalen mittleren Abstraktionsgrads wie etwa Objektteilen. Zudem zeigen wir Möglichkeiten zur Verwendung von zusätzlichen Bild-Informationen, wie sie etwa für Tierbilder in Form von markierten Positionen relevanter Schlüsselpunkte vorliegen. Da diese Annotationen von Experten erstellt werden müssen, sind sie häufig nur für geringe Datenmengen verfügbar. Unser nicht-parametrisches Verfahren erlaubt das Transferieren dieser Informationen auf bisher ungesehene Objekte. In Folge dessen können Objektrepräsentationen auf Objektteile fokussiert werden, welche von Experten als relevant angesehen werden.

Alle vorgestellten Verfahren werden umfangreich auf verschiedenen, etablierten Datensätzen ausgewertet. Die Anwendungsszenarien reichen dabei von Szenenkategorisierung, über Objekterkennung, Gesichtsidentifikation und das Lesen von Ziffern bis hin zur Unterscheidung verschiedener Vogelarten. Unsere qualitativen und quantitativen Resultate bestätigen experimentell die klaren Vorteile der vorgestellten Verfahren im Vergleich zu etablierten Ansätzen. Darüber hinaus bilden die gewonnenen Erkenntnisse die Grundlage zur Realisierung der nächsten Generation lebenslang lernender Systeme.

Contents

1	Introduction	1
1.1	Understanding Images – The Story-telling Machine	1
1.2	Lifelong Learning	3
1.3	Organization of This Thesis	8
1.4	Overview of Contributions	9
2	Machine Learning for Visual Object Recognition	13
2.1	Useful Concepts in Machine Learning	13
2.1.1	Problem Definition	14
2.1.2	The Frequentist Perspective: Risk Minimization	16
2.1.3	The Bayesian Perspective: Probability Maximization	24
2.1.4	Techniques for Classification	30
2.1.5	Machine Learning with Gaussian Process (GP) Models	51
2.2	Image Representations for Visual Object Recognition	73
2.2.1	Local Features	73
2.2.2	Bag-of-visual-words Models	75
2.2.3	Deep-learned Image Representations	78
2.2.4	Parts and Attributes	81
2.3	On Lifelong Learning in Computer Vision Scenarios	83
2.3.1	“Lifelong Learning” Understood as Transfer Learning	85
2.3.2	Open-space Recognition	86
2.3.3	Continuous Learning of Commonsense Knowledge	88
3	Large-scale Learning with GP Models	91
3.1	Related Work on Large-scale Learning	92
3.1.1	Large-scale Learning with Various Models	92
3.1.2	Large-scale Learning with Kernel SVMs	93
3.1.3	Large-scale Learning with Gaussian Process Models	94
3.2	Towards Exact and Scalable GP Models	96
3.2.1	Histograms and Histogram Kernels	96

CONTENTS

3.2.2	Efficient Computations of Classification Decisions	99
3.2.3	Efficient Model Training with (Generalized) Histogram Intersection Kernels	106
3.3	Efficient Optimization of Hyperparameters	110
3.3.1	An Efficiently Computable Upper Bound of the Negative Log Likelihood	110
3.3.2	Gradient-free Optimization	113
3.4	Efficient Variance Computation with Selectable Precision	115
3.4.1	Exact Computations of the Predictive Variance	116
3.4.2	Approximating the Predictive Variance with Adjustable Precision	117
3.4.3	A Coarse Approximation of the Predictive Variance with Histogram Intersection Kernels	119
3.5	Semantic Segmentation with GP Models	122
3.5.1	A Brief Introduction to Semantic Segmentation	122
3.5.2	Efficient Semantic Segmentation with GP Models	124
4	Incremental Learning with GP Models	127
4.1	Related Work on Incremental Learning	128
4.1.1	Incremental Learning for Various Models	129
4.1.2	Incremental Learning for Gaussian Process Models	129
4.2	Incremental Learning with Plain GP Models	130
4.2.1	Efficient Updates of the Kernel Matrix	130
4.2.2	Efficient Updates of Decompositions of the Kernel Matrix	137
4.2.3	Efficient Updates of the Weight Vector	140
4.3	Efficient Incremental Learning with GP and Histogram Intersection Kernels	141
5	Active Learning for Visual Object Recognition	145
5.1	An Introduction to Active Learning	146
5.1.1	Problem Definition	146
5.1.2	Variants of Active Learning Scenarios	149
5.2	Related Work on Active Learning	153
5.2.1	Rapid Exploration of Feature Space	153
5.2.2	Highest Model Uncertainty	154
5.2.3	Largest Estimated Model Change	154
5.2.4	Largest Estimated Reduction of Estimated Error or Entropy	155
5.3	Efficient Active Learning with GP Models	157
5.3.1	Existing Strategies for Active Learning with GP Models	157
5.3.2	Fast Active Learning with GP Models and Histogram Intersection Kernels	160

5.4	Active Learning Based on Estimated Model Changes	161
5.4.1	Estimated Model Changes (EMC)	161
5.4.2	Efficient Computations of EMC with GP	162
5.5	Active Learning Based on Estimated Model Output Changes	166
5.5.1	Estimated Model Output Changes (EMOC)	167
5.5.2	Efficient EMOC with GP Label Regression	172
5.5.3	An Efficiently Computable Approximation of EMOC	173
5.6	Active Learning for Scenarios with Multiple Classes and Unnameable Instances	176
5.6.1	Active Learning for One-vs-all GP Models	176
5.6.2	EMOC for Multi-class Scenarios	177
5.6.3	Active Learning With Unnameable Instances	179
6	Learning of Mid-level Features	183
6.1	Related Work on Feature Learning	183
6.1.1	Codebook Learning	184
6.1.2	Joint Learning of Features and Classifiers: Architectures for Deep Learning	186
6.1.3	Discovery of Mid-level Features	188
6.1.4	A Side-note on Fine-grained Recognition	190
6.2	Learning of Mid-level Representations	191
6.2.1	Identifying Region Candidates for Mid-level Features	192
6.2.2	From Region Candidates to Patch Detectors	195
6.2.3	Convolutional Unsupervised Bootstrapping of Patch Detectors	197
6.2.4	From Detector Responses to Image Representations	199
6.2.5	Asymptotic Computation Times for Patch Discovery	201
6.3	Exemplar-specific Representations via Local Learning	203
6.3.1	Related Work on Local Learning	204
6.3.2	On Finding Similar Objects	205
6.3.3	Asymptotic Computation Times for Exemplar-specific Patch Discovery	207
6.4	Transferring Semantic Part Annotations	207
6.4.1	Parametric Models for Part Annotation Transfer	207
6.4.2	Non-parametric Models for Part Annotation Transfer	210
7	Experiments	213
7.1	Experimental Setup	213
7.1.1	Overview of Datasets Used	213
7.1.2	Evaluation Metrics	218
7.2	Experiments on Large-scale Learning with GP Models	223
7.2.1	Prelude – GP Label Regression versus GP Classification	223

CONTENTS

7.2.2	Experimental Analysis on Large-scale Classification	226
7.2.3	Experimental Analysis on Hyperparameter Optimization	238
7.2.4	Experimental Analysis on Uncertainty Prediction	246
7.2.5	Experiments on Semantic Segmentation with Scalable GP Models	250
7.3	Experiments on Active Learning for Visual Object Recognition	254
7.3.1	Experiments in Binary Classification Scenarios	255
7.3.2	Experiments in Multi-class Scenarios with Unnameable Instances	269
7.4	Experiments on Learning of Mid-level Representations	288
7.4.1	Part Annotation Transfer for Fine-grained Recognition	288
7.4.2	Mid-level Features for Fine-grained Recognition	295
7.4.3	Exemplar-specific Representations via Local Learning	300
7.4.4	Combining Mid-level Features and Semantic Parts	307
8	Conclusions	311
8.1	Summary and Thesis Contributions	311
8.2	Future Work and Open Questions	315
A	Contributions Beyond the Scope of This Thesis	321
B	Proofs	327
B.1	Quantization Error Analysis	327
B.2	A Sharper Quantization Error Analysis	328
B.3	An Upper Bound for a Matrix's Log-determinant	330
B.4	Bounds on Quadratic Forms	334
B.5	Closed Form Updates of Weight Vector α for GP Regression	335
C	Overview of Experimental Findings	339
C.1	Large-scale GP Models: Summary of Findings	339
C.2	Active Learning: Summary of Findings	341
C.3	Mid-level Features: Summary of Findings	342
	Bibliography	345
	List of Figures	381
	List of Tables	387
	List of Definitions, Theorems, and Lemmas	389
	Notations	391

CONTENTS

Acronyms	395
Index	399
Curriculum Vitae	405
List of Own Publications	413

CONTENTS

1 | Introduction

1.1 Understanding Images – The Story-telling Machine

Tens of thousand of years ago, our ancestors struggled with their survival every single minute. In those times, a wrong decision could make the difference between food for a week – or death. The ability to quickly analyze their environment was one of the most crucial abilities for surviving, *e.g.*, by predicting the current locations of prey and predators. In consequence, a successful and efficient perception of the surrounding is assumed to be one of the key aspects of high-level life forms (see (Bruce et al., 2003) for a detailed introduction). If based on visual sensors, this ability is referred to as *visual perception*. It is therefore not unexpected that image understanding is also considered as one of the key steps towards artificial intelligence.

The Human Ability to Describe Images: Let us inspect the image in Fig. 1.1 as an illustrative example for the astonishing human ability of visual perception. What would be your answer on “What is in the image?”

Given only a rough glance, we would most likely start by naming the overall layout of the scene as ‘outdoor’ and mention that we see a person facing the camera. After some more seconds, we would probably start naming striking objects, *e.g.*, an iron tool, sunglasses, or a grinding stone. However, the human ability to visually analyze scenes and images goes far beyond pure object recognition. Quite naturally, we could start putting objects into perspective. As an example, we would mention that it is Alex who can be seen *in front of* a building. Furthermore, we would note that he is standing *next to* a grinding wheel, *holding* a tool. With yet more time allowed, we could even start reasoning about non-visible scene properties, *e.g.*, that it is a warm place (based on the style of clothing), that the sun is shining from the top left (due to the length and orientation of the shadow), or that Alex’ wife is taking the picture (since he is smiling and is wearing a ring).



Figure 1.1: *Given an image like the one above, which story would you tell?*

To tell a long story short, we would become a story-teller narrating a description of almost infinite length. A visualization of possible descriptions is given in Fig. 1.2.

Towards Image Understanding: For several decades, computer vision researchers aim at reproducing some aspects of this incredible perception ability. Notably, within the last years, several great breakthroughs have been achieved by our community. As an example, automated face recognition in consumer cameras or pedestrian detection in today's cars is as natural to us as the existence of smart phones or the world-wide web (see (Viola and Jones, 2004) and (Enzweiler, 2011) for early results in these application areas).

In Figs. 1.2a to 1.2f, we listed several research fields of computer vision to obtain an impression of today's computer vision capabilities. The areas are roughly ordered by an increasing level of complexity and a decreasing amount of available solutions. As recent show-cases, we refer the interested reader to the following publications: (Zhou et al., 2014) for scene recognition as required in Fig. 1.2a, (Zheng et al., 2015) for person identification as required in Fig. 1.2b, (Girshick, 2015) for object detection as required in Fig. 1.2c, (Long et al., 2015) for semantic segmentation as required in Fig. 1.2d, (Lee et al., 2015) for reasoning about scene properties as required in Fig. 1.2e, and (Chao et al., 2015) for reasoning about interactions of persons and objects as required in Fig. 1.2c and Fig. 1.2e.

As can be seen, our current techniques can be applied in a wide range of application scenarios. However, latest approaches require massive amounts of data and computational power. Furthermore, many aspects of visual perception are still unsolved and rarely understood at all. We believe that solving the complex aspects

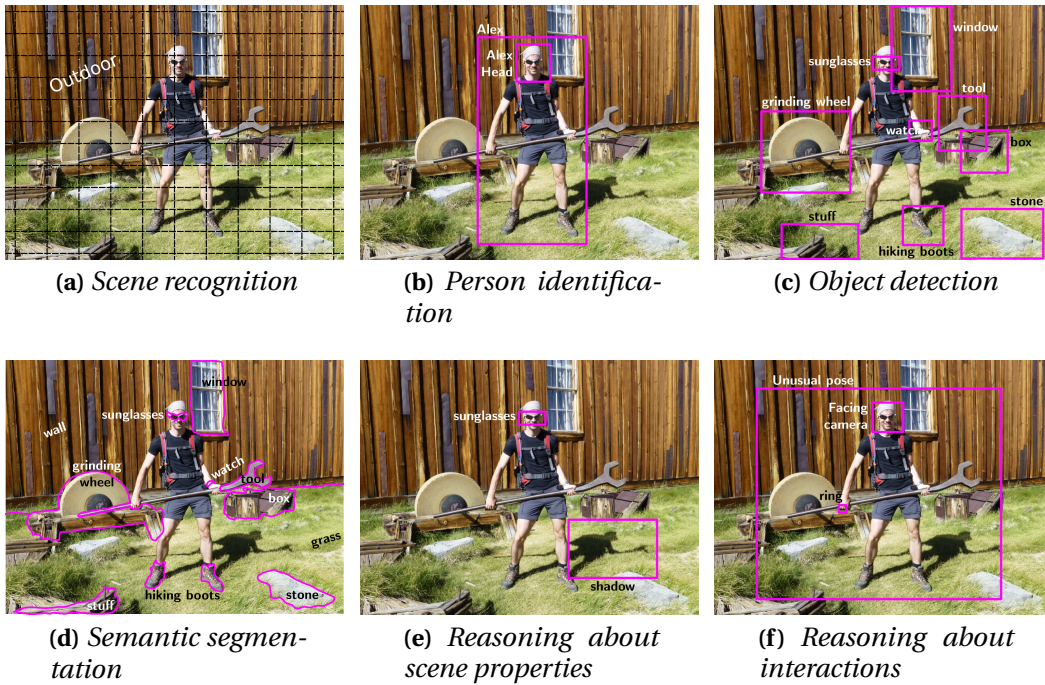


Figure 1.2: Given an image, humans can easily create descriptions of almost infinite length, covering the rough scene layout, persons involved, relations of objects, and even implicit scene properties or estimated actions. Computer vision researchers aim at reproducing some aspects of this ability, to finally build a story-telling machine which is able to analyze a given image, to understand its content, and to create a semantically rich description thereof.

of visual perception will be an important step towards a *story-telling machine* and will ultimately allow us to build smart machines that assist us in every-day life.

1.2 Lifelong Learning

Limitations of Static, Closed-world Systems

As briefly highlighted in the last section, computer vision already lead to breakthroughs in a variety of application areas: ranging from automated address readers to automated license plate recognition, from smart cameras to smart TV screens, or from medical assistance systems to driver assistance systems. Given these successful results, we can indeed ask ourselves: “Are not our recognition systems already powerful enough, capable of reliably representing thousands of visual semantic categories?” A daily growing set of problems can be tackled by powerful algorithms,

and our community is developing new algorithms in an astonishing speed.

Unfortunately, the majority of our approaches is built upon one of the main postulates for pattern recognition: the so-called *closed-world assumption* as introduced by Niemann (1989). In consequence, most of our today's techniques require fixed sets of annotated images or videos to *learn* a solution for a given task. In other words, *every* relevant category or event needs to be recorded and manually verified in advance. Furthermore, recorded data is required to be representative for the tackled problem, *e.g.*, to contain every possible variation, illumination, or viewpoint for known categories. Intuitively, these requirements result in huge annotation efforts and easily increase computational complexity. In the following, we provide arguments why the closed-world assumption is not yet a perfect solution for many of our dynamic real-world scenarios.

Towards Open-world Lifelong Learning: Let us therefore take a step back and reflect how children learn. We easily see that the aforementioned assumption is likely to fail in almost any real-world scenario. Humans do not know *all* possible categories from the very beginning. Even worse, the set of categories is not fixed in our daily changing life. As a direct consequence, humans need to be able to easily adapt their knowledge, to incorporate new information over time, and to re-estimate previous decisions. In other words, humans are able to learn during their entire life span, which we refer to as *lifelong learning*. If we are truly interested in adaptive system which assist us in every-day scenarios, we will sooner or later need to transfer this ability to machines as well.

New Applications in Industry and Science: A successful realization of lifelong learning systems in open-world scenarios could lead the path to a wide range of new application areas. As an almost trivial example, individually adapting cars would no longer need to obtain expensive software or firmware updates to include latest changes. Instead, smart cars could intelligently learn changes over time individually tailored to a drivers surrounding. A different application area could grow in intelligent diagnosis systems, which could detect previously unknown diseases, ask medical experts, and incorporate feedback into their knowledge system. As a third example, consider a domestic robot which expands its knowledge from an initially fixed set of concepts to objects individually relevant for your household, *e.g.*, the appearance of your personal pet or your favorite beer brand. Finally, intelligent monitoring systems for biodiversity applications could individually adapt to new surroundings, discover new species, or realize if species become extinct.

Preliminaries for Lifelong Learning Systems

In order to realize the aforementioned lifelong learning systems, it is insightful to think closer about how humans learn. Perhaps one of the most important aspects in this complex process is the ability for interactive *supervised learning*. In this way, children interact with their parents or teachers by asking questions and receiving information to expand their knowledge. This interplay will be of central importance for this thesis.

In the following, we introduce several key aspects which we believe are most important for realizing lifelong learning systems. We marked topics with an asterisk for which we present contributions in this thesis. A summary of our contributions is given in Section 1.4.

Large-scale Learning*: First of all, note that “learning never stops” according to frequently used idioms. Instead, humans receive a never-ending stream of visual impressions during their life. Similarly, a lifelong learning system is likely to face a huge number of impressions over months or even years. Thus, we need to be capable of dealing with thousands of examples in order to represent the large diversity of possible categories over time.

Incremental Learning*: Humans receive new information frequently and build memories thereof. Nonetheless, not all previous impressions need to be remembered immediately for storing the newly received information in memory. For machine learning, this translates to the necessity of efficient model updates without learning from scratch every time a new sample arrives.

Human-machine Interactions: When children inspect something interesting, unknown, or objects not yet perfectly understood, they immediately pose a variety of questions to expand their knowledge. Implicitly, they assume that their parents or teachers are trustworthy when presenting “ground truth” information to asked questions. In our scenario of lifelong learning, a certain level of human feedback will be helpful as well. Thereby, the collected knowledge set can be updated with reliable information. This incorporation of expert knowledge into the process of machine learning resembles the typical education role of a teacher or a parent.

Novelty Detection and Discovery: To ensure that experts are not asked for nothing, the system should only ask questions where the answer will most likely provide new information. One way for achieving this goal is to ask questions regarding examples which the system has never seen before. This strategy is directly motivated by common children behavior – a phenomenon, every parent knows just too well

('Mom, what's that?'). Thus, our systems need to be able to reliably identify things they have never seen before.

Active Learning*: An obvious application of the interactive question-answer-dialog would be the excessive annotation of large datasets. Huge unlabeled image collections – like Internet search results or video frames collected during a day – could serve as powerful resources for learning if reliable ground truth annotations were available. Unfortunately, these collections are often too large for complete manual annotation. Furthermore, contained data is often redundant, *e.g.*, images of the same object in almost identical poses will often appear in data collections. Intuitively, redundant sets of images carry little information compared to images which contain a variety of different objects. Besides, not only examples with novel objects can contain valuable information. Instead, annotating examples of known categories which are difficult for the currently trained system can improve the learned class separations. Thus, if we reliably estimated the information which can be gained by annotating a specific example, we could flag truly informative examples for annotation.

Knowledge Transfer: After discovering new categories, it is only natural to describe their appearance by relating them to previously seen individuals, species, or objects. By specifying shared properties, previously acquired knowledge can be transferred which allows learning even from extremely small number of views. A perfect example from biology is an Okapi, which can be described as: "It looks like a giraffe, but it is smaller, it has a shorter neck, and it has a zebra-like coloration". During lifelong learning of machines, this aspect of information sharing becomes similarly important. Given new labeled data, classification problems easily become unbalanced and estimation of model parameters will likely be an ill-posed problem. Transferring knowledge from previously learned categories is essential in these situations.

Domain Adaptation: Similar in spirit is the necessity for adaptation due to changing environments or altered perception. A typical example is the substitution of cameras with different resolutions, *e.g.*, between two car manufacturing series. Simply deleting all data collected with the old camera system and instead re-collecting new labeled data for the new camera is obviously a waste of time and money. Instead, adapting previously learned models to the new modality is a desirable solution.

Exploiting Fine-grained Information*: After reaching a certain level of competence, children do not only ask for names of objects, but for fine-grained infor-

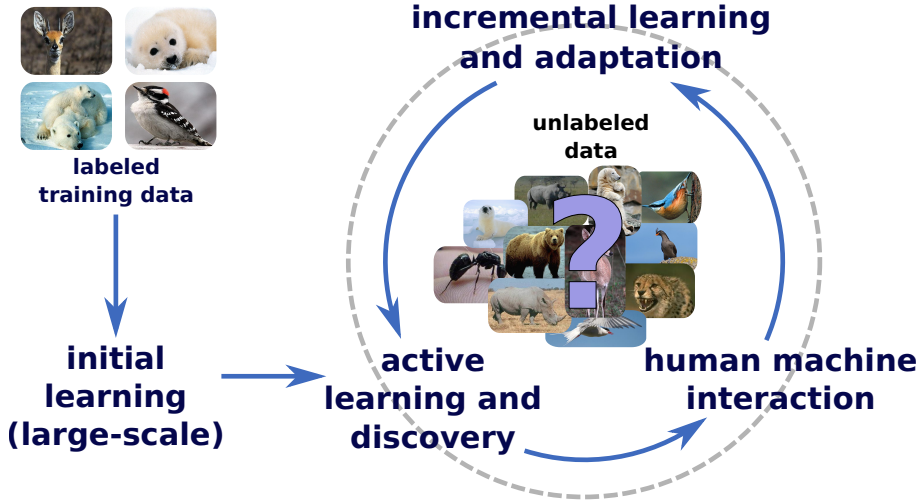


Figure 1.3: Our proposed lifelong learning cycle for visual recognition consists of several important aspects, e.g., large-scale learning, active learning, and adaptation.

mation such as names and properties of object parts. Also for machine learning, we sometimes have access to costly expert annotations for a small set of training data, e.g., hand-labeled points of characteristic body parts such as wings, beak, or neck of a bird. Exploiting these information can guide the process of learning towards semantically important aspects. Furthermore, we can even transfer these annotations to new but similar examples.

Feature Learning*: Last but not least, humans also have an ability which is complementary to the aforementioned process of transferring knowledge to new categories. In fact, new categories are not only described by characteristics shared with known classes, but also by stating their distinctive features. Using these two abilities in combination finally allows for robustly discriminating among thousands of categories learned over time. Thus, also for our lifelong learning scenarios, object representations might be needed to be extended, e.g., by utilizing further feature types, or new attributes. Ideally, those representations are learned automatically, focusing on what makes new categories distinct from known ones.

The Lifelong Learning Cycle: Putting all aspects of lifelong learning together results in a never-ending cycle of collecting data, asking questions, obtaining information, and adapting or updating the current model. In Fig. 1.3, the interplay of those key aspects of lifelong learning systems is visualized. Within this thesis, we make contributions to the majority of previously introduced aspects as will be described in Section 1.4.

1.3 Organization of This Thesis

In this thesis, we aim at scratching at the surface of the vast field of lifelong learning systems to make a first step for closing this gap. We will cover several of the previously introduced key aspects for lifelong learning. The organization of this thesis is outlined in the following.

Basics on Machine Learning, Visual Recognition, and Openended Learning (Chapter 2): To provide a solid background for the subsequent chapters, we will start with a brief introduction to machine learning in Chapter 2. Furthermore, we show how to apply these concepts to visual object recognition tasks using appropriate image representations. A reader experienced in topics such as basic notions of regression and classification, prominent techniques for classification, or frequently used image representations might skip these sections and directly jump to the following ones. Finally, we give an overview on recent attempts to tackle aspects of lifelong learning for visual recognition. The subsequent four chapters are then devoted to four aspects of lifelong learning systems and are based on results presented in own publications.

Large-scale Learning with Gaussian Process Models (Chapter 3): A first step towards lifelong learning is the ability to deal with large amounts of data. In Chapter 3, we present techniques for large-scale learning with *Gaussian process (GP)* models. GPs suffer from cubic computation time and quadratic memory demand in their plain formulation. We show how to significantly reduce required resources for all involved aspects of inference.

Incremental Learning with Gaussian Process Models (Chapter 4): Dealing with a vast number of examples and dimensions does not yet alone tackle problems which occur in changing environments. Instead, models need to be updated frequently when new data is available or existing knowledge needs to be corrected. Therefore, we present techniques for incremental learning with Gaussian process models in Chapter 4.

Active Learning (Chapter 5): Even with fast techniques for training and model updates, annotating all images in never-ending image streams is too expensive for practical use. With a limited budget, we need to specify which of the huge number of unseen images shall be verified by a human expert and then used as additional training examples. In Chapter 5, we present active learning techniques which aim at reducing the expensive and time-consuming process of labeling by selecting only relevant examples. Besides reviewing established approaches, we

derive new criteria for binary and multi-class classification scenarios. In addition, we introduce the problem of unnameable instances which arise in the context of lifelong learning and provide possible solutions.

Feature Learning and Annotation Transfer (Chapter 6): During the last decades, the majority of computer vision systems suffered from the necessity for pre-defined, handcrafted image representations. Thus, expert knowledge and manual fine-tuning was often required to obtain well-performing systems in practice. Although recent deep learning techniques provide solutions to overcome this drawback, they commonly require dedicated hardware and large annotated datasets to be applied successfully. Therefore, we present approaches for learning universal or exemplar-specific representations from data in Chapter 6. Furthermore, we show how to exploit manual annotations to guide feature extraction in fine-grained recognition tasks.

Experiments and Conclusions (Chapters 7 to 8): We evaluate all developed methods on a wide range of tasks and public datasets, including image categorization, scene recognition, and fine-grained recognition of bird species. An overview of the datasets used, details on the conducted experiments, and the findings from extensive experimental evaluations can be found in Chapter 7. To make our findings easily accessible, we provide a one-sentence summary at the end of each conducted experiment. Finally, we conclude this thesis in Chapter 8 and give an outlook on research topics that remain unsolved within the area of lifelong learning.

1.4 Overview of Contributions

In the previous sections, we introduced the lifelong learning cycle and highlighted involved challenges and tasks that need to be solved. Based on the previous outline for this thesis, we shortly introduce our contributions for the tackled aspects.

Contributions to Large-scale Learning with Gaussian Process Models (Chapter 3): We present techniques to significantly reduce required resources for all involved aspects of inference with GP models. Our techniques are based on efficient kernel evaluations using generalized histogram intersection kernels. For many image representations used by the computer vision community, these kernel functions are especially suited. The presented findings for efficient training, classification, and optimization of hyperparameters are based on (Rodner et al., 2012). Furthermore, the extension for fast variance computation and efficient incremental learning was presented in (Freytag et al., 2012b). The chapter closes with an overview of how to apply our techniques to the challenging application of semantic

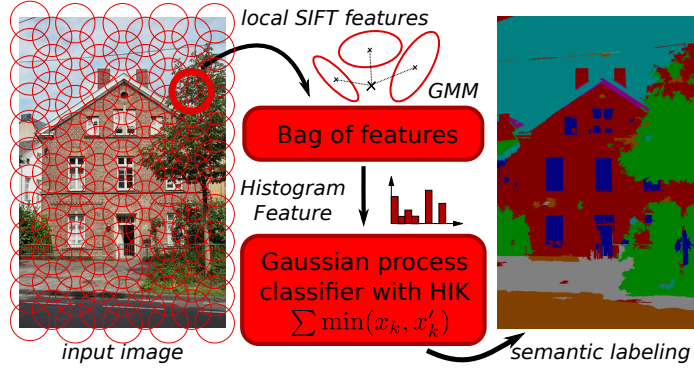


Figure 1.4: Large-scale learning for visual recognition using Gaussian process models. See Chapter 3 for details and our contributions in the area of large-scale learning.

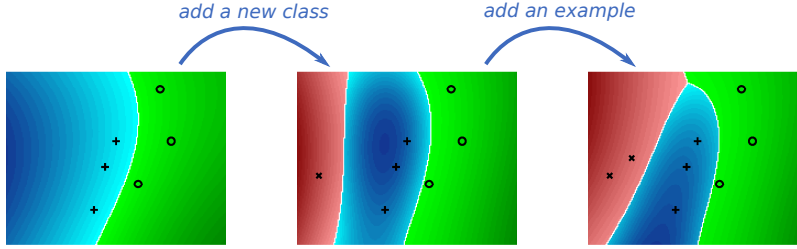


Figure 1.5: Incremental learning of objects and object categories for visual recognition. Details and our contributions in the area of incremental learning can be found in Chapter 4.

segmentation with tens of thousands of examples, which has been presented in (Freytag et al., 2012a) and which is visualized in Fig. 1.4. All aspects are additionally combined in a recent journal article submission (Rodner et al., 2015a).

Contributions to Incremental Learning with Gaussian Process Models (Chapter 4): We present how plain GP regression models can be updated efficiently without training from scratch. The presented techniques allow for adding, changing, or deleting one or several examples in the current training set (see Fig. 1.5 for a visualization). Although those techniques are known for several decades, we present their adaptation to incremental learning of object categories as done in (Lütz et al., 2011, 2013). For generalized histogram intersection kernels, we additionally show how model updates can be done even more efficiently. These results are based on our techniques presented in (Freytag et al., 2012b). Finally, we present how to efficiently predict and compute changes of model parameters and model outputs for GP regression models as introduced in (Freytag et al., 2013) and (Freytag et al., 2014c).

Contributions to Active Learning (Chapter 5): For active learning, we first show how to use our efficient GP techniques of Chapter 3 to speed up existing selection techniques as presented in (Freytag et al., 2012b). To overcome limitations of existing criteria, we derive two new techniques which are based on estimated changes of model parameters and model outputs. The presented techniques are based on our publications in (Freytag et al., 2013) and (Freytag et al., 2014c). A visualization is given in Fig. 1.6. For practical applications, we additionally show efficient implementation strategies for Gaussian process models. We further show how to transfer the developed techniques to multi-class classification scenarios based on the results in (Käding et al., 2015). For these scenarios we analyze nameability problems which arise when applying active learning in lifelong learning scenarios. Finally, we present extensions of our criteria to overcome these limitations.

Contributions to Feature Learning and Annotation Transfer

(Chapter 6): We present three contributions to the area of learning representations from data. We start with an approach to efficiently learn mid-level representations which capture re-occurring visual properties of object categories. The presented technique is published in (Freytag et al., 2014a). A visualization is shown in Fig. 1.7. We additionally drew inspiration from the human behavior to compare unknown objects with similar objects already observed. By analyzing shared and distinct properties, discrimination among most similar categories becomes possible. In this spirit, we show how our mid-level representations can be computed in an exemplar-specific manner for every new test sample as done in (Freytag et al., 2014b). Finally, we follow an orthogonal path and introduce a technique to transfer human-annotated, semantic parts from training images to unseen test images. This

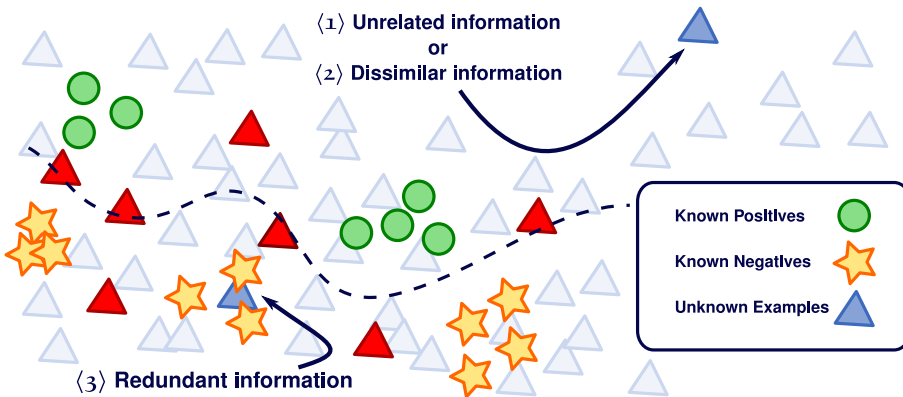


Figure 1.6: Our introduced criteria for active learning allow for estimating the information that is gained by selecting an image for manual annotation. See Chapter 5 for details and our contributions in the area of active learning.



Figure 1.7: *Learning of mid-level representations for visual recognition. Details and our contributions in the area of feature learning and annotation transfer can be found in Chapter 6.*

annotation transfer allows focusing low-level feature extraction on informative image regions as presented in (Göring et al., 2014).

2 | On Machine Learning for Visual Object Recognition

2.1 Useful Concepts in Machine Learning

Humans Love to Categorize: Organizing objects into *categories* is at the core of human nature. This observation is by far not surprising, since grouping similar individual objects into a category allows for focusing on shared, relevant properties. In consequence, categorizing objects dramatically simplifies their representation, memorization, and recognition.

Given the long history of human evolution, different levels of categorization arose quite naturally. Note that a successful categorization with respect to “is dangerous” was essential for the survival of our ancestors. The prediction of “is required for making fire” facilitated controlling physical properties. Identifying individuals within family groups was a precondition for social interactions. Last but not least, the categorization of fruits and plants into different species with individual properties, taste, and consequences after consumption was helpful in every-day life. As an impressive example, Biederman (1987) stated that an average child of six years can already distinguish between $\approx 1,000$ basic level concepts such as chairs or elephants.

Humans Love to Learn New Categories: Going one step further, we observe that humans are never satisfied with the amount of knowledge they already possess. Instead, humans are curious about *learning* yet new objects. Similarly, humans are constantly interested in organizing new objects by specifying properties which shared with known categories as well as separating differences. Comparisons are often based on semantic attributes (*e.g.*, biological taxonomies or social relations), associated interactions (*e.g.*, tools or furniture) or sensory descriptions (*e.g.*, color, taste, or shape).

For learning new categories, humans developed a variety of different techniques. However, the presumably most common principle for human learning is the in-

terplay of teacher and learner. A common example for this way of learning is a parent who presents a new object to his children (“Look at this. . .”) along with the associated class name (“. . .that’s an apple!”).

Categorization and Learning for Machines: As motivated in Section 1.1, imitating or transferring these two abilities of classification and learning to computers would allow for building “intelligent” machines. These smart machines could in turn assist us in every-day scenarios. However, we are not yet close to reach this goal with our current technology. Instead, a multitude of questions is still unanswered and a variety of problems needs to be solved.

Nonetheless, our community successfully developed techniques which partially solve problems under specific constraints and assumptions. In the following, we will provide an overview on established approaches. In (Section 2.1.1), we start by precisely defining the tasks of categorization and learning from a mathematical point of view. Based on these prerequisites, we will then present general concepts and sound theories as well as practical algorithms for learning from collected data and transferring extracted knowledge to new observations (Sections 2.1.2 to 2.1.5).

2.1.1 Problem Definition

Inputs and Outputs: From a mathematical point of view, machine learning is all about discovering yet unknown relations $h : \Omega \rightarrow \mathcal{Y}$ between inputs from space Ω and outputs of space $\mathcal{Y}$. Within this thesis, we can think of $x \in \Omega$ as an image or alternatively its representation as D -dimensional feature vector. With respect to terminology, inputs are usually referred to as samples, patterns, observations, or examples.

We are then interested in predicting relations between inputs and their properties. Interesting aspects could be the estimated size of the contained object, whether or not the image shows a car, or which of a known set of classes the image belongs to. These properties are the outputs of our desired relations h . Thus, we refer to them as outputs are known as target values in the most abstract definition. However, we are mainly interested in classification scenarios in this thesis where $\mathcal{Y}$ is discrete. We reflect this discrete nature by referring to $y \in \mathcal{Y}$ as class label or category label.

Depending on the space $\mathcal{Y}$ of possible outputs, we further refer to the underlying task as *regression* (i.e., $\mathcal{Y} = \mathbb{R}$), *binary classification* (i.e., $\mathcal{Y} = \{-1, 1\}$), or *multi-class classification* (i.e., $\mathcal{Y} = \{1, \dots, M\}$).

Learning with Collected Data: Unfortunately, the space $\mathbb{H}$ of all possible relations $h \in \mathbb{H}$ between inputs $x \in \Omega$ and outputs $y \in \mathcal{Y}$ is large. If we have a given task to

solve, *e.g.*, the classification of images into pedestrians and cars, how can we know that we implement “the correct mapping”? In practice, this question can rarely be answered. Since we are nevertheless interested in predictions, we select a mapping h which seems plausible according to some criteria. To further stress our absence of precise knowledge, we call h a *hypothesis*. In contrast to that, we refer to the unknown, correct mapping of inputs to outputs as *ground truth function* g . Note, however, that g is not available in common machine learning scenarios.

To select a single mapping h , we follow human nature and build-up a memory to guide our selection. Thus, we collect a labeled dataset $\mathcal{L}$ of N training examples $\{(x_i, y_i)\}_{i=1}^N \subset \Omega \times \mathcal{Y}$. For the ease of notation, we will stack training examples x_i in a data matrix $\mathbf{X}$ of dimension $D \times N$, and corresponding labels y_i in a label vector $\mathbf{y}$, respectively. Note that we will indicate unseen test examples with an asterisk in the following sections and chapters, *e.g.*, a data point x_* with (unknown) label y_* .

Learning with Constraints: In addition to collecting data, we can further incorporate our prior knowledge over possible relations into the selection process. One possibility is the restriction to a subset of hypotheses $\mathbb{F} \subset \mathbb{H}$. As an example, if we expect that our data is linearly separable, we can restrict our search to the family of all separations $\mathbb{F}_{\text{linear}}$. In consequence, we only need to search for the hypothesis $\hat{h} \in \mathbb{F}$ which maximizes a given consistency-measure on our observed data $\mathcal{L}$. We can expect that $\hat{h}$ will optimally approximate the true relation g if our assumption of $\mathbb{F}$ was plausible and our collected data was representative.

Regarding terminology, we will refer to the process of selecting $\hat{h}$ as *learning* from data. Furthermore, we will refer to learned hypothesis as *model*, indicated with $\hat{h}_{\mathcal{L}}$ or simply $\hat{h}$ as done previously. To highlight the similarity to the process of human learning, we will further call these scenarios *supervised learning*, since we need to supervise the machine by providing manual annotations.

As mentioned before, the obvious assumption is that the learned model $\hat{h}$ will be satisfactorily close to g given a representative dataset $\mathcal{L}$ and a sufficiently complex set $\mathbb{F}$. This assumption is one of the main postulates for pattern recognition as stated by (Niemann, 1989). However, this assumption likely fails in lifelong learning scenarios. Within this thesis, we will introduce several techniques for resolving this problem.

In the following, we will see two general methods for learning $\hat{h}$ from collected data: (i) the *frequentist perspective* of selecting an optimal hypothesis $\hat{h}$ minimizing the associated risk (Section 2.1.2), or (ii) the *Bayesian perspective* of computing or approximating posterior probabilities and treating latent hypotheses as random variables (Section 2.1.3).

2.1.2 The Frequentist Perspective: Risk Minimization

In this section, we will tackle machine learning from a frequentist perspective. Thereby, we will see how to select a hypothesis $\hat{h}$ with minimal cost with respect to a labeled dataset $\mathcal{L}$. The underlying theory of computational learning will then tell us for which dataset sizes our learned model is likely being close to the best learnable relation h_* . All introduced concepts are valuable prerequisites for several machine learning algorithms introduced in Section 2.1.4. Furthermore, they build the foundation of underlying ideas in the area of active learning (see Chapter 5).

2.1.2.1 Expected Risk Minimization

For a moment, assume that we have already picked a single hypothesis $h \in \mathbb{H}$ from the space of all possible mappings. How can we quantify whether we picked a good hypothesis?

Measuring the Quality of a Hypothesis: If we are interested in the 'quality' of a hypothesis, we need to measure how well the true relation between Ω and $\mathcal{Y}$ is reflected by this hypothesis. Let us therefore define the associated *expected risk* or *expected cost* as follows:

$$\mathcal{R}(h) = \int_{\mathcal{Y}} \int_{\Omega} p(\mathbb{X} = \mathbf{x}, \mathbb{Y} = y) \mathcal{L}(y, h(\mathbf{x})) d\mathbf{x} dy . \quad (2.1)$$

For now, we assume that we know the joint probability distribution $p(\mathbb{X} = \mathbf{x}, \mathbb{Y} = y)$ for inputs $\mathbf{x}$ and targets y . Later on, we will need to estimate these probabilities from collected data, which gives rise to the term “*frequentist perspective*”. For the sake of brevity, we omit explicitly adding the random variable if transferable from the context, *e.g.*, $p(\mathbb{X} = \mathbf{x}, \mathbb{Y} = y)$ is shortened to $p(\mathbf{x}, y)$.

Comparing Outputs: The function $\mathcal{L}(\cdot, \cdot)$ in Eq. (2.1) is a *loss function* $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_*$, $\mathcal{L}(y, y) \stackrel{!}{=} 0$ which measures the cost of mismatching two specific outputs. Note that a loss function always results in non-negative values, *i.e.*, exceptionally good matches are not rewarded additionally. In this thesis, we further assume loss functions to be independent of the actual input. We refer the interested reader to (Schölkopf and Smola, 2002, Chapter 3) for a discussion on input-dependent loss functions. A common example for $\mathcal{L}(\cdot, \cdot)$ is the 0/1-loss for discrete outputs in classification scenarios:

$$\mathcal{L}_{0/1}(h(\mathbf{x}), y) = \begin{cases} 0 & \text{if } h(\mathbf{x}) = y \\ 1 & \text{else} \end{cases} . \quad (2.2)$$

Two common alternatives for regression tasks with continuous outputs are the L_1 -loss:

$$\mathcal{L}_{L_1}(h(\mathbf{x}), y) = |h(\mathbf{x}) - y| \quad , \quad (2.3)$$

and the L_2 -loss:

$$\mathcal{L}_{L_2}(h(\mathbf{x}), y) = (h(\mathbf{x}) - y)^2 \quad . \quad (2.4)$$

Note that in certain domains, asymmetric loss functions might be even more appropriate. As an example, a wrong negative diagnosis of cancer can be significantly worse compared to a wrong positive diagnosis. In addition, if $\mathcal{Y}$ offers a certain structure such as a hierarchical design, then this structure should also be reflected in $\mathcal{L}(\cdot, \cdot)$. As an example, let us think of recognizing birds, where a European woodpecker classified as a North-American woodpecker is still better than classifying it as a penguin.

Selecting the Optimal Hypothesis: From a machine learning perspective, we now seek after a hypothesis $h_* \in \mathbb{H}$ which minimizes the associated expected risk:

$$h_* = \underset{h \in \mathbb{H}}{\operatorname{argmin}} \mathcal{R}(h) \quad . \quad (2.5)$$

With respect to h_* , we can make several comments. First of all, the solution to Eq. (2.5) does not need to be unique. This exemplarily holds if the loss function compares the sign of continuous values against ground truth labels. Since any multiplicative scaling of function values results in the same sign, the corresponding risk of multiplicatively scaled versions of a h remains unchanged.

Besides this ambiguity, we note that h_* is the best (learnable) mapping with respect to the chosen loss function. This aspect is further reflected in the asterisk as symbol. Thereby, we indicate that there is no unseen data $\mathbf{x}_*$ which could be used to find a better hypothesis. It should be further pointed out that the associated risk $\mathcal{R}(h_*)$ does not need to be zero. A simple example is a scenario where inputs are not uniquely assigned to outputs but can instead be probabilistically mapped to different target values.

As pointed out earlier, the concept in Eq. (2.5) of minimizing the expected risk does not yet lead to useful algorithms in practice. The main reasons are the absence of probabilities $p(\mathbf{x}, y)$ as well as the vast number of possible hypothesis in $\mathbb{H}$. In the next section, we will therefore adapt this idea by estimating probabilities from labeled data and by restricting optimization to finite subsets of $\mathbb{H}$.

2.1.2.2 Empirical Risk Minimization

To transfer the previous concept of minimizing the expected risk to practical applications, we simply replace every unknown in Eqs. (2.1) and (2.5) with as much data

and *a priori* knowledge as we can obtain.

Exploiting Labeled Data: In particular, we imitate the process of human learning and rely on a collected memory given as collected and labeled dataset $\mathfrak{L}$. Thereby, we can approximate the unknown probability distribution $p(\mathbf{x}, y)$ with the *empirical distribution* induced by $\mathfrak{L}$:

$$p_{\delta}(\mathbf{x}, y, | \mathfrak{L}) = \frac{1}{|\mathfrak{L}|} \sum_{(\mathbf{x}_i, y_i) \in \mathfrak{L}} \delta_{\{(\mathbf{x}_i, y_i)\}}(\mathbf{x}, y) \quad (2.6)$$

$$= \frac{1}{|\mathfrak{L}|} \sum_{(\mathbf{x}_i, y_i) \in \mathfrak{L}} \delta_{\{\mathbf{x}_i\}}(\mathbf{x}) \cdot \delta_{\{y_i\}}(y) \quad (2.7)$$

Here, the function $\delta_{\mathcal{X}}(\cdot)$:

$$\delta_{\mathcal{X}}(a) = \begin{cases} 1 & a \in \mathcal{X} \\ 0 & a \notin \mathcal{X} \end{cases} \quad (2.8)$$

denotes the Dirac measure induced by a set $\mathcal{X}$. It should be noted that the frequentist's point of view also explains the term *sample* when referring to labeled or unlabeled inputs $\mathbf{x}$, since the process of data collection can be seen as sampling from the unknown distributions $p(\mathbb{X} = \mathbf{x}, \mathbb{Y} = y)$ and $p(\mathbb{X} = \mathbf{x})$, respectively. Given collected data, we can approximate the expected risk in Eq. (2.1) as follows:

$$\mathcal{R}_{\text{emp}}(h, \mathfrak{L}) = \frac{1}{|\mathfrak{L}|} \sum_{(\mathbf{x}_i, y_i) \in \mathfrak{L}} \mathcal{L}(y_i, h(\mathbf{x}_i)) \quad (2.9)$$

which is known as *empirical risk*. A more detailed derivation can be found in (Schölkopf and Smola, 2002, Sect. B.1.3). Regarding terminology, the empirical risk in Eq. (2.9) is referred to as *misclassification rate* for the 0/1-loss. Similarly, the choice of the L_2 -loss results in the *mean squared error*.

Note that we assumed exactly one class label per training example. While multiple class labels or class probabilities are easily possible by treating targets $\mathbf{y}$ as vector variables, these generalizations go beyond the scope of this thesis. In other words, we always assume unique class labels for all inputs within this thesis. Furthermore, for the ease of readability, we will skip the dependency on $\mathfrak{L}$ and instead write $\mathcal{R}_{\text{emp}}(h)$.

Selecting the Empirically Optimal Hypothesis: Besides using a finite dataset, we will further restrict optimization over the set of possible hypotheses in Eq. (2.5) to a certain subset $\mathbb{F}$:

$$\hat{h} = \underset{h \in \mathbb{F}}{\text{argmin}} \mathcal{R}_{\text{emp}}(h) \quad (2.10)$$

Thereby, computational complexity becomes tractable and we can adequately include our prior knowledge. We call the set of possible hypotheses $\mathbb{F}$ model *family*. Furthermore, we refer to the minimizer $\hat{h}$ of the empirical risk in Eq. (2.10) as *learned* model. This naming convention should further highlight the differences to minimizing the exact expected risk in Eqs. (2.1) and (2.5).

The combination of specifying a model family and selecting with respect to collected data gives rise to the name “*frequentist approach*”: we are interested in how *frequent* we need to collect data for a specified model family in order to converge towards h_* with high probability. Thus, frequentists are interested in convergence bounds, laws of large numbers, and in estimates whether or not learning will likely be successful. A different approach is that of Bayesians as we will see in Section 2.1.3.

From a practical point of view, two questions arise quite naturally: (Q1) which dataset $\mathcal{L}$ to pick and (Q2) which model family $\mathbb{F}$ to choose for learning?

Which Data To Collect: The answer to (Q1) is usually as simple as possible: the more the better. A larger dataset likely results in better approximations of induced empirical distributions for unknown distributions. Thus, large datasets are intuitively preferable. However, available memory and computation resources can constrain the maximal usable size of $\mathcal{L}$. For the model family of Gaussian process regression with generalized histogram intersection kernels, we developed techniques to overcome limitations of computation time and memory consumption. Our findings will be presented in Chapter 3.

A second considerable aspect is the required effort for creating a representative dataset $\mathcal{L}$. When a multitude of recorded data is available, but the labeling budget is limited, a subset of data should be selected which is as informative as possible. We propose in Chapter 5 several methods for estimating which examples likely result in high information gains and thus should be preferred for dataset creation.

Which Model Family To Choose: In contrast to (Q1), answering the second question (Q2) is usually more complicated, and often requires expert knowledge. Selecting a specific model family has far-reaching consequences, *e.g.*, memory consumption during learning and model evaluation, asymptotic computation times, asymptotic accuracy bounds, and complexity of learnable relations. As an example, a *support vector machine* (SVM) classifier (see Section 2.1.4) needs roughly $\mathcal{O}(D^3)$ operations during learning and $\mathcal{O}(D)$ operations when evaluating the model on unseen test examples. Furthermore, the family of learnable models is restricted to linear separations of classes in the basic formulation. In contrast, the *K-nearest neighbor* (KNN) classifier (see Section 2.1.4) has no separate learning stage and thus results in zero effort in this phase. However, it requires expensive operations

for model evaluation. Nonetheless, highly non-linear separations can be learned.

A Theory for Computational Learning: The amount of required resources for computation time and memory can usually be easily derived. In contrast to that, the achievable complexity or capacity of learnable models is often difficult to specify. In fact, answering these questions is central to theoretical machine learning research, which is also known as *computational learning theory (COLT)*.

A key framework in COLT research is the *probably approximately correct (PAC)* learning. PAC learning theory is concerned with statements of the type “how likely will future test error be similar to the empirical risk which we evaluated on a finite number of collected examples”. Here, the term “likely” accounts for the probability aspect in PAC and “similar” refers to approximately correct.

The VC Dimension: To measure complexity of a model family, the *VC-dimension* as introduced by Vapnik and Chervonenkis (1971) is a powerful concept. This measure relates the expressive power of a set of models to the input space Ω and its dimensionality D . Loosely spoken, the VC-dimension describes the largest number N_{vc} of examples, such that at least one set $\{\mathbf{x}_1, \dots, \mathbf{x}_{N_{vc}}\} \in \Omega^{N_{vc}}$ under any possible labeling $\{-1, 1\}^{N_{vc}}$ can be separated by members of $\mathbb{F}$ without error. Thus, all possible labelings of the N_{vc} examples can be learned by the specified model family $\mathbb{F}$.

As an example, the VC-dimension of linear models in two dimensions equals three. Note that we can directly verify this fact from our experience, since in two dimensions, any triple of points being not co-linear can be separated by a line for any possible labeling. However, for any set of four points, we can always find a labeling which can not be realized by a straight line. See also Fig. 2.1 for a visualization. More generally, the VC-dimension of linear models in space of dimensionality D is $D + 1$ (Sontag, 1998). We refer to (Sontag, 1998) and (Schölkopf and Smola, 2002) for further derivations and examples.

Note that the definition of *at least one configuration* gives us an upper bound of the feasibility of a model family for collected data: if we collected $N \leq N_{vc}$ examples, the model family might contain a model which is able to separate the classes without error for all possible tasks (induced by given labels). On the contrary, if we collected $N > N_{vc}$ examples, then there will always be a task induced by the given labeling which our chosen model family $\mathbb{F}$ can not solve without error.

Convergence Bounds for Expected Errors: It can be shown that for a specified model family $\mathbb{F}$, learning is only successful with high probability if enough data is available (Schölkopf and Smola, 2002; Vapnik and Chervonenkis, 1971). More precisely, if $N > N_{vc}$, the empirical risk $\mathcal{R}_{emp}$ is likely close to the exact, unknown risk

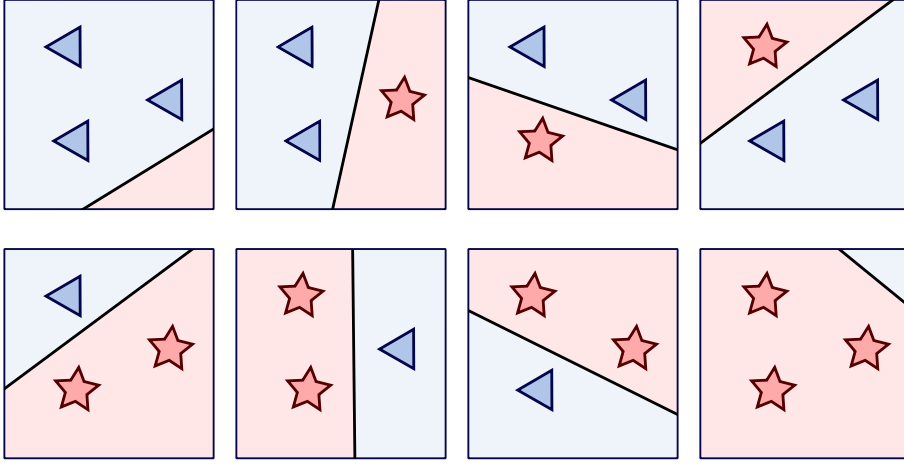


Figure 2.1: Linear models $\mathbb{F}_{\text{linear}}$ in two-dimensional space have a VC-dimension $N_{\text{vc}} \geq 3$, since there is a configuration of three samples (given above) which under any possible binary labeling can be separated error-free by members of $\mathbb{F}_{\text{linear}}$. Since no such configuration of four points exists in $D = 2$, the VC-dimension equals $N_{\text{vc}} = 3$ for $\mathbb{F}_{\text{linear}}$.

$\mathcal{R}$. Thereby, the corresponding minimizers $\hat{h}$ and h_* are likely similar. Following this line of thought, we can directly estimate whether or not our collected training set $\mathcal{L}$ is sufficient for learning once we know the VC-dimension of a given model family $\mathbb{F}$. For an example of possible statements, let us inspect the following theorem which has been presented by Shawe-Taylor et al. (1993).

Theorem 2.1 (A lower bound on sufficient sample size). *The error of approximating h_* with $\hat{h}$ is smaller than ϵ with probability $1 - p_\epsilon$ if at least*

$$N \geq \frac{1}{\epsilon \cdot (1 - \sqrt{\epsilon})} \cdot \left(2 \cdot N_{\text{vc}} \cdot \ln \left(\frac{6}{\epsilon} \right) + \ln \left(\frac{2}{p_\epsilon} \right) \right) \quad (2.11)$$

training samples are available.

Let us investigate this result in more detail. As we can see, the required size of our collected dataset increases with the VC-dimension of our model family $\mathbb{F}$. This is intuitively true, since highly complex models have more (implicit or explicit) parameters which need to be estimated reliably. Similarly, the required number of collected data points increases if we want to obtain low error guarantees, *i.e.*, small ϵ . Finally, N also increases if we are interested in highly confident statements, *i.e.*, large $1 - p_\epsilon$.

While the concept of VC-dimensions only applies to binary classification, generalizations to multi-class scenarios are available. An example is given by Ben-david et al. (1995). For further details, we refer the interested reader to the great

introduction given by Schölkopf and Smola (2002, Chapter 5) or directly to the ground-breaking work by Vapnik (1998).

2.1.2.3 Regularized and Structured Risk Minimization

Given the previous results, we are able to learn a model $\hat{h}$ from a given model family $\mathbb{F}$ such that the resulting empirical risk $\mathcal{R}_{\text{emp}}$ is minimized. However, we still have to face a critical drawback. Directly optimizing the empirical risk as defined in Section 2.1.2.2 is likely to result in *overfitting* to training data.

Overfitting to Data: Overfitting refers to models which work perfect on training data but generalize poorly to unseen test samples. In fact, a model with small empirical risk is not guaranteed to generalize well. As an illustrative example, see the given training data visualized in Fig. 2.2a and the two learned decision functions plotted solid and dotted, respectively. The dotted model $\hat{h}_{\text{complex}}$ results in zero empirical risk $\mathcal{R}_{\text{emp}}(\hat{h}_{\text{complex}}) = 0$. However, we would intuitively doubt the correctness of the learned model due to its high variability and complexity. In contrast, the separation of $\hat{h}_{\text{simple}}$ comes with non-zero empirical risk $\mathcal{R}_{\text{emp}}(\hat{h}_{\text{simple}}) > 0$. Nonetheless, the resulting model is that simple and still supported by the majority of data that we intuitively prefer it over its dotted counterpart.

This observation is closely related to well-known postulates, e.g., the smoothness assumption “*Similar inputs should lead to similar outputs.*”. According to *Occam’s razor*, one should always prefer the simplest solution among all equally good alternatives¹. More precisely, we are seeking for models which can explain our collected data moderately well, but are at the same time expected to generalize well to unknown samples². In the machine learning community, this problem is also known as *bias-variance trade-off*. Here, the term *bias* refers to the *a priori* restriction imposed by the specification of $\mathbb{F}$. Similarly, the term *variance* accounts for how strongly learned models $\hat{h} \in \mathbb{F}$ would differ given small changes in $\mathcal{L}$.

How to Avoid Overfitting During Learning: We have already seen that we can restrict learning to a certain model family $\mathbb{F}$. However, we are often interested in adding further soft penalties on hypotheses in $\mathbb{F}$ which are too complex according to our prior assumptions. As one intuitive option, we could favor models which only use a subspace of the entire input feature space for prediction (sparsity). Alternatively, we could prefer models with smooth changes of output values.

¹See (Blumer et al., 1987) for an introduction of Occam’s razor to machine learning problems.

²An insightful yet controversial discussion on whether simplicity really leads to better generalization expectations is given by Domingos (1999). See also the no-free-lunch theorem introduced by Wolpert (1996).

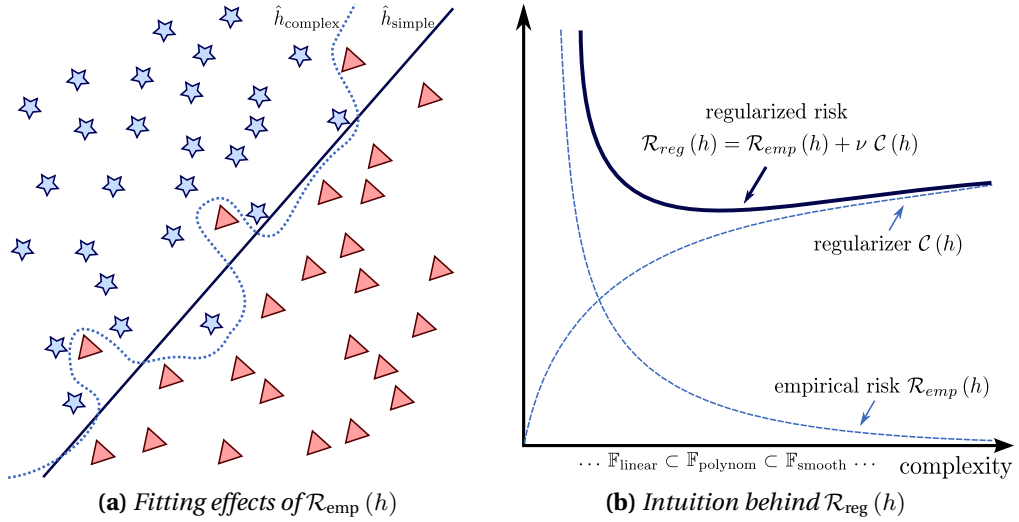


Figure 2.2: Left: drawbacks in empirical risk minimization. Training samples of two classes are given along with two learned models of different complexity (dotted and solid line). While the complex model results in zero empirical risk $\mathcal{R}_{\text{emp}}$, we would intuitively prefer the simple model since we expect it to generalize better to new data. Right: visualization of the regularized risk principle. The regularized risk $\mathcal{R}_{\text{reg}}$ takes the previous observation into account and contains a regularization term $\mathcal{C}$ which penalizes over-complex models.

Adding a monotonically increasing penalty term $\mathcal{C} : [0, \infty) \rightarrow \mathbb{R}$ is often referred to as regularizer. In consequence, the resulting *regularized risk* can be obtained via:

$$\mathcal{R}_{\text{reg}}(h, \mathcal{L}) = \mathcal{R}_{\text{emp}}(h, \mathcal{L}) + \nu \mathcal{C}(h) \quad , \quad (2.12)$$

where ν is a tunable trade-off parameter between capacity and generalization abilities³. The interplay of both terms is visualized in Fig. 2.2b.

In practical applications, we again have to choose a suitable value for ν . Since we usually have no precise knowledge about useful values, we evaluate the regularized risk for a set of possible parameter values, e.g., $\nu \in \{10^{-5}, \dots, 10^5\}$. Note that optimizing the regularized risk in Eq. (2.12) is commonly done on a separate labeled validation dataset to not underestimate the true risk. This strategy is known as *structured risk minimization*. From this concept, the probably most famous technique is the *cross-validation* principle with leave-one-out estimation as a special case. We will see in Section 2.1.4.3 how to apply this technique to optimize parameters of SVM models.

³The trade-off parameter ν is closely related to the parameter C of SVMs (see Section 2.1.4.3) or to the label noise term σ_n^2 of Gaussian process (GP) regression models (see Section 2.1.5)

2.1.3 The Bayesian Perspective: Probability Maximization

In the previous section, we were interested in selecting a hypothesis which matched our prior expectations and was consistent with collected data. To achieve this goal, we tackled machine learning from a frequentist perspective. We will now follow a different path by treating the choice of hypotheses and all involved parameters as random variables. Thereby, we can derive probabilistic distributions over relations between inputs x and targets y . This way of approaching machine learning is known as Bayesian perspective due to the involved computations.

Under the aforementioned frequentist perspective, we introduced learning as risk minimization with respect to a labeled dataset $\mathcal{L}$. In consequence, learning resulted in a single model with smallest associated risk. Learning theory then provided us with statements about the probability that the model learned from finite data $\mathcal{L}$ is close to the optimal relation h_* learned with infinite data. In contrast to that, we will see in this section how (i) our prior assumptions over possible relations are turned into posterior distributions given labeled data, and (ii) how associated outputs of all functions are probabilistically combined into a final decision.

2.1.3.1 Machine Learning from a Probabilistic Point of View

Our original goal was to enable machines to solve visual recognition task. While we required collected training data to learn models, we were ultimately interested in learning to predict target values y_* for unseen examples x_* . In other words: we wanted to *infer* the unknown relation from collected data. From a probabilistic perspective, we are thus interested in computing the conditional probability $p(\mathbb{Y} = y \mid \mathbb{X} = x_*)$. As previously, we omit explicitly mentioning random variables for the sake of brevity.

By definition, the conditional probability equals

$$p(y \mid x_*) = \frac{p(x_*, y)}{p(x_*)} . \quad (2.13)$$

Thus, if we were able to estimate joint probabilities, we could immediately arrive at the intuitive decision strategy of assigning most probable targets to unseen inputs:

$$\hat{h}_{\text{Bayes}}(x_*) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} p(y \mid x_*) , \quad (2.14)$$

which is also known as *Bayes classifier*. Interestingly, we can easily notice that this strategy is optimal in a sense that it minimizes the expected risk $\mathcal{R}$ as defined in Eq. (2.5) for the 0/1-loss. For further details, we refer to (Niemann, 1983, Chapter 4.1).

Discriminative and Generative Approaches: Similar to the observations in Section 2.1.2.1, we again face the problem of unknown joint distributions $p(\mathbf{x}_*, y)$. However, we now aim at estimating probabilities from labeled data $\mathcal{L}$, which is in contrast to the non-probabilistic techniques introduced in Section 2.1.2. To this end, we have two general strategies to choose from: directly estimating $p(y | \mathbf{x}_*, \mathcal{L})$ (*i.e.*, the left hand side in Eq. (2.13)) or applying Bayes' rule and estimating $p(\mathbf{x}_* | y, \mathcal{L})$ instead.

When we decide to directly estimate conditional probabilities $p(y | \mathbf{x}_*, \mathcal{L})$ from data, we “only” need to specify how to estimate these probabilities, *i.e.*, which model family to select for approximating probability distributions. Techniques following this principle are called *discriminative approaches*, since estimated probabilities can directly be used to discriminate among categories in the case of a discrete target space.

An alternative approach is to apply Bayes' rule:

$$p(y | \mathbf{x}_*, \mathcal{L}) = \frac{p(\mathbf{x}_* | y, \mathcal{L}) \cdot p(y | \mathcal{L})}{p(\mathbf{x}_* | \mathcal{L})} \quad (2.15)$$

which factors the joint probability into class-specific likelihood $p(\mathbf{x}_* | y, \mathcal{L})$ and the prior term $p(y | \mathcal{L})$. Here, the prior reflects the estimated *a priori* estimates of class probabilities, whereas the likelihood term accounts for distributions of specific classes. Further applying the sum rule finally leads to:

$$p(y | \mathbf{x}_*, \mathcal{L}) = \frac{p(\mathbf{x}_* | y, \mathcal{L}) \cdot p(y | \mathcal{L})}{\sum_{y \in \mathcal{Y}} p(\mathbf{x}_* | y, \mathcal{L}) \cdot p(y | \mathcal{L})} . \quad (2.16)$$

This strategy is referred to as *generative approach*, since we can generate examples for specified categories given the estimated likelihood function $p(\mathbf{x}_* | y, \mathcal{L})$.

Selecting a Modeling Technique: So far, we only decided *what* to approximate, *i.e.*, whether $p(y | \mathbf{x}_*, \mathcal{L})$ for discriminative approaches or $p(\mathbf{x}_* | y, \mathcal{L})$ for generative approaches. The next step is to specify *how* to approximate corresponding probabilities conditioned on $\mathcal{L}$. This step is often referred to as *selecting a classifier* for discrete $\mathcal{Y}$. Similarly, it is called *selecting a regressor* for continuous $\mathcal{Y}$.

As an example, let us consider parameterized functions $h(\cdot, \vartheta)$, *e.g.*, multivariate normal distributions which form a generative model family (see Section 2.1.4.2). Here, parameters are class-specific mean vector μ_m and covariance matrix Σ_m , *i.e.*, $\vartheta_{\mathcal{N}} = (\mu_1, \Sigma_1, \dots, \mu_M, \Sigma_M)$. Similarly, we could select linear separations as a discriminative approach which are specified by their normal vector and offset, *i.e.*, $\vartheta_{\text{lin}} = (w, b)$. After we selected a classification technique, it remains to estimate useful values θ for all involved parameters ϑ using collected data as will be described next.

With respect to notational clarity, note that ϑ accounts for the *variable* that subsumes all parameters. Thus, the variable ϑ can take specific *values* θ out of the corresponding parameter space Θ . Nonetheless, we again omit the explicit mentioning of $\vartheta = \theta$ and instead write only θ to shorten corresponding formulas.

2.1.3.2 Maximum Likelihood Estimation

When we search for suitable values θ of our model's parameters, we assume that most of the relevant information of $\mathcal{L}$ can be absorbed in θ . From a Bayesian perspective, we thus assume that target values and inputs are independent of $\mathcal{L}$ given a suitable set of parameter values:

$$p(y | x_*, \theta, \mathcal{L}) = p(y | x_*, \theta) \quad \text{and} \quad (2.17)$$

$$p(x_* | y, \theta, \mathcal{L}) = p(x_* | y, \theta) \quad . \quad (2.18)$$

The resulting two-step approach is also known as *plug-in principle* (*i.e.*, parameter estimation followed by prediction using the model with its estimated parameters). As we will see in Section 2.1.4, this principle lies at the core of several machine learning algorithms.

ML Estimates for Generative and Discriminative Approaches: In order to probabilistically determine a plausible set of parameter values θ , an intuitive way is to follow the *maximum likelihood (ML)* principle by maximizing the *likelihood* of θ with respect to the training data itself:

$$\theta_*^{\text{ML}} = \operatorname{argmax}_{\theta \in \Theta} p(\mathcal{L} | \theta) \quad . \quad (2.19)$$

Note that the term likelihood is used differently in the literature. Loosely spoken, it refers to the probability of “some random variable given a fixed parameter setting”. For classification scenarios, these random variables refer to inputs and outputs. Thus, the likelihood quantifies *how well collected data can be explained given the specific parameter values θ in a parameterized function $h(\cdot, \theta)$* .

In consequence, the data likelihood for generative approaches can be expressed as:

$$\theta_*^{\text{ML}} = \operatorname{argmax}_{\theta \in \Theta} p(\mathbf{X} | \mathbf{y}, \theta) \cdot p(\mathbf{y} | \theta) \quad (2.20)$$

$$\stackrel{\text{gen.}}{=} \operatorname{argmax}_{\theta \in \Theta} p(\mathbf{X} | \mathbf{y}, \theta) \cdot p(\mathbf{y}) \quad . \quad (2.21)$$

Note that Eq. (2.21) follows from the independence assumption of class probabilities and parameter values. As we can see, class likelihoods $p(\mathbf{X} | \mathbf{y}, \theta)$ are weighted

by class priors $p(\mathbf{y})$. Therefore, the score to be maximized is also known as *a posteriori estimate with respect to $\mathbf{y}$* when referring to labels instead of parameters.

For discriminative approaches, the likelihood instead accounts for:

$$\boldsymbol{\theta}_*^{\text{ML}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}} p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) \cdot p(\mathbf{X} | \boldsymbol{\theta}) \quad (2.22)$$

$$\stackrel{\text{disc.}}{=} \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}} p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) \quad (2.23)$$

where examples are assumed to be equally likely.

I.I.D. Assumption and Connection to Frequentist's Perspective: A common assumption on the collected dataset is that examples are *independent and identically distributed (i.i.d.)*. As a consequence, the likelihood in (2.19) factorizes to:

$$\boldsymbol{\theta}_*^{\text{ML}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}} \prod_{i=1}^N p(\mathbf{x}_i, y_i | \boldsymbol{\theta}) \quad (2.24)$$

By maximizing the likelihood distribution in Eq. (2.24), we obtain the mode as a point estimate resulting in a single set of parameter values $\boldsymbol{\theta}_*^{\text{ML}}$ which best explain our collected data. A visualization is shown in Fig. 2.3a.

In practice, instead of directly maximizing the data likelihood, we usually minimize the negative log-likelihood for computational efficiency:

$$\boldsymbol{\theta}_*^{\text{ML}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \sum_{i=1}^N -\log p(\mathbf{x}_i, y_i | \boldsymbol{\theta}) \quad (2.25)$$

Interestingly, this reveals a strong connection to the frequentist perspective for discriminative approaches: the minimization of the negative data log-likelihood $\sum_{i=1}^N -\log p(y_i | \mathbf{x}_i, \boldsymbol{\theta})$ is identical to the minimization of the empirical risk if we select $-\log p(y_i | h(\mathbf{x}_i, \boldsymbol{\theta}))$ as loss function.

2.1.3.3 Maximum A-posteriori Estimation

As in the case of empirical risk minimization (see Section 2.1.2.2), directly minimizing the negative log-likelihood in Eq. (2.25) likely results in overfitting to training data. For illustration, we refer the interested reader to the insightful example of curve fitting by Bishop (2006, p. 8).

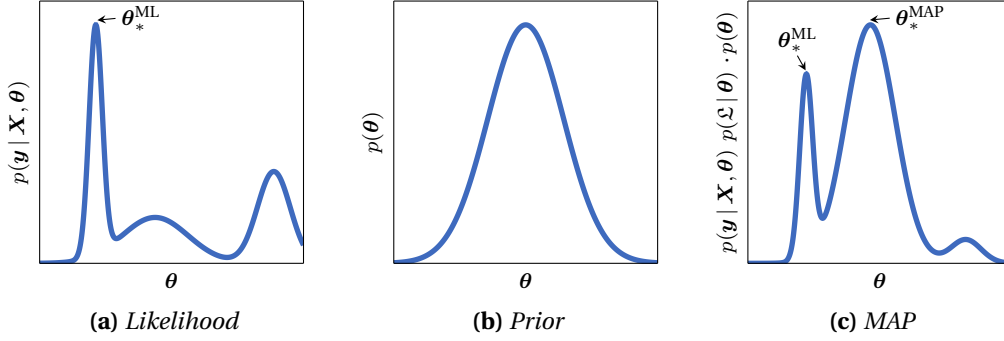


Figure 2.3: Visualization of ML and MAP estimation. While maximizing the data likelihood can likely lead to overfitting (a sharp peak in Fig. 2.3a), incorporating prior assumptions on parameters (Fig. 2.3b) re-weights estimates to better reflect assumptions about the unknown ground truth relation. Thus, the MAP estimation (Fig. 2.3c) is a regularized point estimate of ϑ .

Including Prior Information: We can approach this problem by placing a *prior* $p(\theta)$ on the parameters:

$$\theta_*^{\text{MAP}} = \underset{\theta \in \Theta}{\operatorname{argmax}} p(\mathcal{L} | \theta) \cdot p(\theta) \quad . \quad (2.26)$$

Thus, the obtained point estimate θ_*^{MAP} is the mode of the *posterior* distribution of parameters with respect to collected data, which gives rise to the name *maximum a posteriori (MAP)* estimation. The difference between ML estimation and MAP estimation is further illustrated in Fig. 2.3. Note how the initial data likelihood (Fig. 2.3a) is re-weighted by a prior term on parameter probabilities (Fig. 2.3b) to reflect assumptions regarding the yet unknown ground truth relation.

As in Section 2.1.3.2, we again assume examples being i.i.d., and we further minimize negative log-values:

$$\theta_*^{\text{MAP}} = \underset{\theta \in \Theta}{\operatorname{argmin}} \left(\sum_{i=1}^N -\log p(\mathbf{x}_i, y_i | \theta) \right) - \log p(\theta) \quad . \quad (2.27)$$

Let us inspect Eq. (2.27) in more detail. First of all, we observe that for uniform priors, MAP estimates and ML estimates coincide. Besides, the likelihood term grows with the number of examples, whereas the prior stays constant. In consequence, the MAP estimate converges towards the ML estimate for sufficiently large number of examples, *i.e.*, our prior loses importance given more and more observations. However, data is limited in practice and the choice of a valid prior is often as crucial as difficult.

Again, we notice a strong connection between frequentist and Bayesian perspectives: MAP estimates and regularized risk minimization coincide for accordingly chosen prior and the loss function $-\log p(y_i | h(\mathbf{x}_i, \boldsymbol{\theta}))$. Thus, regularized risk minimization can be seen as MAP estimation from a Bayesian perspective.

2.1.3.4 Marginal Likelihood Estimation

So far, we treated inputs and targets as random variables, and we were interested in selecting a set of parameter values which explain the observed data with highest probability. To obtain ML or MAP estimates, we followed the plug-in principle by assuming that all relevant information of $\mathcal{L}$ can be absorbed in suitable parameters. We then approximated conditional probabilities $p(y | \mathbf{x}_*, \mathcal{L}) \sim p(y | \mathbf{x}_*, \boldsymbol{\theta}_*^{\text{ML}})$ or $p(y | \mathbf{x}_*, \mathcal{L}) \sim p(y | \mathbf{x}_*, \boldsymbol{\theta}_*^{\text{MAP}})$ using the previously found parameter settings $\boldsymbol{\theta}_*^{\text{ML}}$ or $\boldsymbol{\theta}_*^{\text{MAP}}$.

Drawbacks of ML and MAP and How to Solve Them: Selecting only the most probable set of parameter values has an inherent drawback: it neglects any uncertainty during the process of parameter estimation. As an example, imagine a likelihood function which consists of two peaks with similar height. In this situation, the selection of a single point estimate wastes valuable information about equally probable parameter ranges.

Under a full Bayesian perspective, we would instead also treat parameters as random variables rather than maximizing over the parameter space. Let us therefore replace the maximization of *a posteriori* scores in Eq. (2.26) by taking the expectation instead. To this end, we rewrite our originally desired conditional probabilities $p(y | \mathbf{x}_*, \mathcal{L})$ by marginalizing over all possible parameter values $\boldsymbol{\theta}$ from the entire parameter space Θ :

$$p(y | \mathbf{x}_*, \mathcal{L}) = \int_{\Theta} p(y, \boldsymbol{\theta} | \mathbf{x}_*, \mathcal{L}) d\boldsymbol{\theta} \quad (2.28)$$

$$= \int_{\Theta} p(y | \boldsymbol{\theta}, \mathbf{x}_*, \mathcal{L}) \cdot p(\boldsymbol{\theta} | \mathbf{x}_*, \mathcal{L}) d\boldsymbol{\theta} . \quad (2.29)$$

Let us further propose two assumptions: (i) conditional independence of outputs and training examples given parameter values in the first term (as for the plug-in principle), and (ii) that new examples do not influence the likelihood of parameter values with respect to training data within the second term. Then, we can finally obtain the following equality also known as *marginal likelihood*:

$$p(y | \mathbf{x}_*, \mathcal{L}) = \int_{\Theta} p(y | \mathbf{x}_*, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta} | \mathcal{L}) d\boldsymbol{\theta} . \quad (2.30)$$

Since no fixed set of parameter values is required for estimating $p(y | \mathbf{x}_*, \mathcal{L})$, this principle is also referred to as *non-parametric*.

ML and MAP are Special Cases of Marginal Likelihood Estimation: It should be noted that ML and MAP estimates arise as special cases of the marginal likelihood estimate given in Eq. (2.30). This can be easily seen by designing posterior over the parameters $p(\boldsymbol{\theta} | \mathbf{x}_*, \mathcal{L})$ as Dirac delta function δ positioned at $\boldsymbol{\theta}_*^{\text{ML}}$ or $\boldsymbol{\theta}_*^{\text{MAP}}$, respectively:

$$p^{\text{ML}}(y | \mathbf{x}_*, \mathcal{L}) = \int_{\Theta} p(y | \mathbf{x}_*, \boldsymbol{\theta}) \cdot \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_*^{\text{ML}}) d\boldsymbol{\theta} , \quad (2.31)$$

$$p^{\text{MAP}}(y | \mathbf{x}_*, \mathcal{L}) = \int_{\Theta} p(y | \mathbf{x}_*, \boldsymbol{\theta}) \cdot \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_*^{\text{MAP}}) d\boldsymbol{\theta} . \quad (2.32)$$

This additionally highlights the negligence of uncertainty when using point estimates as plug-in parameter settings. Further note that we can also apply Bayes' rule when computing the marginal likelihood in Eq. (2.30):

$$p(y | \mathbf{x}_*, \mathcal{L}) = \int_{\Theta} p(y | \mathbf{x}_*, \boldsymbol{\theta}) \cdot \frac{p(\mathcal{L} | \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(\mathcal{L})} d\boldsymbol{\theta} . \quad (2.33)$$

Thereby, we again observe the interplay of likelihood $p(\mathcal{L} | \boldsymbol{\theta})$ and prior $p(\boldsymbol{\theta})$.

Similar to likelihood maximization, we can also compute the marginal likelihood with respect to training data itself. This estimate can be used to compare different model families without the necessity of separate validation sets as required in structured risk minimization, *e.g.*, during cross validation. Thus, if several model families are possible, the optimal choice from a Bayesian point of view would be the one with highest marginal likelihood. We will see an application for Gaussian process models in Section 2.1.5.7. For a more detailed introduction to Bayesian model selection, the interested reader is referred to (Murphy, 2012, Chapter 5).

2.1.4 Techniques for Classification

In the previous sections, we introduced the two complementary approaches of frequentists and Bayesians. In consequence, we have the required tools at hand to derive different classification techniques by carefully placing assumptions regarding model relations and probabilities. We will restrict ourselves to models used in following chapters to stay self-contained. In addition, we will mainly focus on discriminative techniques as mentioned in Section 2.1.3.2. For further reading, an insightful introduction is given by Kemmler (2013) which nicely highlights connections between different methods. Comprehensive reviews of a wide range of classification techniques can also be found in Bishop (2006) and Murphy (2012).

2.1.4.1 K-Nearest Neighbor Classifiers

Every Assumption Can Be Wrong: Although no assumption should come without motivation, every assumption can turn out to be wrong. This especially holds for assumptions regarding data distributions or probability densities which are at the core of machine learning algorithms. Thus, any algorithm which is built on assumptions can fail in certain situations. As an example, the assumption of normally distributed data can be easily wrong if collected data follows different characteristics.

One important exception which does not risk placing wrong assumptions is to skip every modeling step and instead to memorize all data ever collected. Decisions for new examples can then be done by transferring knowledge from the most similar examples within the memorized collection. In its simplest version, the resulting classification technique is therefore non-probabilistic and discriminative.

Distances and Neighbors: For a more detailed analysis, we require a measure of dissimilarity to be given $d(\mathbf{x}_i, \mathbf{x}_j)$, *i.e.*, a function measuring the distance from $\mathbf{x}_i$ to $\mathbf{x}_j$. Usually, we assume $d(\cdot, \cdot)$ to be a metric, *i.e.*, the following properties hold:

1. $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \Omega$ (non-negative)
2. $d(\mathbf{x}_i, \mathbf{x}_j) = 0 \Leftrightarrow \mathbf{x}_i = \mathbf{x}_j \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \Omega$ (coincident)
3. $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i) \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \Omega$ (symmetric)
4. $d(\mathbf{x}_i, \mathbf{x}_k) \leq d(\mathbf{x}_i, \mathbf{x}_j) + d(\mathbf{x}_j, \mathbf{x}_k) \quad \forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \Omega$ (subadditive).

Well-known examples are the Euclidean distance d_{L_2} and the Manhattan distance d_{L_1} . In Section 2.1.5.4, we will see the concept of kernels which are closely connected to distance functions.

In the simplest case, we now store the training set as-is and leverage the class label of the example being most similar to a new input $\mathbf{x}_*$:

$$\hat{h}_{\text{NN}}(\mathbf{x}_*) = y_i, \quad i = \underset{j \in \{1, \dots, N\}}{\operatorname{argmin}} d(\mathbf{x}_*, \mathbf{x}_j) . \quad (2.34)$$

An intuitive extension additionally smooths labels over a neighborhood of K most similar examples. Thereby, it gives rise to the *K-nearest neighbor (KNN)* algorithm:

$$\hat{h}_{\text{KNN}}(\mathbf{x}_*) = \underset{m \in \{1, \dots, M\}}{\operatorname{argmax}} p_m, \quad p_m = \frac{1}{|\mathcal{S}_K|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_K} \delta_{m, y_i} . \quad (2.35)$$

Here, $\mathcal{S}_K$ is the set of K training examples most similar to $\mathbf{x}_*$ and the Kronecker delta δ_{m, y_i} counts contained examples of each class m .

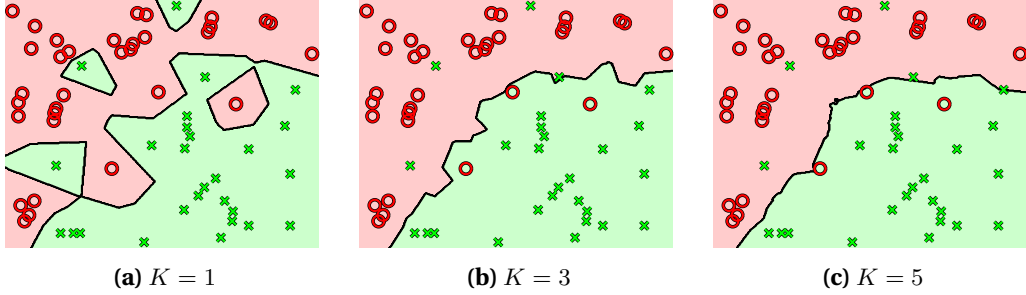


Figure 2.4: Visualization of decision areas for K -nearest neighbor (KNN) classifiers with $K \in \{1, 3, 5\}$. Training examples of two classes are shown as red circles and green crosses, respectively. The resulting decision boundary is indicated as black line.

Advantages and Disadvantages of KNN Approaches: The advantages of this classification technique are obvious: beside its simplicity, the necessity of model assumptions is avoided and no training step is required. For an impressive application of model-free classification with millions of known images, we refer to the work by Torralba et al. (2008). In addition, it was shown by Cover and Hart (1967) that the error rate for $K = 1$ and $N \rightarrow \infty$ converges to a value which is guaranteed to be smaller than twice the Bayes' optimal error. Here, the Bayes' optimal error refers to the minimal error obtained by an optimal classifier h_* using the *a posteriori* decision as introduced in Eq. (2.14).

Besides these obvious benefits, the nearest neighbor technique also comes with severe drawbacks, prohibiting its use in several scenarios. As can be seen in Fig. 2.4, resulting decision boundaries are highly non-smooth. This contradicts one of the most common assumptions in pattern recognition that similar inputs should lead to similar outputs. In addition, only a measure for classification agreement among the K neighbors can be obtained, rather than a valid probabilistic statement for the classification result. Finally, all relevant computations need to be done at test time requiring high memory and computation resources. With respect to the last aspect, techniques for removing likely redundant examples from the training set are usually applied in practice. An example is the so-called condensation techniques as introduced by Hart (1968) which was further refined by Angiulli (2005) and Karacali and Krim (2003).

2.1.4.2 Normal Distributions and Linear Discriminant Analysis

Whereas the K -nearest neighbor technique came without any model assumption, we now look into one of the simplest possible assumptions for classification. We will later apply resulting models in Section 6.2.2 to build detectors individually

trained from single image patches.

Normal Distributions Classifiers: As common in many natural sciences, let us assume that examples of each class are generated from an underlying multivariate Gaussian distribution. Every distribution is parameterized with class-specific mean vector μ_m and covariance matrix Σ_m . For notational convenience, we will write $x \sim \mathcal{N}(\mu, \Sigma)$ to denote that x is assumed to be drawn from the multivariate Gaussian distribution with parameters μ and Σ . We will further denote the probability density of $\mathcal{N}(\mu, \Sigma)$ evaluated at x with $p(x | \mu, \Sigma) = \mathcal{N}(x | \mu, \Sigma)$.

Given such parameters $\theta_m = (\mu_m, \Sigma_m)$, we can directly obtain class-conditional densities in a generative manner (see Section 2.1.3.1):

$$p(x_* | \theta_m) = (2\pi)^{-\frac{D}{2}} \det(\Sigma_m)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (x_* - \mu_m)^T \Sigma_m^{-1} (x_* - \mu_m)\right) . \quad (2.36)$$

As we can see, the plug-in principle nicely applies for normal distribution classifiers, since all information of training examples of class m is absorbed in θ_m . To estimate these parameters, we can again follow the maximum-likelihood principle as introduced in Section 2.1.3.2. By setting the likelihood derivatives with respect to each parameter involved to zero, we can obtain optimal parameters in closed form:

$$\mu_m^{\text{ML}} = \frac{1}{\sum_{i=1}^N \delta_{m,y_i}} \sum_{i=1}^N (\delta_{m,y_i} \cdot x_i) , \quad (2.37)$$

$$\Sigma_m^{\text{ML}} = \frac{1}{\sum_{i=1}^N \delta_{m,y_i}} \sum_{i=1}^N \left(\delta_{m,y_i} \cdot (x_i - \mu_m^{\text{ML}}) (x_i - \mu_m^{\text{ML}})^T \right) . \quad (2.38)$$

Besides this appealing result, we shortly adhere that the ML estimate for Gaussian distributions is not robust to outliers in training data (see (Bishop, 2006, pp.103)).

Let us now assume that all classes are *a priori* equally probable, *i.e.*, $p(y) = \frac{1}{M}$. As a result, we can finally classify new examples by applying the *a posteriori* classification rule as given in Eq. (2.16):

$$\hat{h}_{\mathcal{N}}(x_*) = \underset{m \in \{1, \dots, M\}}{\operatorname{argmax}} \frac{p(x_* | \theta_m) \cdot \frac{1}{M}}{\sum_{m=1}^M p(x_* | \theta_m)} \quad (2.39)$$

$$\propto \underset{m \in \{1, \dots, M\}}{\operatorname{argmax}} \log \det(\Sigma_m)^{-\frac{1}{2}} - \frac{1}{2} (x_* - \mu_m)^T \Sigma_m^{-1} (x_* - \mu_m) \quad (2.40)$$

$$\propto \underset{m \in \{1, \dots, M\}}{\operatorname{argmin}} \log \det(\Sigma_m) + (x_* - \mu_m)^T \Sigma_m^{-1} (x_* - \mu_m) . \quad (2.41)$$

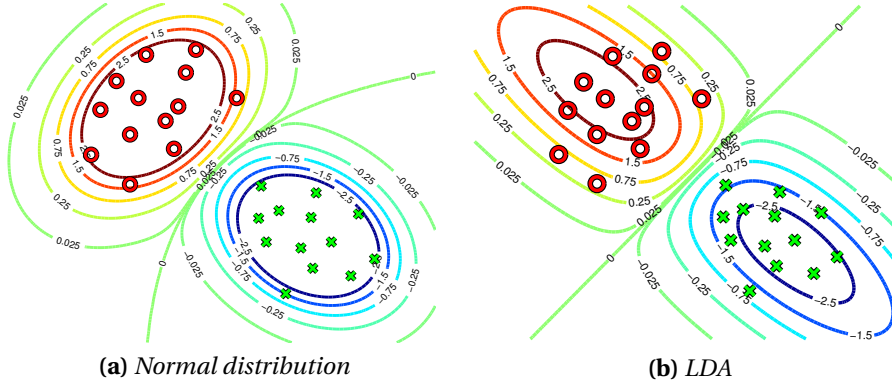


Figure 2.5: Normal distribution classifier (left) and LDA model (right) fitted to training examples of two classes (red and green crosses) following the maximum-likelihood principle. Additionally, several levels of constant decision scores computed via $p(x_* | \theta_1) - p(x_* | \theta_{-1})$ are overlayed. Note that assumptions of class-specific normal distributions lead to a quadratic decision boundary (see Eq. (2.41)), whereas assumptions of LDA force a linear separation (see Eq. (2.45)).

Note that we obtained Eq. (2.41) by replacing $p(x_* | \theta_m)$ with class-conditional densities introduced in Eq. (2.36). Furthermore, we exploited the monotonicity of the logarithm.

Properties of Normal Distribution Classifiers: As can be seen from the second part of Eq. (2.41), the resulting decision boundaries of Normal distribution classifiers are quadratic functions in x_* . Furthermore, the output of Eq. (2.41) is a class decision rather than a distribution of class probabilities. Due to both aspects, the decision rule is also called a quadratic discriminant. A visualization of applying a normal distribution classifier to a $D = 2$ toy example with two categories is given in Fig. 2.5a. Note that we can nicely see the resulting quadratic boundary between both classes.

An obvious generalization is to allow for multi-modal class distributions by assigning multiple Gaussians per class, which leads to so-called *Gaussian mixture models (GMMs)*. For an educational introduction, we refer the interested reader to (Kemmler, 2013, Chapter 3.2.1). A more detailed analysis can be found in (Bishop, 2006, Chapters 2.3.6, 4.2.2, and 9.2).

Linear Discriminant Analysis: Let us now drop the assumption of class-specific covariance matrices in Eq. (2.36). Instead, we assume that all classes share a single covariance matrix Σ_0 . In direct consequence, resulting class distributions are assumed to differ only in location, but coincide in rotation in space and relative

distribution of probability mass.

We observe that the resulting equation of Eq. (2.36) remains almost identical and only differs in replacing Σ_m by Σ_0 . However, the resulting decision boundaries are thereby significantly affected. To investigate this in more detail, let us consider the case of two classes for the simplicity of derivations, *i.e.*, $\mathcal{Y} = \{-1, 1\}$. Minimization over classes in Eq. (2.41) can then be written as:

$$\hat{h}_{\mathcal{N}}(\mathbf{x}_*) \propto \text{sgn}(p(\mathbf{x}_* | \boldsymbol{\theta}_1) - p(\mathbf{x}_* | \boldsymbol{\theta}_{-1})) \quad , \quad (2.42)$$

with $\text{sgn}(z)$ as the *sign operator*:

$$\text{sgn}(z) = \begin{cases} 1 & z > 0 \\ 0 & z = 0 \\ -1 & z < 0 \end{cases} \quad . \quad (2.43)$$

On a closer look, we notice that most of the terms nicely cancel out when inserting class-specific conditional probabilities of Eq. (2.36) with a shared covariance matrix. We further recall that Σ_0 is positive definite and symmetric, thus, $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T \Sigma_0^{-1} \mathbf{x}_* = \mathbf{x}_*^T \Sigma_0^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})$. Given both aspects, the resulting decision in Eq. (2.42) simplifies to:

$$\hat{h}_{\text{LDA}}(\mathbf{x}_*) \propto \text{sgn}\left(2(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T \Sigma_0^{-1} \mathbf{x}_* + c\right) \quad (2.44)$$

$$= \text{sgn}(\langle \mathbf{w}_{\text{LDA}}, \mathbf{x}_* \rangle + c) \quad . \quad (2.45)$$

Here, $\langle \cdot, \cdot \rangle$ is the inner product of two vectors. Furthermore, we introduced the vector $\mathbf{w}_{\text{LDA}} = \Sigma_0^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})$ and the scalar $c = -\frac{1}{2} \langle \boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1}, \mathbf{w}_{\text{LDA}} \rangle$. Since Eq. (2.45) is a linear separation of two classes, it gives rise to the name *linear discriminant analysis (LDA)*. A visualization is given in Fig. 2.5b.

Properties of LDA Classifiers: Especially in real-world applications, LDA models offer several advantages. If only final decisions matter, necessary computations are fast since only a linear projection is required. Additionally, memory requirements are only linear in dimensionality D and constant with respect to number of known examples N . This is a direct result of storing only $\mathbf{w}_{\text{LDA}}$ rather than $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_{-1}$, and Σ_0 .

An additional advantage comes for object detection scenarios, where classes are often highly imbalanced. Thus, we usually have only a restricted number of images showing a specific category (positives), but we can access millions of images from different classes showing what that single category is not like (negatives). For these cases, a single new example is unlikely to significantly affect Σ_0 . Furthermore, new negatives should not change $\boldsymbol{\mu}_{-1}$ which was already estimated from millions

of examples. In consequence, updating LDA models in detection scenarios boils down to only updating μ_1 and thus offers real-time capabilities.

It should be noted that projecting $\mathbf{x}_*$ on $\mathbf{w}_{\text{LDA}}$ can also be interpreted as dimensionality reduction to $D = 1$. During training, the projection is learned from data such that the resulting discrimination ability along that axis is as large as possible. Thereby, the intra-class variance of projected training examples is minimized, while the variance between different categories is maximized. Since the ratio of both is known as Fisher discriminant criterion (Bishop, 2006, p. 188), LDA is also referred to as *Fisher's linear discriminant*. See (Bishop, 2006, Chapter 4.1.4) for a detailed discussion on the correctness of the term *discriminant*.

We finally adhere that a third interpretation views LDA as nearest class mean technique in a decorrelated feature space, which is also referred to as whitening (Hariharan et al., 2012).

2.1.4.3 Support Vector Machines

Models for Linear Separations: In the previous section, we investigated LDA models which arose from the supposition of class-specific multivariate Gaussian distributions with shared covariance matrices. These linear models came with a set of desirable properties, *e.g.*, small memory consumption and fast computation of decision scores. However, the underlying might be not justifiable in every scenario. Thus, we might be interested in decisions obtained from *any* learnable linear separation $f \in \mathbb{F}_{\text{lin}}$:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b, \quad (2.46)$$

which are specified by normal vector $\mathbf{w}$ and offset b . In the following, we will denote these linear separations with $f_{\mathbf{w},b}$. As for LDA, the resulting discriminant function can be obtained by thresholding distance scores, *e.g.*, against zero via:

$$\hat{h}_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(f_{\mathbf{w},b}(\mathbf{x})) . \quad (2.47)$$

Several possibilities for estimating a linear model from data exist, *e.g.*, logistic regression which is relevant for neural networks (Murphy, 2012, Chapter 4.1.2, 16.5). Here, we will focus on a technique which became especially popular in our community and which is called support vector machine (SVM) for reasons which will become clear later. Although being supported from a large body of theoretical statements rooted in statistical learning theory (see Section 2.1.2), we will approach SVMs from a geometric perspective here. As in the previous section, we restrict labels to the binary case, *i.e.*, $\mathcal{Y} = \{-1, 1\}$. A short discussion on extensions to multi-class scenarios follows at the end of the section.

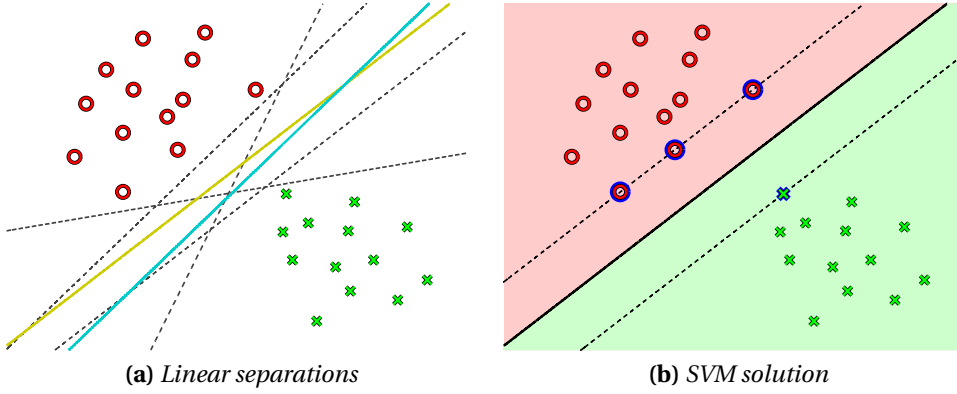


Figure 2.6: Left: a linear separable dataset $\mathcal{L}$ with two classes (same as in Fig. 2.5). As can be seen, there are several correct linear separations for $\mathcal{L}$. Of special interest is the solution of LDA in *blue* (see Section 2.1.4.2) and the SVM hyperplane in *yellow*. Right: The resulting hyperplane and decision regions obtained by solving the SVM optimization problem as stated in Eq. (2.61). We additionally notice the closest points of positive and negative data visualized in *dark blue* as well as the resulting boundaries of the margin plotted as dotted lines.

The Linear Separable Case: Let us start with collected data that is linear separable, *i.e.*, there exists a function $f_{w,b} \in \mathbb{F}_{lin}$ such that the training error is zero:

$$\mathcal{L} \text{ is lin. sep.} \Leftrightarrow \exists_{f_{w,b} \in \mathbb{F}_{lin}} : \forall_{(x_i, y_i) \in \mathcal{L}} : f_{w,b}(x_i) = \begin{cases} > 0 & y_i = 1 \\ < 0 & y_i = -1 \end{cases} . \quad (2.48)$$

Unfortunately, there is no unique solution of w for a linear separable dataset $\mathcal{L}$. See Fig. 2.6a for an intuitive visualization. Thus, the question arises which $f_{w,b}$ we should select and which consequences we can expect thereby.

From a geometric point of view, w and b describe a hyperplane with normal vector w and offset b . As visualized in Fig. 2.7a, any hyperplane divides the feature space in half and can thus be used for binary classification. Based on the geometric view, we can introduce the *margin* $\text{mg}_{\mathcal{L}^+}(f_{w,b})$ of a linear separation $f_{w,b}$ with respect to $\mathcal{L}$. In particular, we define $\text{mg}_{\mathcal{L}^+}(f_{w,b})$ as the difference between oriented distances of closest positive and negative training examples to the hyperplane $f_{w,b}$. For that, we recall that a hyperplane can also be defined as the set of points $\mathcal{S}_{w,b} = \{x | \langle w, x \rangle + b = 0\}$. The (perpendicular) distance between a point x_* and the hyperplane $\mathcal{S}_{w,b}$ is given as:

$$d_{\perp}(x_*, \mathcal{S}_{w,b}) = \frac{\langle w, x_* \rangle + b}{\|w\|} , \quad (2.49)$$

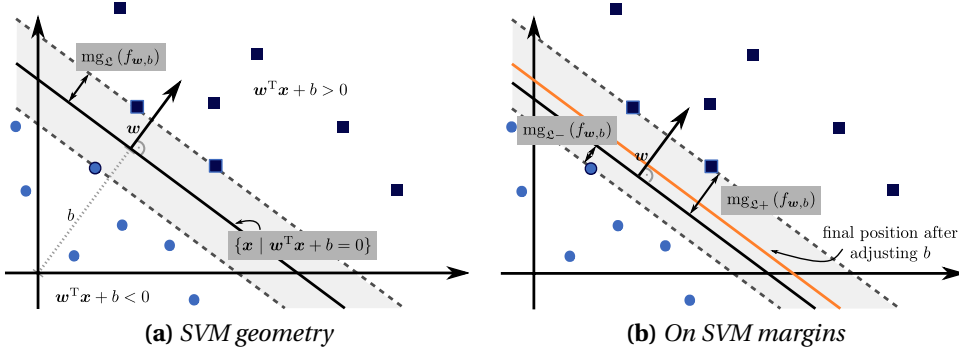


Figure 2.7: Left: Geometric interpretation of an SVM hyperplane for a linear separable dataset, which divides the entire space in half. Training examples are indicated as colored points. The examples with an additional halo are resulting support vectors which define the SVM margin. Right: The SVM margin originates from maximizing oriented distances to closest positive and negative examples.

where $\|v\|$ is the Euclidean norm of a vector $v \in \mathbb{R}^D$:

$$\|x\| = \left(\sum_{d=1}^D x(d)^2 \right)^{\frac{1}{2}}. \quad (2.50)$$

Let us now define the oriented margins of a linear separation $f_{w,b}$ with respect to a labeled dataset $\mathcal{L}$ with positive and negative examples as follows:

$$\text{mg}_{\mathcal{L}+}(f_{w,b}) = \min_{(x_i, y_i) \in \mathcal{L}, y_i=1} d_{\perp}(x_i, \mathcal{S}_{w,b}) \quad , \quad (2.51)$$

$$\text{mg}_{\mathcal{L}-}(f_{w,b}) = \max_{(x_i, y_i) \in \mathcal{L}, y_i=-1} d_{\perp}(x_i, \mathcal{S}_{w,b}) \quad . \quad (2.52)$$

The margin $\text{mg}_{\mathcal{L}}(f_{w,b})$ is then defined as the difference between these two oriented margins:

$$\text{mg}_{\mathcal{L}}(f_{w,b}) = \frac{\text{mg}_{\mathcal{L}+}(f_{w,b}) - \text{mg}_{\mathcal{L}-}(f_{w,b})}{2} \quad . \quad (2.53)$$

Note that if a hyperplane fulfills all constraints in Eq. (2.48), the quantity $\text{mg}_{\mathcal{L}+}$ will be positive whereas $\text{mg}_{\mathcal{L}-}$ will result in negative values. A visualization is given in Fig. 2.7b.

Intuitively, we would select a decision boundary which separates classes of training examples as clearly as possible. Under the previous point of view, this translates to searching the separation with largest margin with respect to $\mathcal{L}$. For our example in Fig. 2.6a, we would thus favor the lines plotted solidly over their dotted counterparts. Note that seeking for a large margin again reflects the assumption

of similar inputs leading to similar outputs, which would not be given for decision boundaries passing a training example unnecessarily close.

We additionally note that rescaling w and b with a scalar $\lambda > 0$ does not affect constraints as specified in Eq. (2.48). Thus, we can restrict our search for large margins *without loss of generality (w.l.o.g.)* to hyperplanes with parameter $\|w\| = 1$. Further note that the offset b cancels out when computing the margin defined in Eq. (2.53) and can for now be neglected. Under the condition that all constraints in Eq. (2.48) are satisfied, *i.e.*, if only valid separations are considered, it can then be shown that maximizing the margin results in a unique solution (Vapnik, 1998, Theorem 10.1):

$$w_* = \operatorname{argmax}_w \operatorname{mg}_{\mathcal{L}}(f_w, \cdot) . \quad (2.54)$$

While w_* only determines the relative orientation of the optimal hyperplane, we can additionally find the corresponding offset as intermediate of closest points:

$$b_* = - \frac{\min_{(x_i, y_i) \in \mathcal{L}, y_i=1} \langle w_*, x_i \rangle + \max_{(x_j, y_j) \in \mathcal{L}, y_j=-1} \langle w_*, x_j \rangle}{2} . \quad (2.55)$$

Thereby, we obtain the final position of the hyperplane in space. An insightful proof is given by Vapnik (1998, pp. 402).

Notes on Popular Definitions and Interpretations: Before turning on the benefits of margin maximization in terms of generalization abilities, let us shortly clarify some details which are used differently in today's literature. First of all, note that a slightly different definition of the margin is more popular today. When directly taking the resulting symmetry of positive and negative margin after adjusting b via Eq. (2.55) into account (see Fig. 2.7b), we can also write the optimization over parameters as:

$$w_*, b_* = \operatorname{argmax}_{w, b} \min_{(x_i, y_i) \in \mathcal{L}} y_i (\langle w, x_i \rangle + b) / \|w\| . \quad (2.56)$$

This quantity is called the *geometrical margin* and was exemplarily introduced by Schölkopf and Smola (2002, Chapter 7.2), Bishop (2006, Chapter 7.1), or Rodner (2011, Chapter 2.5), to name just a few. However, we found that following the original way of Vapnik (1998) is probably more intuitive.

Note further that hyperplanes are often scaled such that the following holds:

$$\min_{(x_i, y_i) \in \mathcal{L}} |\langle w, x_i \rangle + b| = 1 . \quad (2.57)$$

We refer to these scaled versions representations as *canonical hyperplanes*. With the concepts of geometrical margins and canonical hyperplanes at hand, we notice

that the geometrical margin of a canonical separating hyperplane equals $1/\|w_*\|$ and thus relates to a meaningful quantity.

Notably, the SVM solution described so far can additionally be interpreted in at least two other ways: first of all, the SVM-optimal hyperplane turns out to bisect the shortest connection between convex hulls of two classes (see (Schölkopf and Smola, 2002, Figure 7.5)). Besides that, if we assume training examples to be disturbed by measurement errors via additive noise with radius smaller than r , then the SVM solution will still correctly classify our examples if $\text{mg}_{\mathcal{L}}(f_{w,b}) > r$ (see (Schölkopf and Smola, 2002, Figure 7.3)). We will see in the following that the advantages of maximum margin classifiers are additionally supported by theoretical statements with respect to generalization abilities within the PAC framework.

Generalization Bounds for Linear Separable Data: Interestingly, our intuitive preference for large margins is consistent with results from learning theory in terms of expected generalization abilities as introduced in Section 2.1.2. One of these results was presented by Bartlett and Shawe-Taylor (1999).

Theorem 2.2 (A generalization bound for SVM and linear separable data without margin errors). *(based on (Bartlett and Shawe-Taylor, 1999, Theorem 1.7)) Let us assume that all features $x \in \Omega$ are located in a hypersphere of radius R . Let us further consider linear separations $f_{w,b}$ according to Eq. (2.46). For simplicity, let us assume $b = 0$. Finally, let us only consider linear separations with some upper bound on their norm $\|w\| \leq \frac{1}{\Upsilon}$, i.e., with a geometric margin of at least Υ (see Eq. (2.56)). Then there is a constant c such that the probability p_{ϵ} for errors on unseen test examples is not larger than*

$$p_{\epsilon} \leq \sqrt{\frac{c}{N} \left(\left(\frac{R}{\Upsilon} \right)^2 \log^2(N) + \log \left(\frac{1}{\delta} \right) \right)} \quad (2.58)$$

with probability at least $1 - \delta$ and independent from which data distribution the N training examples were drawn.

Let us shortly look into the assumptions of Theorem 2.2 before we inspect the bound in more detail. The requirement that features are located in a hypersphere is often met in computer vision applications. As an example, we have $R = 1$ when each dimension of x represents probability values, e.g., for object-specific attributes. Furthermore, excluding the offset is without loss of generality, since it can easily be included by augmented feature vectors with an additional dimension being constant to one. Thus, the assumptions are no strong restrictions.

Let us now investigate the result of Theorem 2.2. We notice that the expected error can be decreased in several ways. One way is to collect more training data (N increases). Similarly, the expected error decreases if samples occur only in compact

regions of space (R decreases). Besides that, maximizing the margin among all admissible hyperplanes also reduces the expected error (Υ increases).

The result stated above is especially appealing since there is no dependence on the actual dimensionality of the feature space anymore. Thus, maximum margin classifiers are expected to generalize well even for high-dimensional feature. This observation is especially useful for computer vision application where object representations are commonly high-dimensional. Besides, it gains further importance for kernelized variants as shown in Eq. (2.73) where inputs are embedded in even higher-dimensional spaces.

It should be noted that the previous theorem can be generalized to underline the trade-off between model capacity and training error as done by Bartlett and Shawe-Taylor (1999, Theorem 1.7). A slightly reformulated version of the result is given by Schölkopf and Smola (2002, Theorem 7.3)

Theorem 2.3 (A generalization bound for SVM for linear separable data with margin errors). *(based on (Bartlett and Shawe-Taylor, 1999, Theorem 1.7) and (Schölkopf and Smola, 2002, Theorem 7.3)) As previously, we assume a bounded hypersphere of radius R which contains all features $x \in \Omega$. We are interested in linear separations $f_{w,b}$ according to Eq. (2.46) with $b = 0$. As for the last statement, we impose an upper bound on their norm $\|w\| \leq \frac{1}{\Upsilon}$ which leads to a geometric margin of size $\geq \Upsilon$. We finally assume that not more than k training examples have an absolute oriented distance to the decision boundary which is smaller than $\rho \cdot \Upsilon$ (which causes margin errors). Then there is a constant c such that the probability p_ϵ for errors on unseen test examples is not larger than*

$$p_\epsilon \leq \sqrt{\frac{c}{N} \left(\left(\frac{R}{\rho \cdot \Upsilon} \right)^2 \log^2(N) + \log \left(\frac{1}{\delta} \right) \right)} + \frac{k}{N} \quad (2.59)$$

with probability of at least $1 - \delta$ and independent from which data distribution the N training examples have been sampled.

When inspecting the bound of Theorem 2.3, we again notice that collecting more data or enforcing small including hyperspheres reduces expected errors. The second term, called margin error, indicates that models with many violations of the margin during training are unlikely to result in small test errors. While violations can be decreased by reducing the desired margin (k/N decreases), this increases capacity and thus reduces expected generalization ($\rho \cdot \Upsilon$ decreases).

Finding the Optimal Hyperplane: Given the advantages of linear separations with large margins, it now remains to find such an optimal hyperplane for a given set of collected data. From the previous results, this translates to maximizing the

margin while keeping a training error of zero:

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{maximize}} && \text{mg}_{\mathcal{L}}(f_{\mathbf{w}, b}) \\ & \text{subject to} && \forall_{(\mathbf{x}_i, y_i) \in \mathcal{L}} \text{sgn}(f_{\mathbf{w}, b}(\mathbf{x}_i)) = y_i \end{aligned} \quad (2.60)$$

Unfortunately, it turns out that maximizing Eq. (2.60) is difficult. However, we can make use of the aforementioned canonical form of hyperplanes (see Eq. (2.57)). Thereby, the criterion for maximization simplifies to $1/\|\mathbf{w}\|$ and the constraints transform to $y_i \cdot f_{\mathbf{w}, b}(\mathbf{x}_i) > 1$, $(\mathbf{x}_i, y_i) \in \mathcal{L}$. As a final step, we notice that maximizing $1/\|\mathbf{w}\|$ leads to identical results as minimizing $1/2 \cdot \|\mathbf{w}\|^2$. Plugging all together, we arrive at the following quadratic optimization problem, which is also known as *primal optimization problem*:

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{minimize}} && \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{subject to} && \forall_{(\mathbf{x}_i, y_i) \in \mathcal{L}} y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \end{aligned} \quad (2.61)$$

Since Eq. (2.61) is *convex*, it is easy to solve with off-the-shelf optimization routines. As an example, the SVM solution to the $D = 2$ problem given in Fig. 2.6b was computed with Matlabs quadprog solver applied to Eq. (2.61). For an introduction to optimization of constrained convex problems, we refer the interested reader to (Schölkopf and Smola, 2002, Chapter 6.3).

When slightly rewriting the primal problem, we can see the close connection between SVMs and the principle of regularized risk minimization as introduced in Section 2.1.2.3. From that point of view, the optimization problem can also be stated as:

$$\underset{\mathbf{w}, b}{\text{minimize}} \nu \|\mathbf{w}\|^2 + \sum_{i=1}^N \mathcal{L}_{\infty}(f_{\mathbf{w}, b}(\mathbf{x}_i), y_i) \quad (2.62)$$

where ν is any positive value. The loss function $\mathcal{L}_{\infty}(\cdot, \cdot)$:

$$\mathcal{L}_{\infty}(f(\mathbf{x}), y) = \begin{cases} \infty & 1 - y_i \cdot f_{\mathbf{w}, b}(\mathbf{x}_i) > 0 \\ 0 & 1 - y_i \cdot f_{\mathbf{w}, b}(\mathbf{x}_i) \leq 0 \end{cases} \quad (2.63)$$

denotes the infinity-loss for binary-valued labels $y \in \{-1, 1\}$.

SVMs in Dual Formulation: Beside this connection, we can gain further knowledge by applying the technique of *Lagrange multipliers*. Thereby, we arrive at a dual representation of the optimization problem. From optimization theory, we

know that optimizing Eq. (2.61) can be done by incorporating all N constraints into a joint optimization criterion and by adding a corresponding Lagrange multiplier $\lambda_i > 0, i = 1, \dots, N$ for each constraint:

$$L(\mathbf{w}, b, \boldsymbol{\lambda}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \lambda_i (y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1) \quad (2.64)$$

$$= \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^N \lambda_i (1 - y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)) \quad (2.65)$$

The optimum of Eq. (2.65) can be found at the saddle point of the obtained Lagrangian $L(\mathbf{w}, b, \boldsymbol{\lambda})$. Thus, we need to maximize $L(\mathbf{w}, b, \boldsymbol{\lambda})$ with respect to *primal variables* $\mathbf{w}$ and b while minimizing it with respect to *dual variables* $\boldsymbol{\lambda}$.

To obtain the dual solution, we first set the derivatives of $L(\mathbf{w}, b, \boldsymbol{\lambda})$ with respect to its primal variables to zero. After that, we maximize the resulting function over the dual variables $\boldsymbol{\lambda}$ to obtain the final solution. The first step leads us to:

$$\mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i \quad (2.66)$$

with the condition on coefficients $0 = \sum_{i=1}^N \lambda_i$. Substituting both aspects into the Lagrangian in Eq. (2.65) and maximizing over $\boldsymbol{\lambda}$ finally gives us:

$$\begin{aligned} & \underset{\boldsymbol{\lambda}}{\text{maximize}} && \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j=1}^N \lambda_i \lambda_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ & \text{subject to} && \forall_{i \in \{1, \dots, N\}} \lambda_i \geq 0 \\ & \text{and} && \sum_{i=1}^N \lambda_i y_i = 0 \quad . \end{aligned} \quad (2.67)$$

From a computational point of view, solving the primal problem requires optimization of $\mathbf{w}$ and b of dimensionality $D + 1$. In contrast to that, the solution to the dual problem is of dimension N . We can thus conclude that the dual problem is usually faster to optimize than the primal version if the collected dataset is small. Similarly, the primal formulation should be preferred for large datasets.

Decisions in Dual Formulation and Support Vectors: Once the dual optimization problem is solved and we obtained optimal values for $\boldsymbol{\lambda}$, we can insert them into Eq. (2.66). Thereby, we can use the resulting normal vector to solve for an optimal offset as done previously in Eq. (2.55). The final discriminant function

comes in the form:

$$\hat{h}_{\text{SVM}}(\mathbf{x}_*) = \text{sgn} \left(\sum_{i=1}^N \lambda_i y_i \langle \mathbf{x}_i, \mathbf{x}_* \rangle + b \right) . \quad (2.68)$$

Let us inspect Eq. (2.68) in more detail. As can be seen, the resulting decision is composed of a weighted combination of inner products between test example $\mathbf{x}_*$ and each training example $\mathbf{x}_i$. We will come back to this observation in Section 2.1.5.5 when dealing with the representer theorem and Gaussian process models. For now, we note that only some weights λ_i will be larger than zero. These entries correspond to the constraints of the primal problem which are precisely met by the current solution (see Eq. (2.61)). Since these examples build the final decision, they are called *support vectors*, which finally gives rise to the name of the method.

It should be noted that support vectors offer an interesting property: their number can be by orders of magnitudes smaller compared with the number of training examples N . This property is often referred to as a *sparse solution*, since only few entries of λ will be non-zero. We already saw one example of a sparse solution in the $D = 2$ toy problem in Fig. 2.6b.

The Linear Non-separable Case (1): Soft Margin Classifiers: In the current formulation, SVM models offer several computational and theoretical benefits. However, their application is limited to linear separable data which is rarely the case in real-world applications. Two counter-examples are overlapping class distributions or measurement errors. Nonetheless, we might be interested in obtaining a linear model with similar properties which still separates data *as good as possible*.

From an optimization point of view, this means nothing but softening the constraints and adding a corresponding penalty if a constraint can not be met. Under a different perspective, it also translates to changing the loss function within the regularized risk as introduced in Eq. (2.62). In contrast to the infinity-loss used for the hard margin formulation, classification errors will now be penalized with a positive but finite value. The geometric interpretation is that we allow training examples to be located *on the wrong side* of the classification boundary.

Let us therefore introduce a *slack variable* $\xi_i \geq 0$ for each training example, which indicates how strongly the i^{th} training example violates the margin. We can then rewrite the primal optimization problem in Eq. (2.61) with soft constraints as:

$$\begin{aligned}
& \underset{\mathbf{w}, b, \boldsymbol{\xi}}{\text{minimize}} && \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \\
& \text{subject to} && \forall_{(\mathbf{x}_i, y_i) \in \mathcal{L}} y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \\
& \text{and} && \forall_{(\mathbf{x}_i, y_i) \in \mathcal{L}} \xi_i \geq 0 .
\end{aligned} \tag{2.69}$$

We introduced the trade-off parameter C which indicates our preference for the strength of acceptable violations of constraints. While only loose restrictions are to be expected for small values of C , large values lead to high penalization effects. As a look ahead, C is strongly connected to the noise term σ_n^2 for Gaussian process models as introduced in Section 2.1.5.3. From the constraints in Eq. (2.69), we can conclude that an example with $\xi_i = 0$ is on the correct side of the margin. In contrast, values in the interval $0 < \xi_i < 1$ indicate a violation of the margin while still being classified correctly. A value of $\xi_i \geq 1$ finally indicates a classification error.

Soft Margin Classifiers in Dual Formulation: As done for linear separable data, we can again write the optimization objective with all constraints into a joint Lagrangian. Again, we set derivatives with respect to involved primal variables to zero. Thereby, we obtain the following dual form (see (Bishop, 2006, Chapter 7.1.1) for a detailed derivation):

$$\begin{aligned}
& \underset{\boldsymbol{\lambda}}{\text{maximize}} && \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j=1}^N \lambda_i \lambda_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\
& \text{subject to} && \forall_{(\mathbf{x}_i, y_i) \in \mathcal{L}} C \geq \lambda_i \geq 0 \\
& \text{and} && \sum_{(\mathbf{x}_i, y_i) \in \mathcal{L}} \lambda_i y_i = 0 .
\end{aligned} \tag{2.70}$$

Note that the dual formulation is almost identical to its non-soft counterpart in Eq. (2.67) and only differs in the constraints on λ_i . For an intuitive understanding, we note that examples with $\lambda_i = 0$ do not contribute to the final decision. Similarly, $0 < \lambda_i < C$ indicates examples which are located on the margin, and examples with $\lambda_i = C$ violate hard constraints.

In comparison with the hard-margin version, we again see the correspondence to the regularized risk minimization framework (see Eq. (2.62)) when rewriting the primal problem in Eq. (2.69). The loss is now defined as the Hinge loss $\mathcal{L}_H(\cdot, \cdot)$:

$$\mathcal{L}_H(f(\mathbf{x}), y) = \max(0, 1 - y \cdot f(\mathbf{x})) . \tag{2.71}$$

Thereby, the resulting objective for regularized risk minimization translates to:

$$\underset{\mathbf{w}, b}{\text{minimize}} \nu \|\mathbf{w}\|^2 + \sum_{i=1}^N \mathcal{L}_H(f_{\mathbf{w}, b}(\mathbf{x}_i), y_i) \quad (2.72)$$

with $\nu = (2C)^{-1}$. As for the hard-margin version, we can expect sparse results when solving Eq. (2.69). However, we often observe that the majority of training data is still selected as support vectors in real-world applications. This phenomenon is mainly due to the curse of dimension in high-dimensional feature spaces (e.g., see (Léon Bottou, 2007, Chapter 1.4.1)).

The Linear Non-separable Case (2): Kernel-SVM: The soft margin classifiers introduced in the previous section allowed for linear separations with non-zero training errors. Here, we will follow an orthogonal way for approaching non-linear separability of data. The key observation is that we can separate more examples in high-dimensional spaces with linear functions than in low-dimensional spaces. This observation is closely related to the concept of VC-dimension as introduced in Section 2.1.2.2. For SVM models, we will exploit this observation by embedding data in a feature space $\mathcal{H}$ of significantly higher dimension than the original input space Ω . If the chosen feature space is large enough, we can thereby guarantee that a hard margin classifier can be applied successfully.

More formally, we aim for transformations $\Phi : \Omega \rightarrow \mathcal{H}$ and apply all previously reviewed techniques to transformed inputs $\Phi(\mathbf{x})$. For the SVM solution as given in Eq. (2.68), this directly translates to:

$$\hat{h}_{\text{SVM}}(\mathbf{x}_*) = \text{sgn} \left(\sum_{i=1}^N \lambda_i y_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_*) \rangle_{\mathcal{H}} + b \right) . \quad (2.73)$$

For a feature space of dimensionality $D_{\mathcal{H}} > N$, embedded data is guaranteed to be linear separable. Thus, we intuitively seek for embeddings into feature spaces which offer as many dimensions as possible. Note that this especially favors the extreme case of $D_{\mathcal{H}} = \infty$. However, practical considerations with respect to required memory and computation time easily limit the application of transformations to arbitrarily complex spaces.

Efficient Evaluations with Kernels: For the extreme case of embeddings in infinite dimensional spaces, the resulting computations can hardly be handled explicitly. Nonetheless, we can still apply these transformations provided that transformed examples appear only in inner products. As can be seen in Eq. (2.73), this holds for SVM models. In these cases, it is possible to replace inner products

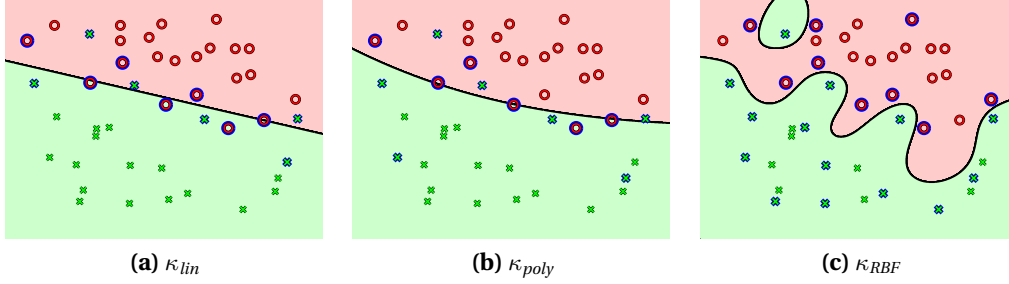


Figure 2.8: Examples for kernelized soft margin classifiers on a non-linear separable binary dataset. Left: the linear kernel κ_{lin} , middle: a polynomial kernel κ_{poly} with $p = 2$, and right: an RBF kernel κ_{RBF} with $\sigma = 2$. The SVM trade-off parameter C was set to 200. Solutions are obtained by solving the dual soft margin optimization problem as given in Eq. (2.70), where inner products have been replaced by kernel evaluations. Resulting support vectors are additionally highlighted in blue.

$\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$ in high-dimensional feature space $\mathcal{H}$ with a corresponding function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ that operates in the original input space Ω . Regarding terminology, we call these functions $\kappa(\cdot, \cdot)$ *kernels*.

It therefore remains to find pairs of kernel function κ and corresponding transformation Φ . Thereby, we obtain the flexibility of high-dimensional spaces while requiring resources which grow only asymptotic in the dimensionality of inputs. As it turns out, it is not even required to know this correspondence. Instead, it can be shown that any “valid” kernel corresponds to a dot product in a Hilbert space (Schölkopf and Smola, 2002, Chapter 2.2). A proper definition and further explanations are given in Section 2.1.5.4. For now, note that the simplest kernel function is the linear kernel:

$$\kappa_{\text{lin}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \quad (2.74)$$

where the associated transformation is the identity transform $\Phi_{\kappa_{\text{lin}}}(\mathbf{x}) = \mathbf{x}$. Besides, the following two examples are common kernel functions: the polynomial kernel with degree p :

$$\kappa_{\text{poly}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^p, \quad (2.75)$$

and the Gaussian *radial basis function* (RBF) kernel with bandwidth σ :

$$\kappa_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \quad \sigma \in \mathbb{R}_+. \quad (2.76)$$

For all three kernels, we visualized a learned SVM model with the resulting decision boundary in Fig. 2.8. As can be seen, the choice of a kernel significantly effects the

shape of possible decision functions, *e.g.*, for κ_{lin} we obtain a linear surface, whereas κ_{poly} allows for polynomials of degree p . Finally, note that a similar formulation to Eq. (2.73) can also be obtained for the primal problem:

$$\hat{h}_{\text{SVM}}(\mathbf{x}_*) = \text{sgn} \left(\sum_{i=1}^N \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_*) + b \right) , \quad (2.77)$$

with $\alpha_i \in \mathbb{R}$. This property is a direct consequence of the representer theorem which we will discuss in Section 2.1.5.5.

Asymptotic Requirements of Computation Resources: Solving the primal or dual formulation in Eq. (2.61) and Eq. (2.67) requires optimizing a convex *quadratic program* (QP) with linear (inequality) constraints. While solving a non-convex QP is NP complete as proven by Vavasis (1990), a convex QP can be solved in polynomial time (see also (Vavasis, 2009) for a detailed discussion on that topic). As noted by Murphy (2012), common solvers scale cubically in the number of inputs for the SVM optimization problem. Thus, solving the primal formulation can be done in $\mathcal{O}(D^3)$ operations, whereas the dual formulation scales with $\mathcal{O}(N^3)$. For a detailed introduction to solving convex optimization problems, we refer the interested reader to (Schölkopf and Smola, 2002, Chapter 6) and (Boyd and Vandenberghe, 2004).

A computation time which grows cubically in the number of training examples can be infeasible in practical applications. Therefore, specialized SVM solvers have been developed, *e.g.*, the *sequential minimal optimization* (SMO) algorithm by Platt (1998) which only scales with $\mathcal{O}(N^2)$ in practice (see also (Léon Bottou, 2007, Sect. 1) for an insightful discussion on that topic). With respect to the memory consumption during training, we observe that the primal formulation in Eq. (2.61) requires access to all examples. Thus, it demands $\mathcal{O}(ND)$ memory. In contrast, the dual formulation entirely relies on inner products for all pairs of known examples and thus scales with $\mathcal{O}(N^2)$. It should be noted that retaining all training examples with $\alpha_i \neq 0$ is still necessary for classifying unseen examples as can be seen in Eq. (2.68).

Hyperparameter Optimization for SVM Models: We have already seen several SVM hyperparameters which significantly effect the resulting decision areas. Two examples are the trade-off parameter C for soft margin classifiers and the bandwidth σ of an RBF kernel in Eq. (2.76). For the Bayesian perspective as introduced in Section 2.1.3, we obtained the elegant framework of likelihood maximization to compute optimal values for hyperparameters with respect to collected training. In contrast to that, techniques derived from the frequentist's perspective lack these

benefits. Hence, we need to rely on other techniques to estimate the quality for a set of parameters to then pick the one with smallest estimate.

The presumably most common technique in practice is cross-validation as mentioned in Section 2.1.2.3. During cross-validation, we estimate a set of possible values for every hyperparameter. This is done by training a model for each combination of hyperparameter values on a subset of $\mathcal{L}$. The quality of all learned models are validated on remaining examples, *e.g.*, by computing the classification accuracy. Finally, the parameter setting with highest quality score is greedily chosen to train the final model with. To increase statistical reliability of the resulting accuracy estimates, the procedure is repeated on multiple (random) splits of the dataset, giving rise to the *k-fold* cross-validation scheme. For an application in practice, it only remains to specify which parameter settings to evaluate. A simple solution is a *grid search* scheme, where a set of possible values is regularly placed in an admissible interval for each hyperparameter, *e.g.*, $C \in \{10^{-5}, \dots, 10^5\}$.

An advantage of the cross-validation technique is that it directly optimizes parameter values with respect to the desired measure of performance. In other words, high classification accuracies might be intuitively more valuable than high likelihood values. However, model training needs to be done for each split and for each possible value which needs to be estimated. Thus, cross-validation comes with an enormous overhead in computations. In addition, values optimized on a smaller subset of $\mathcal{L}$ are not guaranteed to also work well on the full set $\mathcal{L}$. Finally, this grid search scheme easily fails for increasing number of parameters due to the sheer explosion of space to search over. Besides these negatives aspects, cross-validation is often the method of choice for real-world SVM applications. For a deeper discussion on this topic and possible ways out, we refer the interested reader to (Schölkopf and Smola, 2002, Chapter 7.8). As probabilistic counterpart, we will see how to optimize hyperparameters within the framework of Gaussian process models by maximizing the data likelihood in Section 2.1.5.7.

Multi-class Classification with SVM Models: Until now, we mainly considered the scenario of two classes for the sake of clarity, *i.e.*, $\mathcal{Y} = \{-1, 1\}$. In lifelong learning scenarios, however, we will discover more than two concepts over time. Thus, we need an adequate extension to multiple classes with $\mathcal{Y} = \{1, \dots, M\}$. In general, we have two possibilities for dealing with multiple categories: either a direct extension of existing techniques to a *multi-class formulation*, or a reduction to several binary formulations which need to be fused later.

For a joint treatment of multiple categories, Vapnik (1998) proposed to learn M hyperplanes $h_{\mathbf{w}_m, b_m}$. This directly generalizes the soft margin optimization problem of Eq. (2.69) to multiple classes. For the ease of readability, we consider the formulation suggested by Weston and Watkins (1999) and refined by Schölkopf

and Smola (2002):

$$\begin{aligned}
 & \underset{\mathbf{w}_m, b_m, \xi_i^m}{\text{minimize}} && \frac{1}{2} \sum_{m=1}^M \|\mathbf{w}_m\|^2 + C \sum_{m=1}^M \sum_{\substack{i=1 \\ y_i \neq m}}^N \xi_i^m \\
 & \text{subject to} && \forall_{\substack{(\mathbf{x}_i, y_i) \in \mathcal{L} \\ m \in \{1, \dots, M\} \setminus y_i}} \langle \mathbf{w}_{y_i}, \mathbf{x}_i \rangle + b_{y_i} \geq \langle \mathbf{w}_m, \mathbf{x}_i \rangle + b_m + 2 - \xi_i^m \\
 & \text{and} && \forall_{(\mathbf{x}_i, y_i) \in \mathcal{L}} \xi_i^m \geq 0 .
 \end{aligned} \tag{2.78}$$

As we can see in the first set of constraints, parameters of class-specific hyperplanes are optimized such that for an example $\mathbf{x}$ and all M hyperplanes, the oriented distance to the hyperplane of class $y(\mathbf{x})$ is largest. Unfortunately, optimizing Eq. (2.78) is cumbersome in practice due to the huge number of involved parameters.

Let us therefore turn to the second solution, which can also be interpreted as a divide-and-conquer strategy. To transform a multi-class problem into several binary problems, we again have two possibilities: *one-vs-one* and *one-vs-all*. The former trains $\binom{M}{2}$ binary classifiers separating each possible pair of classes $(i, j) \in \mathcal{Y}^2$. Combination of binary decisions is then done based on counting which class was superior most often. This strategy is implemented within the well-known SVM solver LIBSVM by Chang and Lin (2011). More common, though, is the *one-vs-all* technique. Here, M binary classifiers are trained which separate each class from all remaining data. A final decision is then obtained by comparing the resulting scores. For the case of SVMs, this results in:

$$\hat{h}_{\text{SVM}}^{\text{1-vs-all}}(\mathbf{x}) = \underset{m \in \{1, \dots, M\}}{\text{argmax}} \langle \mathbf{w}_m, \mathbf{x} \rangle + b_m . \tag{2.79}$$

As stated by Vapnik (1998, p.438), “*this procedure usually gives good results*” and is therefore supported by LIBLINEAR (Fan et al., 2008), the second well-known SVM solver.

Comparing Outputs of Different SVM Models: Notably, it is not at all justifiable to directly compare scores of multiple class-specific models. This aspect gains even more importance if optimization of hyperparameters was carried out independently. To eliminate this drawback, it is common practice to transform (incomparable) class-scores into (comparable) probability estimates. This transformation can be done by squashing scores through a function $s : \mathbb{R} \rightarrow [-1, 1]$. One example for such *squashing functions* is the fit of a class-specific sigmoid, e.g., the logistic function:

$$s(z) = \frac{1}{1 + \exp(\gamma \cdot z + \beta)}, \quad \beta, \gamma \in \mathbb{R} . \tag{2.80}$$

This approach was originally presented by Platt (2000) and later modified by Lin et al. (2007) to increase numerical stability. Alternatively, we can apply a softmax function as shown in Section 5.6. For both techniques, estimation of fitting parameters β and γ can be done by cross-validation.

As a final note, we want to emphasize that all previously introduced techniques are designed according to the closed-world assumption as briefly introduced in Section 1.2. An example is the assignment of the class with highest score in Eq. (2.79), even if all scores are significantly negative. However, a more sophisticated system should reject the classification in these cases. Thereby, it indicates that the example likely stems from a new category and needs special treatment (*e.g.*, additional expert input). Obviously, this open-space assumption becomes especially important in lifelong learning scenarios, where the number of categories is not known in advance (see also Section 1.2).

2.1.5 Machine Learning with Gaussian Process (GP) Models

Towards Bayesian Perspectives: In the previous section, we dealt with two general ways for modeling relations between inputs and outputs. The first technique came without any model assumption and instead learned data by heart (the non-parametric KNN classifier in Section 2.1.4.1). As alternative approach, we introduced several models which followed parametric assumptions. Normal distribution classifiers and LDA models were rooted in a Bayesian perspective where optimal parameter values were obtained by applying the maximum-likelihood principle. Furthermore, we investigated maximum margin classifiers which stem from a frequentist perspective and are supported by strong theoretical results with respect to convergence properties. Nonetheless, techniques of both perspectives followed the plug-in principle as introduced in Eq. (2.17). Thus, parameters fully determined the models predictions once they were estimated from training data.

The only notable exception was the introduction of kernelized SVMs in Eq. (2.73). There, the final decision was not longer dependent on a selectable set of parameters, but instead required training examples even during prediction. In this section, we will follow that path of non-parametric models. We will even go one step further by dealing with a specific class of non-parametric models under a full Bayesian perspective. Although our primary interest is in models for classification, it is insightful to start with Gaussian process models for regression. We will see in Section 2.1.5.8 how to adapt these techniques to scenarios with discrete labels. Again, we use the notation f for referring to models with continuous outputs.

Marginalizing Over Functions: In contrast to the previous sections, we no longer assume that the relation between inputs and outputs is determined by a single set

of parameters. Instead, we marginalize over *all* possible relations f weighted by their prior probability $p(f)$:

$$p(y | \mathbf{x}) = \int_{\mathbb{H}} p(y | \mathbf{x}, f) p(f | \mathbf{x}) \mathrm{d}f \quad (2.81)$$

$$= \int_{\mathbb{H}} p(y | \mathbf{x}, f) p(f) \mathrm{d}f . \quad (2.82)$$

Note that we obtained Eq. (2.82) by *a priori* assuming independence between models and inputs. Note further the close connection to the marginalization over parameters as introduced in Eq. (2.28): while we marginalized over parameters for parameterized functions, this concept directly applies to marginalizing over non-parametric functions. As an intuitive argument, note that we can interpret non-parametric functions as parametric counterparts with a parameter set of infinite size.

Another similarity to the marginalization over parametric function arises when we inspect the role of prior terms. In Eq. (2.82), we can use the prior term $p(f)$ to express our assumptions regarding suitable function classes. Thereby, a Gaussian process assumption can lead to a preference of smooth functions as shown in Section 2.1.5.1. We will also see how observations will affect our prior assumptions and how preferred functions will pass through collected examples (Section 2.1.5.2) or close by (Section 2.1.5.3). A detailed understanding of involved concepts is essential since Gaussian process models will play an important part within the majority of subsequent chapters.

2.1.5.1 Gaussian Process Priors

Stochastic Processes and Gaussian Processes: As a first step, let us define the rather theoretical construct of a *stochastic process*, which will turn out to be highly useful later on.

Definition 2.4 (Stochastic process). (Schölkopf and Smola, 2002, p. 580) *A stochastic process F on a set Ω is a random quantity indexed by $x \in \Omega$. This means that for every x , we get a random quantity $F(x)$ taking values in $\mathbb{R}$, or more generally, in a set $\mathcal{R}$. A stochastic process is characterized by the joint probability distribution of F on arbitrary finite subsets of Ω ; in other words, of $(F(x_1), \dots, F(x_N))$.*

This definition sounds incomprehensible on first glance. However, we could as well say that a stochastic process F is an arbitrary number of random variables which share some common distribution. The joint distribution of variables allows us to infer information (see Section 2.1.5.2). If we observe realizations of a subset of the desired random variables, we can thus adapt our predictions for the remaining variables.

The choice of the joined distribution is what finally defines a specific stochastic process. Under the assumption of joint Gaussian distributions, we directly arrive at the concept of *Gaussian processes*.

Definition 2.5 (Gaussian process). (*Rasmussen and Williams, 2006, p. 13*) A *Gaussian process (GP)* is a collection of random variables $\mathbf{F} = (\mathbf{F}(\mathbf{x}))_{\mathbf{x} \in \Omega}$, any finite number of, which have a joint Gaussian distribution.

Since Gaussian distributions are fully specified by their first and second order moments, we can translate the previous definition of a Gaussian process into a bi-tuple of mean function $m : \Omega \rightarrow \mathbb{R}$ and covariance function $\kappa : \Omega \times \Omega \rightarrow \mathbb{R}_*$:

$$m(\mathbf{x}) = \mathbb{E}[\mathbf{F}(\mathbf{x})] \quad , \quad (2.83)$$

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}[(\mathbf{F}(\mathbf{x}_i) - m(\mathbf{x}_i))(\mathbf{F}(\mathbf{x}_j) - m(\mathbf{x}_j))] . \quad (2.84)$$

Note that by definition, the covariance function κ is symmetric and positive definite. According to Definition 2.5, any finite set $\{\mathbf{x}_i\}_{i=1}^N = \mathcal{X} \subset \Omega$ indexes N variables which are jointly distributed as:

$$(\mathbf{F}(\mathbf{x}))_{i=1}^N \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (2.85)$$

with $\boldsymbol{\mu} = (m(\mathbf{x}_1), \dots, m(\mathbf{x}_N))^T$ and $\boldsymbol{\Sigma}(i, j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Under the viewpoint of machine learning, we can now interpret random variables as latent function values, and samples drawn from input space $\{\mathbf{x}_i\}_{i=1}^N = \mathcal{X} \subset \Omega$ serve as index set. In consequence, we can denote the assumption of f being distributed according to a GP prior with:

$$f \sim \mathcal{GP}(m(\cdot), \kappa(\cdot, \cdot)) \quad . \quad (2.86)$$

Note that this directly replaces the generic prior in Eq. (2.82) when marginalizing over latent functions.

Details on GP Priors: Let us now further familiarize ourselves with the assumptions of GP models. To this end, note that Definition 2.5 of a joint Gaussian distribution especially applies to the extreme case of a single indexing sample. Thus, in every single point $\mathbf{x}$, the latent function value is distributed as a 1D Gaussian $\mathcal{N}(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}))$. Furthermore, we observe that function values of several points are correlated, determined by the covariance function $\kappa(\cdot, \cdot)$. In absence of training examples, these two observations fully summarize our prior assumptions.

For a specific choice of $m(\cdot)$ and $\kappa(\cdot, \cdot)$, we can now draw *function samples* from the GP prior. A common example for the mean function is the *zero-mean assumption* $m(\cdot) \equiv 0$. Regarding the covariance function, an RBF kernel κ_{RBF} as introduced in Eq. (2.76) is a frequently chosen option. We visualized function samples drawn from $\mathcal{GP}(0, \kappa_{\text{RBF}}(\cdot, \cdot))$ in Fig. 2.9. For clarity, note that we did *not*

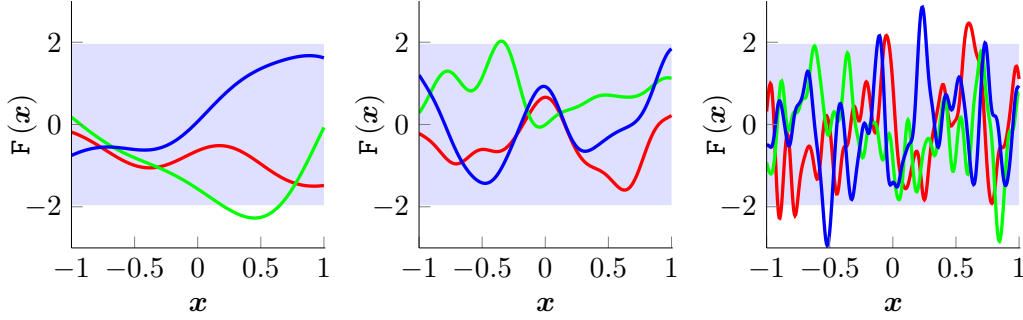


Figure 2.9: Functions sampled from a Gaussian process prior with zero mean and κ_{RBF} as covariance function. Shaded regions indicate twice the standard deviation. The bandwidth parameter σ was set to 0.5, 0.2, and 0.04 from left to right, respectively.

sample “complete” functions in Fig. 2.9. Instead, we sampled function values for finite index sets according to mean and covariance function. Note additionally that the distribution of function values at every index point is a 1D Gaussian. For visualization, shaded areas indicate twice the standard deviation. Furthermore, we observe that the mean equals zero due to the zero mean assumption (compare with Eq. (2.82)). In the next section, we will see how the distributions over function values will be affected once we observe training examples.

2.1.5.2 GP Regression (1): Noise-free Observations

Once we observe training data, we are interested in adapting our predictions from prior assumptions to posterior probabilities. Similar to the way in which we approached SVMs in Section 2.1.4.3, let us first seek for models which perfectly explain our training data. Later, we will relax this assumption and derive a probabilistic counterpart to the soft margin SVM extension.

Transferring Information Between Observed and Unseen Examples: Let us start at the definition of a GP as given in Definition 2.5. As we can see, any finite set of inputs indexes variables which are jointly distributed (see Eq. (2.85)). For notational convenience, let us now split the index set in two terms, $\mathcal{X}$ and $\mathcal{X}_*$. Note that the two sets do not need to be disjoint. Furthermore, let $\mathbf{F}(\mathcal{X}) = (\mathbf{F}(x_i))_{x_i \in \mathcal{X}}$ denote the random variables indexed by $\mathcal{X}$. Let us additionally introduce two abbreviations which will significantly shorten the remaining formulas. Previously, we defined mean function and covariance function with one and two index elements, respectively. Let us now generalize this notation to inputs which are index sets. To this

end, let us denote the mean function which operates on an index set $\mathcal{X}$ as:

$$m(\mathcal{X}) \doteq (m(\mathbf{x}_1), \dots, m(\mathbf{x}_N))^T, \quad \mathbf{x}_1, \dots, \mathbf{x}_N \in \mathcal{X}, \quad N = |\mathcal{X}|. \quad (2.87)$$

Thus, we obtain a $|\mathcal{X}|$ -dimensional vector $m(\mathcal{X}) \in \mathbb{R}^{|\mathcal{X}| \times 1}$. Furthermore, we will denote the output of a covariance function which operates on two index sets $\mathcal{X}_1$ and $\mathcal{X}_2$ as:

$$\kappa(\mathcal{X}_1, \mathcal{X}_2) \doteq \begin{bmatrix} \kappa(\mathbf{x}_{i_1}, \mathbf{x}_{j_1}) & \dots & \kappa(\mathbf{x}_{i_1}, \mathbf{x}_{j_{N_2}}) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_{i_{N_1}}, \mathbf{x}_{j_1}) & \dots & \kappa(\mathbf{x}_{i_{N_1}}, \mathbf{x}_{j_{N_2}}) \end{bmatrix}. \quad (2.88)$$

$\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_{N_1}} \in \mathcal{X}_1, \mathbf{x}_{j_1}, \dots, \mathbf{x}_{j_{N_2}} \in \mathcal{X}_2, N_1 = |\mathcal{X}_1|, N_2 = |\mathcal{X}_2|$

Thus, we have $\kappa(\mathcal{X}_1, \mathcal{X}_2) \in \mathbb{R}^{|\mathcal{X}_1| \times |\mathcal{X}_2|}$. We can now rewrite the joint distribution of indexing variables in Eq. (2.85) as

$$\begin{bmatrix} \mathbf{F}(\mathcal{X}) \\ \mathbf{F}(\mathcal{X}_*) \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}_* \\ \boldsymbol{\Sigma}_*^T & \boldsymbol{\Sigma}_{**} \end{bmatrix} \right). \quad (2.89)$$

For the ease of readability, we introduced $\boldsymbol{\mu}_* = m(\mathcal{X}_*)$, $\boldsymbol{\Sigma}_* = \kappa(\mathcal{X}, \mathcal{X}_*)$, and $\boldsymbol{\Sigma}_{**} = \kappa(\mathcal{X}_*, \mathcal{X}_*)$, using the notation defined in Eq. (2.87) and Eq. (2.88). Note that the symmetry $\boldsymbol{\Sigma}_* = \boldsymbol{\Sigma}_*^T$ follows from the definition of $\kappa(\cdot, \cdot)$ as covariance function (see Eq. (2.84)).

If we now observe the correct function values $\mathbf{y}$ corresponding to the first index set during training, we can obtain conditional posterior probabilities as follows (e.g., see (Murphy, 2012, Chapter 4.3)):

$$\mathbf{F}(\mathcal{X}_*) \mid \mathcal{X}, \mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_* + \boldsymbol{\Sigma}_*^T \boldsymbol{\Sigma}^{-1}(\mathbf{y} - \boldsymbol{\mu}), \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_*^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}_*). \quad (2.90)$$

An application to the aforementioned 1D toy example is shown in Fig. 2.10.

Details on GP Posteriors: When inspecting Eq. (2.90), we can immediately make several interesting observations. First of all, it can be seen that the mean estimate for new data is influenced by observed training examples in two ways. The term $\boldsymbol{\Sigma}_*^T$ gives larger influence to training examples which strongly correlate with new data. This reflects the assumption that information should only be transferred between similar examples. Furthermore, the term $(\mathbf{y} - \boldsymbol{\mu})$ gives higher weights to examples which are not already explained by the prior and carry additional information.

Besides, we can inspect the effect of an example which occurs in training set $\mathcal{X}$ and test set $\mathcal{X}_*$. Since the corresponding product in $\boldsymbol{\Sigma}_*^T \boldsymbol{\Sigma}^{-1}$ results in a value of 1, the observed target value y is retained.

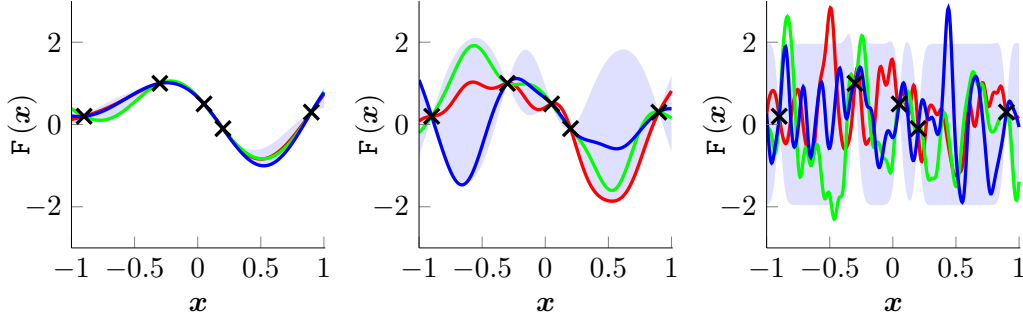


Figure 2.10: Functions sampled from a Gaussian process with zero mean and κ_{RBF} as covariance function after observing training examples (shown as black crosses). Shaded regions indicate twice the standard deviation. The bandwidth parameter σ was set to 0.5, 0.2, and 0.04 from left to right, respectively.

With respect to the posterior variance, we notice that it equals zero for test examples which are identical to training inputs. In general, the variance is decreased by a term that expresses how much information of unknown examples $\mathcal{X}_*$ is assumed to be unveiled by training data $\mathcal{X}$. Interestingly, this holds independent of observed target *values*. Thus, even for unexpected observations with large differences between $\mathbf{y}$ and $\boldsymbol{\mu}$, the conditional posterior variance will not increase. In less formal words: any observable data carries some valuable information, no matter how small.

2.1.5.3 GP Regression (2): Noisy Observations

In many real-world applications, observing error-free labels is challenging, expensive, or even impossible, *e.g.*, due to measurement errors or quantization effects. From a different perspective, we are sometimes not even interested in models that perfectly explain collected data, but would favor some regularization instead. We already tackled these scenarios within the regularized risk framework as introduced in Section 2.1.2.3 or the soft margin version of SVMs in Section 2.1.4.3. Let us now turn to the probabilistic counterpart thereof.

Noise Models for Continuous Targets: In realistic scenarios, the exact function values can often not be observed directly. Instead, we commonly measure signals after undergoing an additional process of corruption ε , which is considered as *noise*:

$$y(\mathbf{x}) = \varepsilon(f(\mathbf{x}), \mathbf{x}) \quad . \quad (2.91)$$

With respect to the marginalization over latent functions in Eq. (2.82), the incorporation of this additional process translates to:

$$p(y|\mathbf{x}) = \int_{\mathbb{H}} \int_{\mathbb{R}} p(y|\mathbf{x}, f, F) p(F|\mathbf{x}, f) p(f) dF df . \quad (2.92)$$

Here, F represents the latent function value before undergoing the noise process and y is the output which is finally observable. The presumable most common noise process is that of additive noise:

$$y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon(\mathbf{x}) . \quad (2.93)$$

If observations are further assumed to be identically and independently disturbed, which is known as *homogeneous noise*, we arrive at:

$$y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon . \quad (2.94)$$

The quantity ε can now be seen as a random variable distributed according to a noise distribution model which is independent of $\mathbf{x}$. Under this assumption, the marginalization in Eq. (2.92) simplifies to:

$$p(y|\mathbf{x}) = \int_{\mathbb{H}} \int_{\mathbb{R}} p(y|F) p(F|\mathbf{x}, f) p(f) dF df . \quad (2.95)$$

It is not unexpected that a common choice for the distribution of $p(y|F)$ is a 1D Gaussian with zero mean and variance σ_n^2 , which is also known as *white Gaussian noise*:

$$y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) . \quad (2.96)$$

Let us now incorporate this noise process into our previous results for Gaussian process regression.

Predictions with Noisy Inputs: We recall that the sum of two independent random variables which are normally distributed is again a normal distribution with summed means and covariances. Transferred to the previously introduced assumptions regarding distributions of latent function f and noise process ε leads to:

$$\begin{bmatrix} y(\mathcal{X}) \\ y(\mathcal{X}_*) \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma} + \sigma_n^2 \mathbf{I} & \boldsymbol{\Sigma}_* \\ \boldsymbol{\Sigma}_*^T & \boldsymbol{\Sigma}_{**} + \sigma_n^2 \mathbf{I} \end{bmatrix} \right) . \quad (2.97)$$

Identity matrices $\mathbf{I}$ stem from the assumption of independently distributed noise and are chosen with appropriate size. If we again compute the posterior by condi-

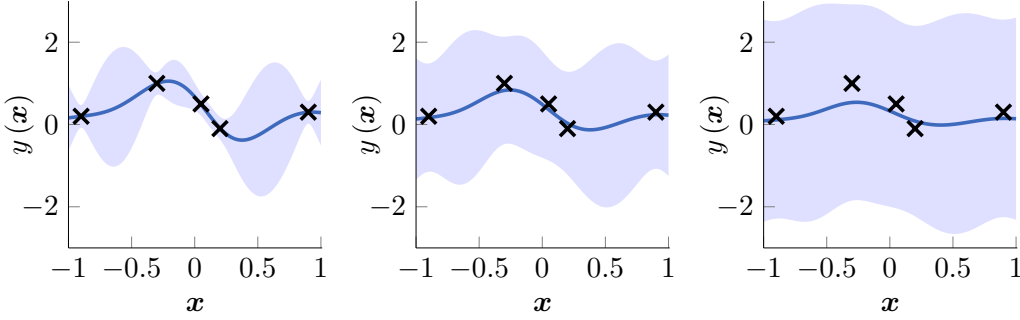


Figure 2.11: Predictive distributions of a Gaussian process with zero mean and κ_{RBF} as covariance function after observing noisy training examples (shown as black crosses). Shaded regions indicate twice the standard deviation, whereas the predictive mean is plotted as solid line. The bandwidth parameter σ was set to 0.25, and the expected noise variance σ_n^2 was assumed as 0.1, 0.5 and, 1.0 from left to right, respectively.

tioning on the observed labels during training, we finally arrive at:

$$\mathbf{y}(\mathcal{X}_*) \mid \mathcal{X}, \mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_* + \boldsymbol{\Sigma}_*^T (\boldsymbol{\Sigma} + \sigma_n^2 \mathbf{I})^{-1} (\mathbf{y} - \boldsymbol{\mu}), \right. \\ \left. \boldsymbol{\Sigma}_{**} + \sigma_n^2 \mathbf{I} - \boldsymbol{\Sigma}_*^T (\boldsymbol{\Sigma} + \sigma_n^2 \mathbf{I})^{-1} \boldsymbol{\Sigma}_*\right). \quad (2.98)$$

A visualization using the previous 1D example is given in Fig. 2.11. As we can nicely observe, the influence of training examples on the resulting prediction decreases with an increasing noise level. This especially holds for re-predictions of already observed inputs, which are no longer perfectly explained. More specifically, neither the predictive mean nor any sampled function is any more guaranteed to exactly pass observations, which is reflected in the non-zero variance for training inputs. In general, the uncertainty of estimates for arbitrary inputs increases, which again shows the additional unreliability of observations induced by the noise process.

In the remainder of this thesis, we will always follow the zero mean assumption, since we usually have no additional domain knowledge telling us differently. Consequently, the vectors of prior mean estimates in Eq. (2.98) will vanish, which simplifies the predictive mean estimate. Note that this also releases the symbol $\boldsymbol{\mu}_*$ for different use. In the following sections and chapters, we will often encounter the prediction tailored to a single test example, *i.e.*, $\mathcal{X}_* = \{\mathbf{x}_*\}$. For notational convenience, we will write the first and second moments of the predictive distribution

as:

$$\mu_* \doteq \mu(\mathbf{x}_* | \mathbf{X}, \mathbf{y}) = \Sigma_*^T (\Sigma + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \quad (2.99)$$

$$\sigma_*^2 \doteq \sigma^2(\mathbf{x}_* | \mathbf{X}, \mathbf{y}) = \Sigma_{**} + \sigma_n^2 - \Sigma_*^T (\Sigma + \sigma_n^2 \mathbf{I})^{-1} \Sigma_* \quad (2.100)$$

Note that Σ_* is now of dimension $N \times 1$. Similarly, variables Σ_{**} and $\sigma_n^2 \mathbf{I}$ reduced to scalar variables.

2.1.5.4 Details on Kernel Functions

Before we move on to techniques for optimizing involved hyperparameters, let us shortly look into the conceptual foundations of kernels and the resulting properties with respect to GP models.

Covariance Functions and Kernel Functions: First of all, we notice that the computation of predictive distributions for GP regression includes inverting the covariance matrix Σ (see Eq. (2.90)). In consequence, the covariance matrix Σ needs to be *positive definite*, i.e.,

$$\forall_{\substack{\mathbf{v} \in \mathbb{R}^N \\ \mathbf{v} \neq \mathbf{0}}} : \mathbf{v}^T \Sigma \mathbf{v} > 0 \quad (2.101)$$

This further highlights the necessity of κ being positive definite as required in the definition of a GP in Eq. (2.84). We can thus conclude that for any collection of inputs $\mathcal{X} \in \Omega^N$, the resulting matrix $\Sigma = \kappa(\mathcal{X}, \mathcal{X})$ is positive semi-definite. Furthermore, the equality to zero can only occur for $\mathbf{v} = \mathbf{0}$ or if training inputs are linear dependent, e.g., $\mathbf{x}_i = \mathbf{x}_j$. In addition, we note that $\Sigma = \Sigma^T$ since covariance functions are symmetric by definition. Interestingly, this opens the door to a much broader field of functions being admissible as covariance functions: the field of *positive definite kernels* which we already tackled with Kernel-SVMs in Section 2.1.4.3. In the following, we will present relevant details of this type of functions and start with a sound definition of the concept of a kernel.

Definition 2.6 ((Positive definite) kernel). (based on (Rasmussen and Williams, 2006, p. 80)) A kernel denotes any function $\kappa(\cdot, \cdot)$ which maps a pair of inputs to a real-valued scalar:

$$\kappa : \Omega \times \Omega \rightarrow \mathbb{R} \quad (2.102)$$

A kernel is said to be positive semi-definite (psd) if, for any finite collection $\mathcal{X}$ of N inputs, it gives rise to a kernel matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$, $\mathbf{K}(i, j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ which is positive semi-definite:

$$\forall_{N \in \mathbb{N}_+} \forall_{\mathcal{X} \in \Omega^N} : \mathbf{v}^T \mathbf{K} \mathbf{v} = \sum_{i,j} \kappa(\mathbf{x}_i, \mathbf{x}_j) v_i v_j \geq 0 \quad (2.103)$$

for any vector $v \in \mathbb{R}^N$.

Due to the resulting interchangeability of covariance functions and (psd) kernels with respect to GP models, we also used the same symbol κ for both concepts so far. From now on, we will further highlight this connection by writing solely $\mathbf{K}$ instead of Σ for involved covariance matrices. Similarly, we will replace the kernel vector Σ_* and self-similarity Σ_{**} computed for a single test input in Eq. (2.99) and Eq. (2.100) with $\mathbf{k}_*$ and $\mathbf{k}_{**}$, respectively.

Properties of Kernels: An important property of kernel functions is their closure under several operations, e.g., scaling, addition, multiplication, or feature combination (see (Bishop, 2006, p.296) for a detailed overview). Thus, kernels can offer various forms which can be exploited for efficient evaluations. As an example, a kernel is called *additive* if its computation can be decomposed into 1D kernel functions κ_d operating on each input dimension independently:

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^D \kappa_d(\mathbf{x}(d), \mathbf{x}'(d)) \quad . \quad (2.104)$$

If these functions are further linear with respect to multiplicative scaling:

$$\kappa(c\mathbf{x}, c\mathbf{x}') = \sum_{d=1}^D c\kappa_d(\mathbf{x}(d), \mathbf{x}'(d)) \quad , \quad c \in \mathbb{R}_+ \quad , \quad (2.105)$$

the resulting kernel is called *homogeneous*⁴. A kernel is further said to be *stationary* if it depends only on the difference between inputs rather than the inputs directly (Genton, 2002):

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \kappa_{\text{stat}}(\mathbf{x}_i - \mathbf{x}_j) \quad . \quad (2.106)$$

In addition, we call a kernel *isotropic* if it only depends on the distance between both inputs. Distances are either measured by a distance function $d(\cdot, \cdot)$ as introduced in Section 2.1.4.1, or expressed in terms of the corresponding norm $\|\cdot\|_d$:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \kappa_{\text{iso}}(d(\mathbf{x}_i, \mathbf{x}_j)) = \kappa_{\text{iso}}(\|\mathbf{x}_i - \mathbf{x}_j\|_d) \quad . \quad (2.107)$$

Interestingly, it was shown by Bochner (1955) that the Fourier transform of a stationary kernel function needs to fulfill several criteria. As an example, it is required to be real-valued and non-negative in order to be permissible, i.e., for being psd (see

⁴The term homogeneous is sometimes also used as synonym for *isotropic* as defined in Eq. (2.107), e.g., by Genton (2002) or Bishop (2006). We instead follow Vedaldi and Zisserman (2012) and understand homogeneous as introduced in Eq. (2.105).

(Genton, 2002) for details). As a result, not every norm leads to a valid psd kernel in Eq. (2.107). In fact, many isotropic kernels require $\|\cdot\|_d$ to be the Euclidean norm, as shown by Christakos and Papanicolaou (2000).

Well-known Kernel Functions: In Eq. (2.76), we already introduced a kernel function which is isotropic and as such also stationary: the RBF kernel κ_{RBF} . Note that κ_{RBF} is also referred to as squared exponential kernel (Rasmussen and Williams, 2006, p.83). While we introduced κ_{RBF} with Euclidean norm, other p -norms with $0 < p \leq 2$ still lead to a valid psd kernel function. This especially includes the L_1 -norm (see (Rasmussen and Williams, 2006, p.86) or directly (Christakos and Papanicolaou, 2000) for a proof).

In addition to the RBF kernel κ_{RBF} , we already introduced two other kernel functions when dealing with Kernel-SVMs: the linear kernel κ_{lin} and the polynomial kernel κ_{poly} (Eqs. (2.74) to (2.75)). However, we are interested in lifelong learning for *visual* recognition. Thus, we can and should incorporate properties of common image representations into the computation of similarity (see Section 2.2 for an overview of image representations). As an example, non-negativity of feature components ($x(d) > 0$, $d = 1, \dots, D$) and L_1 normalization ($\sum_{d=1}^D x(d) = 1$) could be taken into account, which are properties of commonly used histogram features.

A well-known kernel function which is psd for these features is the chi-square kernel:

$$\kappa_{\chi^2}(x, x') = 2 \sum_{d=1}^D \frac{(x(d) x'(d))^2}{x(d) + x'(d)} . \quad (2.108)$$

As the name suggests, the chi-square kernel is related to the chi-square distance $d_{\chi^2}(x, x') = \sum_{d=1}^D \frac{(x(d) - x'(d))^2}{x(d) + x'(d)}$ with its roots in statistics. A proof for the positive definiteness of κ_{χ^2} was given by Fowlkes et al. (2004).

Another frequently used kernel function for histogram features is the *histogram intersection kernel (HIK)*:

$$\kappa_{\text{HIK}}(x, x') = \sum_{d=1}^D \min(x(d), x'(d)) \quad (2.109)$$

which is also known as minimum kernel. The positive definiteness of κ_{HIK} was shown by Barla et al. (2002). Interestingly, a GP model with zero mean assumption and κ_{HIK} as covariance function becomes a *Wiener process* and as such describes the *Brownian motion* of particles. Furthermore, the histogram intersection kernel can be nicely generalized as shown in Eq. (3.1), Section 3.2.1.

Note that neither κ_{χ^2} nor κ_{HIK} are stationary kernels. However, both are additive and homogeneous. We will see in Chapter 3 how to exploit this property for large-scale applications with several thousands of training examples.

Mercer's Theorem and Kernel Trick: So far, we have seen several types of kernel functions to evaluate the (dis)similarity of pairs of inputs. To fully understand which benefits we can expect from using kernels, though, the following statements are valuable.

Theorem 2.7 (Mercer's theorem). *(Mercer, 1909, Part 4) For any psd kernel function $\kappa(\cdot, \cdot)$, there exists a feature map $\Phi_\kappa : \Omega \rightarrow \mathcal{H}_\kappa$ from input space Ω to a Hilbert space $\mathcal{H}$ such that $\kappa(x, x') = \langle \Phi_\kappa(x), \Phi_\kappa(x') \rangle_{\mathcal{H}_\kappa}$, $x, x' \in \Omega$.*

Note that the original formulation of Mercer's theorem in Mercer (1909) is actually more general. However, we believe that the formulation above is more insightful under the scope of this thesis. Furthermore, note that the converse of the theorem also holds (see (Schölkopf and Smola, 2002, Problem 2.23)). The interested reader is referred to (Mercer, 1909) for details.

Corollary 2.8 (Kernel trick). *Any learning algorithm based on a psd kernel $\kappa(\cdot, \cdot)$ can be cast into an alternative, equivalent formulation by replacing κ with another psd kernel $\kappa'(\cdot, \cdot)$.*

A justification for Corollary 2.8 is given in (Schölkopf and Smola, 2002, p. 34).

Note that Corollary 2.8 also covers a special case of the kernel trick which we used for kernelizing SVM classifiers in Eq. (2.73). To see this, let $\kappa(\cdot, \cdot)$ be the original inner product in input space Ω , i.e., $\kappa(x, x') = \langle x, x' \rangle_\Omega$ and $\Phi_\kappa(x) = x$. Based on the corollary, we can now safely replace the original inner product with any psd kernel $\kappa'(x, x') = \langle \Phi_{\kappa'}(x), \Phi_{\kappa'}(x') \rangle_{\mathcal{H}}$. As direct consequence, we can circumvent explicit computations of inner products in space $\mathcal{H}$ which might be even of infinite dimensionality as for κ_{RBF} . Thus, we retain computationally tractable models.

Due to the combination of Mercer's theorem and the kernel trick, we can implicitly transform features from dimensionality D to D' , while keeping computational complexity of evaluations in $\mathcal{O}(D)$.

2.1.5.5 The Representer Theorem

In Section 2.1.3.3, we already noticed a connection of frequentist perspective and Bayesian perspective for the choice of a specific loss function. In the following, we will derive a second connection.

Reproducing Kernel Hilbert Spaces: In the previous sections, we derived the predictive mean of GP regression for an *a priori* zero-mean assumption (see Eq. (2.90) and Eq. (2.98)). For noise-free and noisy scenarios, it is insightful to note that we can express the predictive mean as follows:

$$\mu_* = \mathbf{k}_*^T \boldsymbol{\alpha} \quad , \quad (2.110)$$

with $\alpha = K^{-1}y$ and $\alpha = (K + \sigma_n^2 I)^{-1} y$. Here, we immediately observe a striking similarity to the solution of Kernel-SVMs in dual formulation as shown in Eq. (2.73). The reason for this similarity is that the *representer theorem* – which plays a central role in machine learning research – applies for both methods.

For approaching the representer theorem, we note that each kernel function induces a unique *reproducing kernel Hilbert space (RKHS)* $\mathcal{H}$. An RKHS can be roughly understood as a Hilbert space of real-valued functions f over an index set Ω where the existence of an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ allows for computing the norm of functions $\|f\|_{\mathcal{H}}$. While this definition is relatively loose, we refer to (Rasmussen and Williams, 2006, Chapter 6.1), (Schölkopf and Smola, 2002, Chapter 2.2.3), and (Vapnik, 1998, Chapter 11.12.1) for details.

Minimizing the Regularized Risk Allows for Useful Representations of Decision Functions: In Section 2.1.2.3, we introduced the regularized risk $\mathcal{R}_{\text{reg}}(h, \mathcal{L})$ of a hypothesis h evaluated on data $\mathcal{L}$. This quantity consists of the empirical risk $\mathcal{R}_{\text{emp}}(h, \mathcal{L})$ and an additional term $\nu \mathcal{C}(h)$ which penalizes complex models. For the latter one, we can benefit from the RKHS with its possibility for computing the norm $\|f\|_{\mathcal{H}}$ of functions.

To this end, let us define a monotonically increasing scoring function $\omega : [0, \infty) \rightarrow \mathbb{R}$, e.g., the squared norm $\omega(z) = z^2$. Thus, $\omega(\|f\|_{\mathcal{H}})$ can serve as complexity measure $\mathcal{C}(h)$. Interestingly, this allows recasting the obtained decision function as follows.

Theorem 2.9 (Representer theorem). *(based on (Schölkopf and Smola, 2002, Theorem 4.2)) Each minimizer $f \in \mathcal{H}$ of the regularized risk functional as defined in Eq. (2.12) also admits a representation of the form:*

$$f(\mathbf{x}_*) = \sum_{i=1}^N \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_*) \quad (2.111)$$

with $\alpha_i \in \mathbb{R}$. Thus, the decision function can be rewritten as a weighted summation of kernel functions centered on training examples $\mathbf{x}_i, i = 1, \dots, N$.

Note that the result of Theorem 2.9 also gives insights into the origins of the name “RBF kernel”: each kernel evaluation of $\kappa_{\text{RBF}}(\cdot, \mathbf{x}_i)$ equals a 1D Gaussian centered at $\mathbf{x}_i$. Since every two resulting Gaussians differ for distinct training inputs, they serve as basis functions to build a final decision $f(\mathbf{x}_*)$. The dependence on a radial distance (e.g., the Euclidean distance) finally gives rise to the name RBF kernel.

In summary of the previous results, the power of kernel tools lies in three aspects: efficient computations of inner products in possibly high-dimensional feature spaces (Mercer’s theorem), interchangeability of mappings (kernel trick), and

recasting of decision functions as weighted combination over kernel evaluations on training examples (representer theorem).

2.1.5.6 An Alternative Interpretation on GP: the Weight Space View

It is insightful to note that the previously introduced concepts of feature embeddings allow for approaching GP models from a different perspective.

Bayesian Linear Regression: Gaussian process regression can also be derived as Bayesian non-linear regression with specific assumptions on the distributions of weights and noise. To see this, let us start with the ordinary parametric model for linear regression as introduced in Section 2.1.4.3:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} . \quad (2.112)$$

For brevity, we skipped explicitly mentioning offsets and instead include them by augmenting $\mathbf{x}$ with constant one. We again assume white Gaussian noise on observable inputs:

$$y = \mathbf{w}^T \mathbf{x} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) . \quad (2.113)$$

This can accidentally lead to linear non-separable data as encountered for soft margin classifiers. However, instead of seeking a single vector $\mathbf{w}$ which optimizes some criterion (as done for SVMs), we will follow a Bayesian perspective by marginalizing over $\mathbf{w}$ instead. For that, we need to place a prior on the distribution of $\mathbf{w}$. Again, an intuitive assumption is a Gaussian distribution:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p) \quad (2.114)$$

with given prior covariance matrix Σ_p . We now follow the marginalization idea with assumptions regarding conditional independence as introduced in Eq. (2.30). Thereby, the prediction f_* for an unseen test example $\mathbf{x}_*$ can be obtained as:

$$p(f_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(f_* | \mathbf{x}_*, \mathbf{w}) p(\mathbf{w} | \mathbf{X}, \mathbf{y}) d\mathbf{w} \quad (2.115)$$

$$= \mathcal{N}\left(\frac{1}{\sigma_n^2} \mathbf{x}_*^T \mathbf{C}^{-1} \mathbf{X} \mathbf{y}, \mathbf{x}_*^T \mathbf{C}^{-1} \mathbf{x}_*\right) , \quad (2.116)$$

where $\mathbf{C}$ denotes the covariance matrix which is given by:

$$\mathbf{C} = \frac{1}{\sigma_n^2} \mathbf{X} \mathbf{X}^T + \Sigma_p^{-1} . \quad (2.117)$$

Note that for brevity, we skipped involved details, which can be found in (Rasmussen and Williams, 2006, Sect.2.1).

Kernelizing Linear Regression: Since a linear model is often too restrictive, we apply the same idea as done for kernelizing SVM models and project data into a high-dimensional space. Due to the increase of dimensionality, we can expect a linear model to result in useful predictions with respect to the original input space. As previously, we transform features $\Phi(x)$ with $\Phi : \Omega \rightarrow \mathcal{H}$. Each occurrence of x within the linear regression model is thus replaced by its transformed counterpart. This leads to

$$f(x) = w^T \Phi(x) \quad (2.118)$$

where w is now of dimension D' . Analogously, the marginalization translates to

$$p(f_* | x_*, X, y) = \mathcal{N} \left(\frac{1}{\sigma_n^2} \Phi(x_*)^T C^{-1} \Phi(X) y, \Phi(x_*)^T C^{-1} \Phi(x_*) \right), \quad (2.119)$$

with corresponding modifications also in the covariance matrix

$$C = \frac{1}{\sigma_n^2} \Phi(X) \Phi(X)^T + \Sigma_p^{-1}. \quad (2.120)$$

By going from Eq. (2.116) to Eq. (2.119), we still assumed w to be distributed according to a white Gaussian noise model but now with respect to the feature space $\mathcal{H}$. Thus, Σ_p^{-1} is now of dimension $\mathbb{R}^{D' \times D'}$. but still shows the strict diagonal form due as imposed by the white Gaussian noise assumption.

As we can see, computing the explicit solution in feature space is extremely costly. This is mainly caused by the inversion of the transformed covariance matrix of size $D' \times D'$ in Eq. (2.119). To compensate this drawback, we use the previously introduced concepts of kernels. Therefore, we rewrite Eq. (2.119) such that operations on transformed features only appear as inner products in space $\mathcal{H}$ and with respect to Σ_p . To keep notation simple, let us introduce $\phi_* = \Phi(x_*)$ and $\Phi = \Phi(X)$. The predictive distribution can then be written as follows (see (Rasmussen and Williams, 2006, p.12) for detailed derivations):

$$p(f_* | x_*, X, y) = \mathcal{N} \left(\phi_*^T \Sigma_p \Phi (\Phi^T \Sigma_p \Phi + \sigma_n^2 I)^{-1} y, \phi_*^T \Sigma_p \phi_* - \phi_*^T \Sigma_p \Phi (\Phi^T \Sigma_p \Phi + \sigma_n^2 I)^{-1} \Phi^T \Sigma_p \phi_* \right). \quad (2.121)$$

If we now define $\Sigma_p^{\frac{1}{2}}$ such that $\Sigma_p^{\frac{1}{2}} \Sigma_p^{\frac{1}{2}} = \Sigma_p$, and $\Psi(x) = \Sigma_p^{\frac{1}{2}} \Phi(x)$, we can express each occurrence of transformed inputs in Eq. (2.121) as evaluation of a suitable kernel function $\kappa_\Psi(x, x') = \langle \Psi(x), \Psi(x') \rangle$. Thus, we can finally rewrite Eq. (2.121)

as:

$$p(f_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N} \left(\kappa_\Psi(\mathbf{X}, \mathbf{x}_*)^T (\kappa_\Psi(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \right. \\ \left. \kappa_\Psi(\mathbf{x}_*, \mathbf{x}_*) - \kappa_\Psi(\mathbf{X}, \mathbf{x}_*)^T (\kappa_\Psi(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \kappa_\Psi(\mathbf{X}, \mathbf{x}_*) \right). \quad (2.122)$$

We immediately observe the correspondence to results for μ_* and σ_*^2 which we obtained from the perspective of random processes and distribution over functions (see Eq. (2.99) and Eq. (2.100)). Note that the correspondence is even more striking if we assume $\Sigma_p = \mathbf{I}$ in absence of further knowledge. In summary, Gaussian process regression can also be interpreted as kernelized counterpart of Bayesian linear regression under specific assumptions on the distribution of $\mathbf{w}$ and ε .

2.1.5.7 Hyperparameter Optimization for GP

In Section 2.1.3.2, we were interested in determining a set of parameters θ_*^{ML} for parametric models such that the resulting model could optimally explain observed data. We therefore introduced the technique of maximizing the data likelihood $p(\mathbf{y} | \mathcal{X}, \theta)$, possibly re-weighted by a prior term which lead to MAP estimates θ_*^{MAP} . As a full Bayesian alternative, we additionally noticed the marginalization over the entire parameter space where no plug-in of parameters was required anymore. In this section, we are interested in transferring these ideas to non-parametric models.

The Marginal Likelihood for GP Regression: Since Gaussian process models are non-parametric, no parameters need to be optimized. Nonetheless, covariance functions and noise models are commonly parameterized. To stress the difference to parametric approaches, these parameters are called *hyperparameters*. We already noticed the strong dependence of model outputs on the values of these hyperparameters in Fig. 2.10 and Fig. 2.11. In direct consequence, techniques for a suitable treatment of these hyperparameters are required.

As for the parametric case, we again have two options. The first path is to follow a full Bayesian treatment, by introducing prior distributions over hyperparameters (so-called *hyperpriors*) and integrating over functions and hyperparameters simultaneously. Alternatively, we can follow the idea of model comparison (see Section 2.1.3.4) by maximizing the marginal likelihood over a set of possible hyperparameters. This option is also known as *empirical Bayes* or *type-2 maximum likelihood*.

Let us inspect the second option in more detail. For notational brevity, let us collect all hyperparameters in a vector η and let $\mathcal{L} = (\mathcal{X}, \mathbf{y})$ denote the collected training data. From the definition of GP models (see Eq. (2.85) and Eq. (2.90)), it

follows that the negative log-likelihood can be expressed as:

$$-\log p(\mathbf{y} | \mathbf{X}, \boldsymbol{\eta}) = \frac{1}{2} \log \det (\mathbf{K}_{\boldsymbol{\eta}}) + \frac{1}{2} \mathbf{y}^T \mathbf{K}_{\boldsymbol{\eta}}^{-1} \mathbf{y} + \frac{N}{2} \log (2\pi) \quad . \quad (2.123)$$

We explicitly put a subscript on the kernel matrix to indicate its dependence on the current specification of $\boldsymbol{\eta}$ (see also (Rasmussen and Williams, 2006, Sect. 5.4)). It is insightful to compare Eq. (2.123) with the simpler version for multivariate Gaussian distributions as derived in Eq. (2.41). For a set of possible hyperparameter values, we can now evaluate Eq. (2.123) and select the best setting.

Including White Gaussian Noise in Kernel Functions: Note that the computations in Eq. (2.123) remain almost the same for noisy labels under the assumption of white Gaussian noise (see Eq. (2.96)). The only necessary adaptation is to replace the kernel matrix $\mathbf{K}$ by its regularized variant $\tilde{\mathbf{K}} = (\mathbf{K} + \sigma_n^2 \mathbf{I})$ (see Eq. (2.98)). Nonetheless, we can skip the additional notation since adding white Gaussian noise can also be viewed as changing the covariance function to:

$$\tilde{\kappa}(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}') + \sigma_n^2 \delta_{\mathbf{x}, \mathbf{x}'} \quad . \quad (2.124)$$

According to Section 2.1.5.4, this again results in a psd kernel. Thus, σ_n^2 can be absorbed in $\boldsymbol{\eta}$ for notational convenience. In consequence, we can skip explicitly adding σ_n^2 when computing the predictive variance since it is already contained in the self-similarity k_{**} .

Applications of Hyperparameter Optimization: Besides optimizing the bandwidth parameters σ or the noise parameter σ_n^2 , an important application lies in determining the relevance of input dimensions. This approach is known as *automatic relevance determination (ARD)* and has been introduced by Neal (1995) and Williams and Rasmussen (1996). A successful realization of ARD is possible if the kernel function offers a learnable, non-negative weight parameter for each feature dimension. A straight-forward example is to add a dimension-specific weighting scalar in additive kernels:

$$\kappa_{\text{ARD}}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^D \boldsymbol{\eta}(d) \cdot \kappa_d(\mathbf{x}(d), \mathbf{x}'(d)) \quad . \quad (2.125)$$

Based on the marginal likelihood given in Eq. (2.123), we can thereby find optimal weights for every dimension. We show an application of ARD in Section 7.2.3.5.

Furthermore, we can even combine different feature types, which is known as *multiple kernel learning (MKL)*. Again, a simple combination is a weighted

summation of kernel functions specific for each feature type:

$$\kappa_{\text{MKL}}(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^K \eta_k \cdot \kappa_k(\mathbf{x}_{(k)}, \mathbf{x}'_{(k)}) \quad (2.126)$$

with $\mathbf{x} = (\mathbf{x}_{(1)}^T, \dots, \mathbf{x}_{(K)}^T)^T$. As reviewed in Section 2.1.5.4, $\kappa_{\text{MKL}}(\cdot, \cdot)$ is a psd kernel if all $\kappa_k(\cdot, \cdot)$ are so, too. Thus, optimizing weights for each feature type for feature combination is another important benefit in practical applications.

How To Optimize The Marginal Likelihood: Without further assumptions, minimizing Eq. (2.123) with respect to $\boldsymbol{\eta}$ can be done by exhaustive grid search techniques or derivative-free optimization methods. One example is the *downhill-simplex algorithm* by Nelder and Mead (1965) which we will apply in Section 3.3. Note that the likelihood function in Eq. (2.123) is usually non-convex, hence, multiple local optima are likely to occur. Depending on the initialization, minimizing Eq. (2.123) can result in different solutions.

However, globally optimal solutions can be found by computing the corresponding partial derivatives and setting them to zero:

$$\frac{\partial}{\partial \eta_i} \log p(\mathbf{y} | \mathcal{X}, \boldsymbol{\eta}) = \frac{1}{2} \mathbf{y}^T \mathbf{K}_{\boldsymbol{\eta}}^{-1} \frac{\partial \mathbf{K}_{\boldsymbol{\eta}}}{\partial \eta_i} \mathbf{K}_{\boldsymbol{\eta}}^{-1} \mathbf{y} - \frac{1}{2} \text{tr} \left(\mathbf{K}_{\boldsymbol{\eta}}^{-1} \frac{\partial \mathbf{K}_{\boldsymbol{\eta}}}{\partial \eta_i} \right) \quad (2.127)$$

$$= \frac{1}{2} \text{tr} \left((\boldsymbol{\alpha} \boldsymbol{\alpha}^T - \mathbf{K}_{\boldsymbol{\eta}}^{-1}) \frac{\partial \mathbf{K}_{\boldsymbol{\eta}}}{\partial \eta_i} \right) \quad (2.128)$$

See (Bishop, 2006, Appendix C) for an overview of computation rules for matrix derivatives as well as (Rasmussen and Williams, 2006, Sect. 5.4) for detailed derivations. Note that Eq. (2.127) is only well-defined for continuous hyperparameters and differentiable covariance functions with respect to each hyperparameter η_i . An example for computing derivatives of the Gaussian kernel with respect to the length scale parameter can be found in (Rodner, 2011, Sect. 2.6.10).

2.1.5.8 GP Models for Classification

In the previous section, we introduced GP models for regression and aimed at continuous relations between inputs and outputs. For visual recognition scenarios, however, we often require classification decisions rather than continuous scores of arbitrary scale. Consequently, our desired functions do not longer map into the entire real axis $\mathbb{R}$. Instead, our target space is the discrete set $\mathcal{Y} = \{1, \dots, M\}$ for hard decisions or class-specific probabilities $\mathcal{Y} = [0, 1]^M$ in a probabilistic setting.

GP Classification in Binary Scenarios: Let us start by looking into the binary case. In terms of the GP framework, we still assume a Gaussian process prior on the underlying functional relationship f as in Eq. (2.86), *i.e.*, $f \sim \mathcal{GP}(m(\cdot), \kappa(\cdot, \cdot))$. However, we replace the white Gaussian noise in Eq. (2.91) with different *transfer functions* $\varepsilon(f(\mathbf{x}))$ which are suited for classification. A discrete decision function can then be obtained as:

$$\hat{h}(\mathbf{x}_*) = \varepsilon(f(\mathbf{x})) \quad . \quad (2.129)$$

One straight-forward transfer function is the *round-off noise*:

$$\varepsilon_{-1,1}(f(\mathbf{x})) = \begin{cases} 1 & \text{if } f(\mathbf{x}) > c \\ -1 & \text{else} \end{cases} \quad (2.130)$$

for binary labels $y \in \{-1, 1\}$ and a decision threshold $c \in \mathbb{R}$. Note that thresholding against zero, *i.e.*, $\varepsilon_{-1,1}(\cdot) = \text{sgn}(\cdot)$, would intuitively follow from a zero mean assumption. However, different thresholds might be better suited in certain scenarios, *e.g.*, for imbalanced class sizes during training. Another popular alternative is placing a continuous transfer function which maps to the unit interval $[0, 1]$, *e.g.*, a logistic sigmoid as given in Eq. (2.80). Thus, transformed outputs can be interpreted as probability estimate.

For the majority of useful transfer functions, the resulting integrals in Eq. (2.95) can no longer be solved in closed-form. In consequence, we face two general options: either solving integrals only approximately (by analytic or numeric approximations), or discarding the discrete nature of class labels to retain computational tractability. For the first direction, analytic approximation techniques such as *Laplace approximation* by Williams and Barber (1998), or *Expectation-Propagation* by Minka (2001) are most famous. Numeric approximations based on Monte Carlo techniques have been introduced by Neal (1998). Since these techniques are still computationally expensive, they will play no role within the remainder of this thesis. Thus, we skip further derivations here and instead refer the interested reader to (Rasmussen and Williams, 2006) and (Nickisch and Rasmussen, 2008) for detailed introductions.

We will instead follow the second option by directly applying GP regression techniques even if labels are discrete. Thus, we will use the continuous outputs for final decisions as already done for SVMs (see Eq. (2.73)). Introduced by Nickisch and Rasmussen (2008) as *label regression* and based on the regularized least-square classification as proposed by Rifkin and Klautau (2004), this technique gained further popularity by applying it to visual recognition problems in (Kapoor et al., 2010). In Section 7.2.1, we experimentally justify the utilization of GP label regression by comparing the resulting classification performance of label regression against Expectation-Propagation and Laplace approximation.

GP Classification in Multi-class Scenarios: As mentioned previously, lifelong learning systems will sooner or later need to cope with more than two known categories. Unfortunately, category labels are by definition categorical instead of ordinal, *i.e.*, there exists no intrinsic ordering among classes. Thus, simple regression on multi-class labels is obviously unfeasible.

A theoretically sound solution comes with probabilistic alternatives. An example is the combination of soft max, which is the multi-class generalization of the logistic function, and the Laplace approximation as introduced by Williams and Barber (1998). However, a simpler solution lies in the previously introduced techniques of one-vs-one and one-vs-all. Due to computational benefits and shorter formulas, we will follow the latter one throughout this thesis. Let us therefore introduce for every category $m \in \{1, \dots, M\}$ a category-specific binary label vector $\mathbf{y}_m$ with $\mathbf{y}_m(i) = 1$ if $\mathbf{y}(i) = m$ and zero otherwise:

$$\mathbf{y}_m(i) = 2 \cdot \delta_{\mathbf{y}(i), m} - 1 . \quad (2.131)$$

A classification decision for unseen test inputs $\mathbf{x}_*$ can then be obtained by returning the class with highest predictive mean:

$$\hat{h}(\mathbf{x}_*) = \operatorname{argmax}_{m \in \{1, \dots, M\}} \mu_*^m \quad (2.132)$$

$$= \operatorname{argmax}_{m \in \{1, \dots, M\}} \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{y}_m . \quad (2.133)$$

On closer inspection of Eq. (2.132), we notice that the one-vs-all scheme allows for sharing information between categories since the kernel matrix $\mathbf{K}$ only needs to be computed once. This aspect gains further importance when new categories occur over time. Since the kernel matrix will also be shared with new categories, most of the computations for model training can be skipped. As we will show in Chapter 4, we can reuse the old kernel matrix and only need to perform partial updates.

Furthermore, we already observed that the predictive variance is independent of target labels (see Eq. (2.100)). Thus, the uncertainty of predictions holds uniformly over classes for a shared kernel matrix. Finally, if we seek for global hyperparameters which are shared by all binary tasks, the negative log-likelihood as introduced in

Eq. (2.123) can be optimized jointly over all M binary models:

$$\boldsymbol{\eta}^{\text{ML}} | \mathbf{y}, \mathcal{X} = \underset{\boldsymbol{\eta}}{\operatorname{argmax}} p(\mathbf{y}_1, \dots, \mathbf{y}_M | \mathcal{X}, \boldsymbol{\eta}) \quad (2.134)$$

$$= \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \sum_{m=1}^M \left(\frac{1}{2} \log \det(\mathbf{K}_{\boldsymbol{\eta}}) + \frac{1}{2} \mathbf{y}_m^{\text{T}} \mathbf{K}_{\boldsymbol{\eta}}^{-1} \mathbf{y}_m + \frac{N}{2} \log(2\pi) \right) \quad (2.135)$$

$$\propto \underset{\boldsymbol{\eta}}{\operatorname{argmin}} M \cdot \log \det(\mathbf{K}_{\boldsymbol{\eta}}) + \sum_{m=1}^M \mathbf{y}_m^{\text{T}} \mathbf{K}_{\boldsymbol{\eta}}^{-1} \mathbf{y}_m . \quad (2.136)$$

Note that from Eq. (2.134) to Eq. (2.135), we assumed conditional independence between binary tasks given hyperparameters $\boldsymbol{\eta}$ as suggested by Lawrence et al. (2005). This assumption allowed for successful applications in image categorization tasks as shown by Kapoor et al. (2010).

2.1.5.9 A Note on Computational Issues

If we want to bring machine learning techniques to practical use, their computational efficiency is at least as important as error convergence rates or associated probabilistic statements. Thus, scalability of formulations with respect to computation time and memory consumption are crucial, especially with respect to lifelong learning applications.

Asymptotic Memory Consumption: In its plain formulation, GP models need to store the kernel matrix and all training examples which requires $\mathcal{O}(N^2 + DN)$ memory. Additionally, storing the weight vector $\boldsymbol{\alpha}$ requires $\mathcal{O}(N)$ memory for every category. In summary, this leads to an overall memory consumption of $\mathcal{O}(N^2 + N(D + M))$. For large-scale datasets with thousands of examples and hundreds of categories, this asymptotic behavior is undoubtedly unfeasible.

To reduce memory consumption, different techniques have been developed, *e.g.*, to select an informative subset of all available data points, to enforce sparsity of $\mathbf{K}$, to compute kernel values on-the-fly, or to approximate $\mathbf{K}$ by a diagonal matrix as we introduced in (Bodesheim et al., 2013a). For an overview of related techniques, we refer to (Quiñonero Candela and Rasmussen, 2005), (Nickisch and Rasmussen, 2008), and the review presented in Section 3.1. In addition, note that the memory demand during testing can be reduced to $\mathcal{O}(N(D + M))$ if only the predictive mean estimate is of interest.

Furthermore, we present in Chapter 3 how to obtain the full and exact predictive distribution while requesting only $\mathcal{O}(MDN)$ memory. Our techniques are based on the choice of κ_{HIK} as kernel function (or generalizations thereof) and are applicable to all stages of inference, including training, optimization of hyperparameters,

classification, and variance prediction.

Asymptotic Computation Time: The computation time during training is mainly dominated by computing the kernel matrix $\mathbf{K}$ and its inverse. To perform both steps, $\mathcal{O}(DN^2 + N^3)$ operations are required.

It is well known that for numerical reasons, matrix inversion should be avoided whenever possible. Thus, numerically robust alternatives should be applied, *e.g.*, using the Cholesky decomposition $\mathbf{K} = \mathbf{L}\mathbf{L}^T$ (see Section 4.2.2 for details). Note that the asymptotic computation time is thereby not affected. To further increase numerical stability, it is beneficial to explicitly increase the condition of $\mathbf{K}$ by adding a diagonal matrix $\lambda\mathbf{I}$ with a small positive scalar λ . For sufficiently large λ , all eigenvalues of $\tilde{\mathbf{K}}$ are guaranteed to be significantly positive which allows for robust inversion or decomposition. Note that this is implicitly done when we expect that labels are disturbed by white Gaussian noise (see Eq. (2.98) in Section 2.1.5.3).

Once the kernel matrix is inverted or decomposed, the M weight vectors α_m can be obtained in $\mathcal{O}(MN^2)$ operations. This leads to $\mathcal{O}(N^3 + (D + M)N^2)$ as overall complexity during training.

With respect to the optimization of hyperparameters, we already noticed two alternatives: (i) directly optimizing the marginal likelihood in Eq. (2.123) using grid search techniques, or (ii) computing derivatives in Eq. (2.127) with respect to involved hyperparameters and setting them to zero. The first technique requires computing and inverting one kernel matrix for each hyperparameter setting, and thus scales with $\mathcal{O}(N^3)$. In contrast, the second version is usually more efficient and only requires $\mathcal{O}(N^2)$ per evaluation of the gradient once the kernel matrix is computed (see also (Rasmussen and Williams, 2006, Chapter 5.4) for a detailed discussion). However, it is only applicable to differentiable kernel functions and continuous hyperparameters.

Let us finally look into required computation times for predictions on unseen test examples. Since computing the predictive mean and the predictive variance requires evaluating a linear and quadratic function, the prediction time scales asymptotically with $\mathcal{O}(MDN)$ and $\mathcal{O}(N^2 + DN)$, respectively. This already includes kernel evaluations between a new example and all training examples.

In Chapter 3, we present techniques to reduce the required computation times for all involved steps. By exploiting κ_{HK} as covariance function, the computation time for training and optimization of hyperparameters can be reduced to $\mathcal{O}(DN \log N)$ and $\mathcal{O}(DN)$. Computing the predictive variance can be done with $\mathcal{O}(N)$ or $\mathcal{O}(D)$ operations. Finally, computing the predictive mean only requires $\mathcal{O}(MD \log N)$ or $\mathcal{O}(MD)$ operations.

2.1.5.10 Connections to Other Methods

It should be noted that Gaussian process models have been co-developed in the (geo)statistics community under the name *kriging* (Cressie, 1990). A special case of GP models are spatial random fields which are popular in earth science (Christakos and Papanicolaou, 2000). Furthermore, GP models are closely related to Kalman filters (Sorenson, 1970) and neural networks (Bishop, 2006, pp. 319). In Section 2.1.5.6, we have already seen the connection of Bayesian linear regression and GP models via kernelization. Furthermore, from the point of view of regularized risk minimization (see Section 2.1.2.3), Gaussian process models are directly related to SVM formulations with a squared loss instead of the commonly chosen Hinge loss. For a more detailed discussion on connections of Gaussian process models to other methods, we refer the interested reader to (Kemmler, 2013, pp. 60) or (Murphy, 2012, Chapter 15.4).

2.2 Image Representations for Visual Object Recognition

In the previous section, we introduced machine learning for discovering unknown relations between inputs $x \in \Omega$ and outputs $y \in \mathcal{Y}$. We derived several classification techniques and shed light on algorithmic details and resulting advantages. However, we always assumed the feature space Ω to be given. For a successful application in visual object recognition scenarios, powerful image representations are of utmost importance. Intuitively, image representations need to capture relevant details of images, *e.g.*, parts of objects or scene properties. Thus, it is not obvious in advance which representation to choose. Even worse, the adequacy of different image representations might depend on the task defined by $\mathcal{Y}$. Therefore, we will give a short overview on popular and powerful image representations in this section. For the scope of this thesis, we will mainly focus on representations which have proven to be especially suited for visual object recognition.

2.2.1 Local Features

To describe images, an intuitive approach is to first describe smaller parts individually, and to build a combined representation thereof later on. For the description of image parts, *local features* have been popular in computer vision scenarios. A local feature is defined as bi-tuple $lf = (p, d)$, with position information p and actual description d of the sub-image defined by p . During the last decades, a variety of different types of local features has been developed. In the following, we briefly review a subset of them which are used within this thesis.

Computing d – How to Describe Local Regions: One of the most frequently used techniques for local feature computation is the *scale-invariant feature transformation (SIFT)* as introduced by Lowe (2004), SIFT is probably today's most cited article in computer vision⁵. The main idea behind SIFT's descriptor is to calculate image gradients, to locally pool gradients into histograms by quantizing gradient orientations, and to normalize histograms individually for local sub-images. Since the SIFT algorithm has been reviewed extensively in a vast number of PhD thesis (e.g., (Rodner, 2011)), we skip an introduction here. Instead, we refer the interested reader to (Lowe, 2004).

A visualization of SIFT descriptors is given in Fig. 2.12a. Circles encode positions p obtained with SIFT's detector (see following paragraph), and star-like structures represent gradient orientations occurring within each cell of d . An extension of SIFT to color images is possible via OpponentSIFT which extracts SIFT descriptors in opponent color space (van de Sande et al., 2010).

Dalal and Triggs (2005) introduced *histogram of oriented gradients (HOG)*, which is a descriptor similar to SIFT. Although HOG was originally introduced for object detection, it mainly differs from SIFT in how assignment of gradient orientations and local normalization is done. The version currently used most often was provided by (Felzenszwalb et al., 2010). A visualization of HOG features extracted on a dense grid is given in Fig. 2.12b.

As a third technique for local descriptions, we want to mention histograms of color probabilities. In (Van De Weijer et al., 2009), the authors propose to transform every color given as RGB triple into a probability vector showing how likely the color would be named as one of all eleven nameable colors existing in the English language. Local representations can then again be obtained by pooling of responses within a specified region. In consequence, the authors call their descriptor *color name (CN)*. A visualization is given in Fig. 2.12c where every color is mapped to its most likely counterpart of nameable colors. Van De Weijer et al. (2009) additionally show how to learn the non-linear color mapping based on images obtained by Google image search with color names as search queries.

Note that there exist a huge number of further description techniques, e.g., SURF (Bay et al., 2006), BRIEF (Calonder et al., 2010), or ORB (Rublee et al., 2011) to name just a few. However, we will skip reviewing further techniques as they are not relevant for the remaining part of this thesis.

Finding p – Which Regions to Describe: In general, two strategies for finding positions p for local feature extraction exist: (i) compute a sparse set of key points using interest point detectors, or (ii) compute a dense grid of keypoints. Early approaches followed the first line and aimed at locating positions which were

⁵According to Google scholar, (Lowe, 2004) received more than 32,000 citations so far.

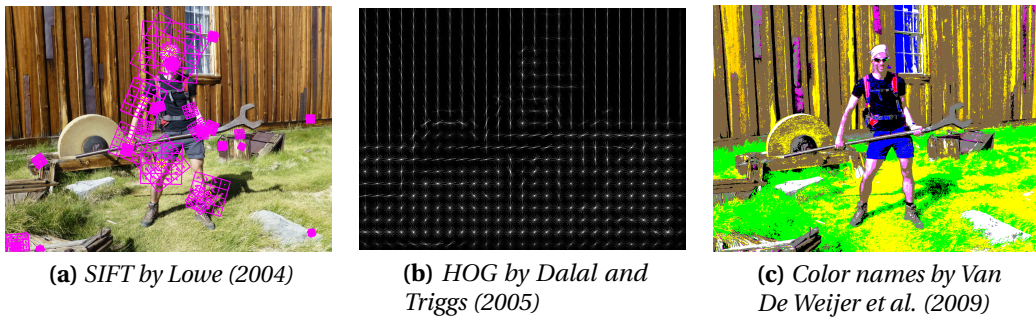


Figure 2.12: Visualization of famous local feature descriptors. Left: SIFT descriptors extracted at some points of interest - computed with code provided by VLFeat (Vedaldi and Fulkerson, 2008). Middle: an array of HOG cells - computed with code adapted from (Felzenszwalb et al., 2010). Right: RGB triples are mapped to their most likely, nameable color - computed with code provided by (Van De Weijer et al., 2009). The original image is given in Fig. 1.1.

regarded as informative for the current object category. Well-known techniques are the prominent SIFT detector (Lowe, 2004) or FAST (Rosten and Drummond, 2006). Later on, Jurie and Triggs (2005) reported that replacing interest point detectors by random sampling leads to comparable results. Furthermore, Nowak et al. (2006) reported that recognition rates are strongly coupled with the number of extracted local features. Since this number can not be easily increased when using interest point detectors, dense feature extraction should be preferred. Zhang et al. (2007) further showed that the classification accuracy is even decreased by adding invariance properties to detectors, e.g., invariance with respect to rotation and affine transformation.

2.2.2 Bag-of-visual-words Models

Given a set of local features, it remains to build a single representation thereof. The the probably most widely used approach for doing so is the *bag-of-visual-words* (BoW) model. It is inspired by language processing and text classification, where it was found that document classes can be identified by words which only occur frequently in documents of the same class (Joachims, 1998). As with several other techniques, the idea was transferred into the image domain and named as bag-of-*visual*-words (Csurka et al., 2004). A visualization is given in Fig. 2.13a.

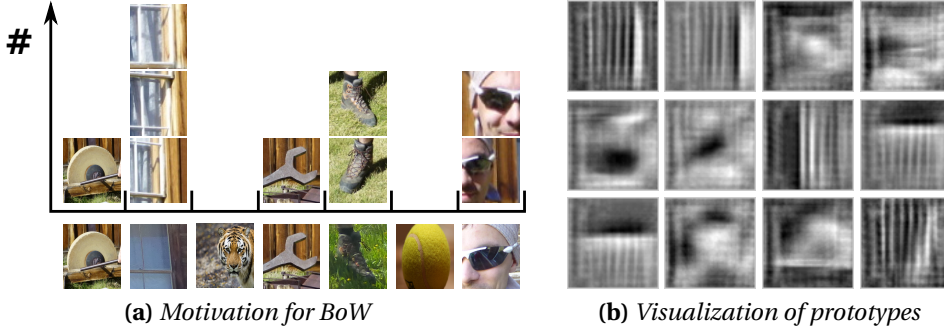


Figure 2.13: Left: The Bag-of-visual-words approach is adapted from natural language processing and aims at describing images based on occurrence statistics of local image patches. Right: Found prototypes are hardly human understandable. See Sect. 2.2.2 for details.

Quantization: Following the BoW-principle, local features are assigned into a precomputed codebook C which consists of prototypes⁶. Thus, every local features is associated with the most similar visual words. Traditionally, every local feature (p_i, d_i) is hard-assigned via v_i to one of the $\mathfrak{K}$ prototypes:

$$v_i \in \{0, 1\}^{\mathfrak{K}} \text{ and } \|v_i\|_1 = 1 . \quad (2.137)$$

Assignment is usually done to the prototype which is closest in feature space under a given distance function. Since feature vectors are thereby quantized to a single scalar, this process is also known as vector quantization. It should be noted that the position within the image is of no interest during quantization.

In contrast to hard assignments, van Gemert et al. (2010) and Coates et al. (2011) showed that a soft assignment can lead to higher recognition performance:

$$v_i \in [0, 1]^{\mathfrak{K}} \text{ and } \|v_i\|_1 = 1 . \quad (2.138)$$

The intuitive reason in the introduced flexibility which is especially important for ambiguous codebooks. In this case, assignments are usually done by transforming distances between d_i and prototypes into weights such that small distances result in high weights and *vice versa*. As an example, note that the well-known locality-constrained linear coding scheme (LLC) as proposed by Wang et al. (2010) equals the selection of a subset of highly similar prototypes for every local feature followed by soft-voting with respect to those prototypes only.

⁶For the ease of mathematical notation, we assume the codebook C to be built by stacking together $\mathfrak{K}$ prototypes into a $D \times \mathfrak{K}$ -matrix. However, there is no ordered relationship between prototypes. Hence, a representation as set of prototypes would be adequate, too.

Interestingly, the common color representations in computers as a weighting of R-channel, G-channel, and B-channel can also be interpreted as soft assignments of descriptors (colors) into a pre-defined codebook (consisting of three basis color vectors). As a final remark, we adhere that more sophisticated encoding schemes exist that do not only assign local features into a given codebook. Two popular examples are VLAD (Arandjelovic and Zisserman, 2013; Jégou et al., 2010) or Fisher vectors (Perronnin and Dance, 2006; Perronnin et al., 2010) which additionally add the difference between each feature and assigned prototypes into the representation.

Spatial Pooling: After extracting $\mathcal{N}$ local features from a given image and assigning them into a codebook, a final histogram can be computed by spatially pooling assignments over the entire image:

$$\mathbf{x}_{\text{BoW}} \left((\mathbf{p}_i, \mathbf{d}_i)_{i=1}^{\mathcal{N}} \right) = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \mathbf{v}_i . \quad (2.139)$$

Thus, any spatial information $\mathbf{p}$ is discarded, which is regarded as efficient option for obtaining occlusion-invariant features. However, simple spatial extensions based on constraining pooling to specified areas exist which re-incorporate rough spatial information. The most prominent technique here is spatial pyramid matching by Lazechnik et al. (2006). Thereby, images are divided in increasingly finer cells, pooling is performed for each cell individually, and final representations are obtained by histogram concatenation.

Learning of Codebooks: So far, we considered the codebook $\mathbf{C}$ as pre-defined and given. Thereby, we were able to focus on assigning local features to a subset of given prototypes. A typical example for a given codebook in practice is the quantization of colors, where the RGB basis is defined in color standards. In contrast, codebooks for visual object recognition are usually *learned* from data.

A prominent set of techniques for learning codebooks is (un)supervised clustering of local features. Among this family, one of the simplest techniques is *K-means*⁷. It can be shown that which minimizes the codebook's reconstruction error with respect to collected data:

$$(\hat{\mathbf{C}}, \{\hat{\mathbf{v}}_i\}) = \underset{\substack{\mathbf{C} \in \mathbb{R}^{D \times \mathcal{K}}, \{\mathbf{v}_i\} \in \{0,1\}^{\mathcal{K}} \text{ and } \|\mathbf{v}_i\|_1 = 1}}{\text{argmin}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{d}_i - \mathbf{C}\mathbf{v}_i\|_2 . \quad (2.140)$$

⁷A great overview on K-means and further clustering techniques can be found in (MacKay, 2003, pp. 285) or (Bishop, 2006, pp. 424).

It should be noted that the desired number of prototypes K is usually orders of magnitudes smaller than the number N of local features extracted from all training images. As direct consequence, the learned prototypes will tend to represent frequently occurring background structures and often fail at capturing discriminative local features. In fact, it was shown by Nowak et al. (2006) that even random selection during codebook creation leads to similar performance in many scenarios. Rühle et al. (2015) additionally confirmed this observation for automotive scenarios. This phenomenon was explained in (Jia et al., 2013, p. 6) by drawing parallels to the Nyström method (Kumar et al., 2012b). Due to the moderate results of K-means clustering, different techniques for codebook computations have been developed. We refer to Section 6.1.1 for a review of related work on codebook learning.

On the Semantic Interpretability of BoW Representations: In contrast to what the name suggests, BoW codebooks for low-level features do usually not correspond to semantically rich object parts. Instead, they mainly correspond to tiny sub-images which hard to interpret at all and not related to semantic words. We visualized this aspect in Fig. 2.13b by clustering densely sampled HOG features and visualized the learned prototypes using the technique introduced by Vondrick et al. (2013).

This observation is consistent with David MacKays statement: *“In vector quantization, we do not necessarily believe that the templates [prototypes] have any natural meaning; they are simply tools to do a job.”* (MacKay, 2003, p.285). For the rest of this thesis, we can therefore consider BoW representations as discriminative low-level statistics rather than semantically rich representations – a fact we believe is worth being pointed out.

2.2.3 Deep-learned Image Representations

The previously reviewed framework for image classification using bag-of-visual-words models can be summarized as follows: (i) extraction of local features, (ii) assignment of local features to a learned set of prototypes and spatial pooling of assignment results, and (iii) training of the final classifier. On a closer look, though, the first two steps internally follow a surprisingly similar processing pipeline.

Shallow or Deep: Pipelines for Visual Object Recognition: To better see this pipeline, let us exemplarily inspect SIFT as technique for local feature extraction. Descriptors are computed on a dense grid by extracting image gradients (obtained with gradient filters) and assigning them to a fixed set of pre-defined orientations. Assignments are then non-linearly pooled over small image regions, leading to the cell structure as visualized in Fig. 2.12a. During vector assignment, almost the

same routines are applied, but on a different level of complexity and with different parameters. Distances between each local feature of the previous step and each prototype are computed and assignments are then spatially pooled to form the BoW histogram. The image representation obtained after these two processing stages is finally fed into the trained classifier.

As we can see, we essentially have a feed-forward architecture with three stages. Similar observations have been presented by Sydorov et al. (2014) for image classification with Fisher vectors. Unfortunately, all stages are specified independently in advance. Thus, important aspects need to be done manually, *e.g.*, choosing an informative set of local feature descriptors as codebook or selecting a suitable algorithm for codebook learning with useful parameter settings. For this reason, techniques which follow this general layout are referred to as *shallow learning*.

Intuitively, performing each step independently can easily result in sub-optimal behavior. Instead, we would favor algorithms which perform all steps simultaneously by maximizing a single criterion, *e.g.*, the resulting classification accuracy. Interestingly, techniques which follow this approach are known for several decades, *e.g.*, deep artificial neural networks with multiple hidden layers. While a detailed review of the entire field goes far beyond the scope of this section, the underlying idea can be summarized quickly: combine multiple layers of (non-)linear operations and train all involved parameters from all layers jointly. This joint optimization of multiple processing stages gives rise to the prominent name *deep learning*. In consequence, deep networks are end-to-end learnable, *i.e.*, without the necessity of explicitly specifying parameters of intermediate layers.

Convolutional Neural Networks for Visual Object Recognition: A specific network type called *convolutional neural network (CNN)* recently gained popularity for various computer vision tasks. In these network types, stationary invariant filter masks are learned in several layers which are sensitive to patterns of varying complexity. The learned filters on the first layer are directly convolved with raw image pixels. Convolution results, which are also referred to as feature maps, are then non-linearly transformed and eventually pooled over spatial regions. Resulting maps finally serve as input for the subsequent layer. Thus, a typical building block in CNN architectures consists of convolutions, non-linear transformations, optional pooling, and feeding to the following stage. Depending on the specified architecture, this pattern is repeated until the final classification layer outputs classification scores.

Due to the hierarchy of multiple layers, CNNs can be seen as end-to-end learnable counterparts to the aforementioned bag-of-visual-words pipeline. Of major importance was the paper by Krizhevsky et al. (2012) which brought deep learning with CNNs into the focus of the computer vision community. A simplified version

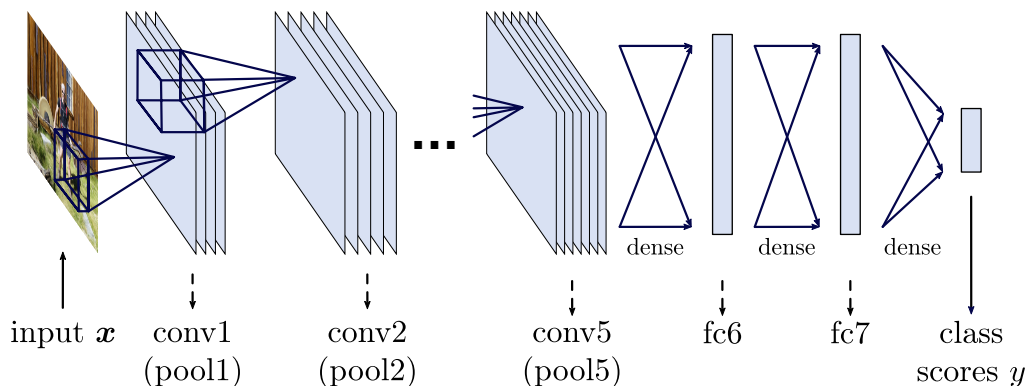


Figure 2.14: A simplified version of the network layout as introduced in (Krizhevsky et al., 2012). Filters in each layer operate on outputs of previous layers, results are non-linearly transformed, and optionally pooled over spatial regions. Besides running the entire network on input images, outputs of each layer can serve as representation with increasing level of complexity, leading to an extractable feature hierarchy. Figure is inspired by (Simon et al., 2014).

of their network’s layout is visualized in Fig. 2.14.

Why Learning Deep Networks Is Difficult: While the concept of deep learning sounds like the perfect solution for many computer vision problems, it also comes with severe drawbacks. First of all, massive amounts of data are required to prevent overfitting of models with millions of learnable parameters. As an example, the successful model by Krizhevsky et al. (2012) which won the ILSVRC 2012 challenge (Russakovsky et al., 2015) could only be trained successfully using millions of images from ImageNet (Deng et al., 2009) which even needed to be artificially augmented. Additionally, processing of these amounts of data heavily relies on powerful computation capacities *e.g.*, large clusters or hardware accelerators such as GPUs. Finally, the design of network layouts as well as setting of involved hyperparameters still requires expert knowledge (Deng and Yu, 2013) and is currently understood as “black art”.

Due to these drawbacks, training a CNN from scratch for a new task is currently hardly possible in small-scale and medium-scale projects common for university-based research (for possible ways out, the interested reader is referred to the inspiring work by Hoffman et al. (2015) and Tzeng et al. (2015)). We therefore skip a more detailed review on techniques for training these networks. A short review for learning and adapting CNN models is given in Section 6.1.2. Here, we follow (Donahue et al., 2014) and assume that a pre-trained network is given, *e.g.*, the one by Krizhevsky et al. (2012).

Applying CNNs to Computer Vision Tasks: Applying feature hierarchies from pre-trained CNN models for a variety of different tasks has shown promising results, *e.g.*, for visual object classification (Chatfield et al., 2014a; Donahue et al., 2014; Gong et al., 2014; Razavian et al., 2014), object detection (Chatfield et al., 2014a; He et al., 2014), scene recognition (Donahue et al., 2014; Gong et al., 2014; He et al., 2014; Razavian et al., 2014), fine-grained recognition (Branson et al., 2014b; Donahue et al., 2014; Razavian et al., 2014), or image retrieval (Babenko et al., 2014; Chatfield et al., 2014b; Freytag et al., 2015; Razavian et al., 2014).

Depending on the chosen layer, filters can be sensitive to simple structures in lower levels (*e.g.*, edges, corners, or junctions) or to semantically rich objects of almost arbitrary complexity (*e.g.*, human faces, cats, or dogs). An example for the visualization of learned filters in different levels can be found in (Zeiler and Fergus, 2014). Thus, besides running the entire network as classifier on new images, we can grep outputs of different layers to serve as representations with increasing level of complexity. By using outputs of multiple layers simultaneously, a given network can even serve as black-box mechanism for computing a feature hierarchy (Hariharan et al., 2015; Long et al., 2015).

2.2.4 Parts and Attributes

In contrast to image representations which are hand-designed (Section 2.2.1) or learned by optimizing a given criterion (Section 2.2.2 and Section 2.2.3), humans can describe images and objects most easily by naming semantically meaningful properties. Common human descriptions either rely on *attributes*, *e.g.*, color, shape, or texture, or are based on counting visible *parts*, *e.g.*, birds have two wings, elephants have a trunk, and so on. To meet this observation, image representations based on detected parts or attributes have been developed. In this section, we exemplarily review one prominent technique for detecting parts or predicting attributes in images, respectively. A more detailed review for techniques which learn corresponding predictors from data is given in Section 6.1.3.

Detecting Objects and Parts: One of the most common approaches for detecting objects and parts in 2D images is the *deformable parts model (DPM)* of Felzenszwalb et al. (2010) which we denote with M_{DPM} . As the name suggests, locations of a whole object and parts therein are modeled jointly. Parts are further allowed to deviate from fixed positions with adjustable deformation costs. Thus, a DPM consists of a set of filters $\mathcal{F} = \{w_R, w_1, \dots, w_{n_p}\}$, where the root filter w_R is intended to cover the whole object and the remaining filters the object's parts, respectively. Note that a filter is essentially a linear model as introduced in Section 2.1.4.3. The spatial configuration of parts and object are expressed as a deformation model s .

Let us assume a given DPM with a root filter w_R of fixed width and height. We can then compute the detection score for a sub-image centered at position $p = (x, y)^T$ and with width and height defined by w_R as follows:

$$f_M(p) = \max_{z \in \mathbb{N}^{2 \times n_p}} f_{w_R}(p) + \sum_{k=1}^{n_p} f_{w_k}(p + z_k) - \mathcal{L}_{\mathcal{D}}(z; s) . \quad (2.141)$$

We denoted with $Z = (z_1, \dots, z_{n_p})$ latent part locations such that $z_i = (x_i, y_i)^T$. Furthermore, $\mathcal{L}_{\mathcal{D}}$ is the cost⁸ of Z with respect to the learned deformation model s . For the sake of a clear notation, $f_{(\cdot)}(p)$ refers to evaluating a model (or filter) on the corresponding features extracted from the given image at position p .

Training of filters and deformation models is usually done with a latent SVM scheme, and a detailed description is given by Felzenszwalb et al. (2010). A learned model for persons together with the obtained detection result is shown in Fig. 2.15.

Attributes for Semantic Descriptions: Attributes in computer vision usually refer to “visual concepts that can be detected by machines, understood by humans, and shared across categories” (Duan et al., 2012), e.g., is-striped, has-dotted-neck, or has-legs. Humans use attributes in a variety of different ways. Attributes can be either correspond to entire objects (e.g., is-big) or specific for certain parts of an object (e.g., has-dotted-neck). Further, humans use attributes to describe the appearance of an object (e.g., is-striped), to describe quantities of parts (e.g., has-four-legs), and even for properties which are derived from different senses (e.g., is-fluffy) or from commonsense knowledge (e.g., eats-fish).

Driven by the previous observations, attributes can often be modeled as a binary classification problem stating whether or not a specific attribute is present in an image. Alternatively, continuous outputs can model the strength of a specific attribute, e.g., the estimated beauty of a person.

One of the first works on attributes as semantically rich descriptions was introduced by Lampert et al. (2009b). Each attribute is modeled as a Kernel-SVM as introduced in Eq. (2.73) and a chi-square kernel κ_{χ^2} (see Eq. (2.108)) is chosen to measure similarity. For obtaining probability estimates, Platt scaling on SVM outputs is applied (see Eq. (2.80)). As SVM inputs, different feature types are combined, e.g., color (RGB) and texture (HOG) features extracted on different image resolutions as well as BoW representations for multiple descriptor types including SIFT.

In consequence, attributes as high-level image descriptions can be interpreted as an additional layer within the image recognition pipeline which explicitly incorporates human interpretability. A visualization of attributes as image descriptions is

⁸ $\mathcal{L}_{\mathcal{D}}$ can be interpreted as a loss function (see Sect. 2.1.2.1) specifically suited for object detection.

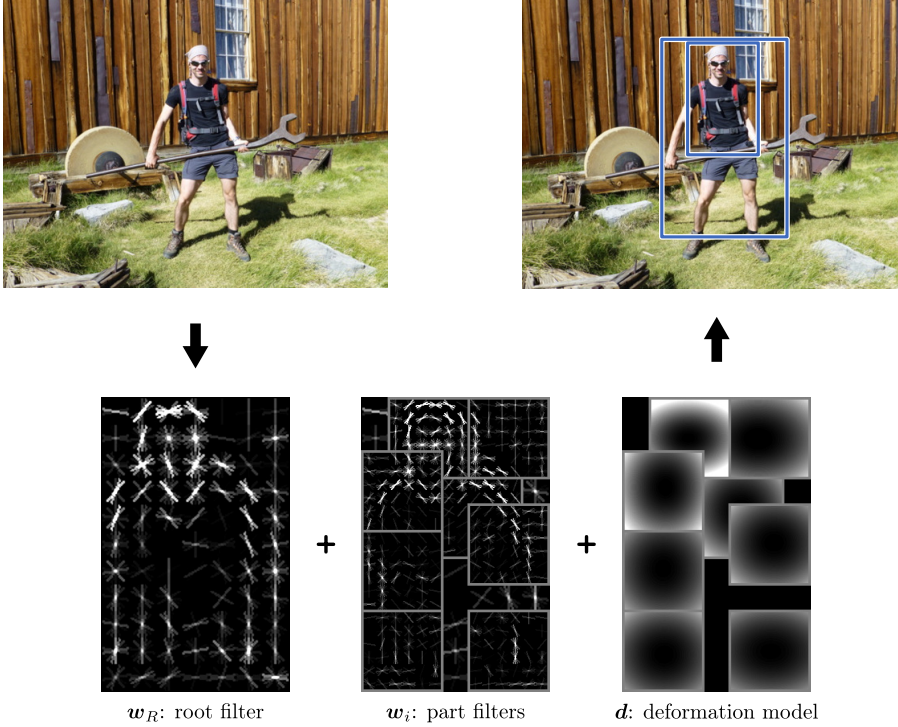


Figure 2.15: Detecting objects and parts with deformable part models. The position p for evaluating the root filter w_R and part filters w_k is optimized such that filter responses are maximized while part deviations z_k with respect to learned deformation model d regularize deformation costs. Results are obtained with code by Girshick et al. (2014b) as described in (Felzenszwalb et al., 2010).

given in Fig. 2.16. A more detailed review on state-of-the-art in learning of attribute representations is given in Section 6.1.

2.3 On Lifelong Learning in Computer Vision Scenarios

The Focus on Static Systems: In the previous sections, we were concerned with visual recognition in static, closed-world scenarios. To tackle this task, we required two main ingredients: (1) techniques for representing images and (2) techniques to learn the underlying relations between image representations and categories. Image representations were designed to reduce the vast dimensionality of raw image inputs to a low-dimensional subspace. Ideally, this subspace is as compact as possible while still containing the majority of information relevant for the current recognition task (Section 2.2). For a given condensed representations, we then

Person attributes

male:	yes
long hair:	no
mustache:	no
glasses:	yes
is happy:	yes

Scene attributes

outside:	yes
forest:	no
water:	no
man-made:	yes



Figure 2.16: *Attributes are high-level image descriptions which are semantically rich and therefore allow for human understanding.*

placed different assumptions on distributions of categories which finally lead to a variety of learning algorithms (Section 2.1).

For all these techniques, however, we followed one underlying assumption: if we were able (1) to collect enough training data and (2) to specify sufficiently complex models, then we could learn the unknown ground truth relation between images and labels without error. In terms of the previously introduced machine learning concepts, we thus assumed that we can precisely estimate the joint distribution $p(\mathbb{X} = \mathbf{x}, \mathbb{Y} = \mathbf{y})$ in Eq. (2.1), Section 2.1.2.1 with enough data and sufficiently complex models. However, this assumption is unlikely to hold for lifelong learning scenarios. As mentioned earlier, this is due to the fact that the joint distribution might change over time.

In previous years, our community spent major efforts on solving object classification and detection tasks in static environments by learning models from collected datasets of increasing size. Meanwhile, lifelong learning of similar systems has been outside of the main focus of research. We argue that tackling these issues will be among the most important challenges of our research field for the upcoming years.

Relevant Aspects for Lifelong Learning: In Section 1.2, we already introduced our concept for lifelong learning which consists of a closely connected interplay of several aspects. Among them is the necessity to analyze new incoming data, *e.g.*, by detecting novel objects or by filtering objects the current system is highly uncertain about. In addition, the amount of information which can be gained by asking questions about the content of individual images should be estimated. As a result, the best-ranked examples of previously analyzed data can be actively selected and handed to an expert-in-the-loop.

After receiving feedback, *e.g.*, as class labels or part annotations, the obtained information needs to be incorporated adequately. This requires efficient update routines for existing models (online learning). Furthermore, new categories might become known and corresponding models can be learned robustly by transferring knowledge from related categories that are already known. In addition, adaptation techniques are required to deal with changes in data distributions which occur over time. Finally, object representations need to be extended incrementally to compensate for the growing amount of known categories. A visualization of our understanding of the resulting lifelong learning cycle for visual recognition is given in Section 1.2, Fig. 1.3.

Given the previous aspects, we argue for the necessity of a holistic approach to lifelong learning, where each step is carefully modeled and results serve as inputs for the subsequent module. However, we are not aware of any holistic approach to lifelong learning which covers all fields. To the best of our knowledge, there is also no system reported so far which reliably implements all aspects. Nonetheless, a few related working areas emerged over time, which we will shortly introduce in the following.

2.3.1 “Lifelong Learning” Understood as Transfer Learning

The term lifelong learning has already been used two decades ago when referring to approaches which are today also known as *knowledge transfer*. Thrun (1995) was among the first who introduced *lifelong learning* in this context. In contrast to the scope of this thesis, Thrun understood lifelong learning as “*situations where a learner faces a stream of learning tasks*” (Thrun, 1995, p. 3). As presented later by Thrun (1996), it addresses the question “*Is Learning The N^{th} Thing Any Easier Than Learning The First?*”. Thus, the incoming stream of data is provided as-is, *i.e.*, without the possibility of actively influencing the data acquisition or selection. Regarding our terminology, our lifelong learning cycle is more general and subsumes the task of knowledge transfer as a highly relevant subtopic. With respect to our visualization of this cycle in Fig. 1.3, it covers the upper part.

A detailed overview of early approaches in this domain is given by Rodner (2011, Sect. 1.2). For three representative approaches, we point to the work of Fei-Fei (2006), Tommasi and Caputo (2009), and Eaton and Ruvolo (2013). More recently, Pentina and Lampert (2014) introduced error bounds for transferring knowledge from known categories to a previously unknown task. Their formulation is based on treating priors over transferable parameters as random variables under the assumption that all tasks (*e.g.*, one-vs-all separations) are sampled i.i.d. from a large set of possible tasks. Later on, Pentina et al. (2015) presented how the i.i.d. assumption can be overcome. Their approach is motivated by human education where easy concepts are learned first and harder concepts are followed later by

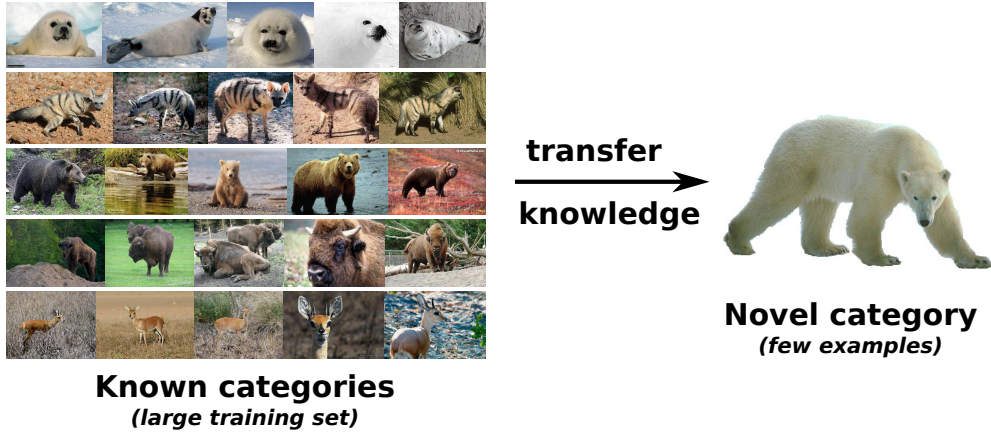


Figure 2.17: Motivation for transfer learning which is also known as lifelong learning according to Thrun (1995). Transfer learning tackles the question “Is Learning The N^{th} Thing Any Easier Than Learning The First?”. Accordingly, models or estimated parameters should be transferred from previously known categories (left) to novel categories in incoming data (right). Thereby, models for novel categories can be learned adequately even with few training examples. The figure is inspired by (Rodner, 2011, Sect. 1.2).

relating them to simpler aspects. The authors derive a strategy to optimize the order of category learning such that benefit of parameter transfer is maximized. For each new category, an adaptive soft-margin SVM is trained such that weight vectors are transferred from the previously learned SVM. It should be noted that their approach assumes that all categories and all data are known in advance.

In summary, a notable amount of research uses the term lifelong learning when referring to learning categories over time by efficiently sharing knowledge between subsequent tasks. In contrast to the current thesis, there is usually no active step involved to decide which data to select for requesting expert feedback.

2.3.2 Open-space Recognition

The previously reviewed concept of transfer learning tackled the problem of transferring knowledge to recently observed categories which allows training models more reliably. An additional line of research investigates the questions which models are sensible to be applied in lifelong learning scenarios. To this end, let us recall that we do usually not know the set of all possible classes in advance, *i.e.*, $\mathcal{Y}$ is not fixed. Furthermore, our initially collected training set is hardly representative, *i.e.*, Ω is not sufficiently covered by collected data. In consequence, it is hardly justifiable to apply classification techniques which assign any input vector x to one out of M currently known categories. We already noted in Section 1.2 that

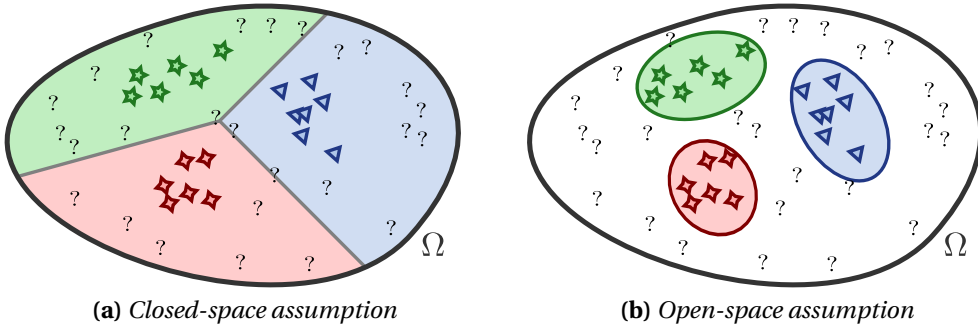


Figure 2.18: Illustration of closed-world and open-space recognition. Left: Models are trained from collected data (red, blue, and green stars) to minimize empirical errors. Novel examples (indicated as question marks) are treated without considering that they can stem from previously unknown categories. This leads to decision areas which cover the entire input space Ω . Right: Trained models still minimize empirical errors but additionally reduce confusion with yet unknown categories. Novel examples are only classified if contained in estimated classification areas. Otherwise, they are marked for further treatment or simply rejected.

this would correspond to a closed-world assumption. Instead, we are confronted with open-space scenarios in lifelong learning. Thus, we would intuitively require a combination of discriminative aspects (to separate known parts of space) and generative aspects (to specify which regions are trustworthy for separation). An illustrative example is shown in Fig. 2.18.

Several approaches have been introduced for transferring established machine learning techniques to open-space recognition scenarios. Scheirer et al. (2013) extended the concept of empirical risk minimization (see Eq. (2.9)) towards open-space scenarios. The authors introduced a second term which they call *open set risk* and which accounts for “overgeneralization”, *i.e.*, it penalizes models which classify large portions of input space Ω as belonging to known categories. They further restricted their analysis to binary classification scenarios and derive a variant of SVM models which jointly optimizes empirical risk and open set risk. Similarly in spirit are classification models with a build-in reject options, *e.g.*, the SVM extension presented by Bartlett and Wegkamp (2008) (see also cited references for further approaches). Later on, (Bendale and Boulton, 2015b) extended their previous work to handle multi-class scenarios where new categories are incrementally added over time. The resulting approach can be seen as two-stage pipeline where classification decisions are returned if a novel example is not rejected by a novelty detection system. Based on that, the authors further derived techniques to train convolutional neural networks which are suitable for classifying known examples and rejection unknown data (Bendale and Boulton, 2015a). In a different approach,



sunglasses is used by **person**

shoe is used in **outdoor**

window is part of **wall**

grounding wheel is a **tool**

...

Figure 2.19: Commonsense knowledge expresses relations between categories.

Wah and Belongie (2013) showed how to classify known objects and identify abnormal examples using the presence or absence of attributes such as “striped” or “is furry”. Jain et al. (2014) proposed a calibration technique for SVM models to reject examples from low-density regions in space.

Finally, it is not only important to train models which do not overgeneralize, but also to actively search for informative examples to increase knowledge over time. To tackle this goal, Hospedales et al. (2012) introduced a framework to discover new categories and simultaneously search for informative, unlabeled examples to be annotated by human experts. Similar in spirit is our technique presented in Section 5.6.3 which additionally allows for potentially noisy data. Except for this aspect, we do not focus on novelty detection problems in this thesis. For a detailed analysis of reject options, novelty detection, and open-space techniques, we instead refer to (Bodesheim, 2016).

2.3.3 Continuous Learning of Commonsense Knowledge

The previously introduced research areas required incoming data streams. For providing data, we can expect that the Internet is today’s largest source for such information. Especially with the availability of Internet search engines and sharing platforms, the steadily increasing amount of textual and visual data can be accessed and exploited. When we focus on news and social platforms, the contained data implicitly provides a variety of facts from our every-day life. These facts – such as “wheel is part of car” – are also known as *commonsense knowledge*. Among others, commonsense knowledge provides statements about different concepts (wheel and car) which are put into relation (is part of). A visualization of commonsense knowledge is shown in Fig. 2.19.

For several years, researchers aim at designing systems and algorithms which

are able to extract commonsense knowledge by continuously mining Internet data. An early approach was presented by Carlson et al. (2010), who proposed the Never-Ending Language Learner (NELL) which was reformulated later in (Mitchell et al., 2015). NELL consists of several subroutines which learn extract different relations between words from Internet text search results. In every iteration, NELL's components compute sets of possible relations (candidate facts), which are then integrated as beliefs into NELL's knowledge base if their confidence is high enough. In subsequent iterations, the grown knowledge base is used by the subroutines to propose new candidate facts.

Inspired by NELL, Chen et al. (2013) introduced the Never-Ending Image Learner (NEIL). Based on provided image tags, NEIL initially trains SVM classifiers for individual categories in a semi-supervised setting. Later on, NEIL continuously accesses Google image search results and uses highly scored image regions to update previously learned models without additional manual supervision. Outputs of detection models are then used to extract two types of commonsense knowledge: "concept A is part of concept B" and "concept X is similar to concept Y". Going one step further, Sadeghi et al. (2015) recently presented the Visual Knowledge Extraction System (VisKE). Their approach goes beyond NEIL and allows for discovering more general relations, *e.g.*, "drinking" or "hunting".

In contrast to the previous approaches, we do not focus on extracting commonsense knowledge from Internet data. Instead, we focus on image categorization tasks in this thesis. Furthermore, we put a strong focus on interactive learning by actively selecting highly informative images for manual verification (see Chapter 5 for details).

3 | Large-scale Learning with Gaussian Process Models

“How to deal with tens of thousands of examples in a Bayesian manner?”

In the previous chapter, we reviewed techniques for representing images (Section 2.2) and for estimating relations between these representations and target values from collected data (Section 2.1). For a given problem, we can now choose appropriate representations and models according to desired properties. However, we commonly observe further limitations due to available memory and computation time. Especially for increasing amounts of data – both with respect to number of examples and dimensionality of feature dimensions – these aspects gain further importance.

Especially non-parametric methods substantially suffer from large datasets since all training examples are retained even after the training stage. Besides the simple nearest neighbor model (see Section 2.1.4.1), this also holds for the powerful concepts of kernel-based methods as introduced in Section 2.1.5, *e.g.*, Kernel-SVM or GP models. In lifelong learning scenarios, though, available data is likely to become large. Furthermore, representations likely need to increase in size over time to distinguish previously unknown categories. In this chapter, we will therefore present techniques for Gaussian process models which significantly reduce the computational burden by orders of magnitude – both with respect to required computation time as well as allocated memory. A key ingredient will be the choice of generalized histogram intersection kernels (see Section 2.1.5.4 for a short introduction).

Contributions of This Chapter: We will start with an overview on related work in large-scale learning scenarios in Section 3.1. We then present in Section 3.2 how GP label regression models can be trained and evaluated efficiently. In addition, we derive techniques for optimizing hyperparameters in Section 3.3. Furthermore, we

show in Section 3.4 how the predictive variance can be efficiently computed. Finally, we shortly introduce the task of semantic segmentation as a potential application in Section 3.5. The main results presented in this chapter have been published in (Rodner et al., 2012), (Freytag et al., 2012b), and (Freytag et al., 2012a) as well as in the summarizing journal article (Rodner et al., 2015a).

3.1 Related Work on Large-scale Learning

Until the end of the last decade, algorithms in machine learning and computer vision have usually been evaluated on standard benchmark datasets with several hundreds or thousands of examples. Well-known datasets include Caltech-101 (Fei-Fei et al., 2004) and Caltech-256 (Griffin et al., 2007) for the task of object recognition, MSRC for semantic segmentation (John Winn, 2004), PASCAL for object detection (Everingham et al., 2010), or 13 natural scenes (Fei-Fei and Perona, 2005) and 15 natural scenes (Lazebnik et al., 2006) for scene recognition. Based on increasing computation power, storage abilities, and cheap image annotation tools such as Amazon Mechanical Turk, however, we noticed a significant shift over recent years towards large datasets (Deng et al., 2009; Lin et al., 2014; Quattoni and Torralba, 2009; Torralba et al., 2008) and models handling the growing amount of data (Russakovsky et al., 2015). As an example, one of the most important milestones was the release of ImageNet – a large-scale dataset by Deng et al. (2009) – which contained in its initial version already more than 3 million images of more than 5,000 categories. In this section, we briefly review a collection of relevant work from the field of large-scale visual recognition with a special focus on solutions for Kernel-SVMs and Gaussian process models.

3.1.1 Large-scale Learning with Various Models

The straight-forward solution for reducing training time with increasing amounts of data is to choose models which do not involve any training step at all, *e.g.*, the nearest neighbor classification technique introduced in Section 2.1.4.1. For an impressive application of model-free classification with millions of known images, we refer to the work by Torralba et al. (2008). An easily scalable variant is the nearest class mean technique by Mensink et al. (2013) which collapses all examples of a category into a single representative and incorporates metric learning to compensate the information loss.

Changing the scale of datasets additionally raises the question whether the relative ordering of classification techniques is stable with respect to classification accuracy. While learning theory can provide bounds on estimated errors for specific dataset sizes and models (see Section 2.1.2), resulting bounds are usu-

ally hard to compute exactly and mainly concerned with worst-case statements. Therefore, Deng et al. (2010) experimentally analyzed the justifiability of transferring experimental findings from small to large datasets. According to their results, even simple methods can perform surprisingly well for large-scale scenarios, *e.g.*, a nearest neighbor classifier obtained a classification accuracy slightly superior to a linear SVM. Besides, efficiency of computations heavily matters for successful applications. Due to the existence of highly optimized solvers, linear SVMs (see Section 2.1.4.3) and logistic regression as linear models are perfectly suited for being applied in large-scale scenarios (Fan et al., 2008; Perronnin et al., 2012).

A hybrid classifier combining decision trees and SVMs was proposed by (Chang et al., 2010). Their technique learns a decision tree for a supervised hierarchical division of the feature space, and examples of each cell are modeled by an SVM. Classifying new examples is then done by traversing the tree and applying the learned model of the corresponding leaf node. Similarly, Ristin et al. (2014) combine random decision forests and class mean classifiers for large-scale object recognition.

Finally, we want to emphasize that recent successful results of deep learning with convolutional neural networks (see Section 2.2.3) would not have been possible without the existence of large-scale datasets. Since representations and classification models are trained jointly with millions of parameters involved, large amounts of data are required to prevent overfitting. Nonetheless, given enough data and sufficient computation power, resulting application scenarios are manifold, *e.g.*, visual object categorization (Krizhevsky et al., 2012; Sermanet et al., 2014; Szegedy et al., 2014), object detection (Girshick et al., 2014a; Hariharan et al., 2014; Sermanet et al., 2014), fine-grained recognition (Zhang et al., 2014), and semantic segmentation (Brust et al., 2015; Hariharan et al., 2014; Long et al., 2015).

3.1.2 Large-scale Learning with Kernel SVMs

In Section 2.1.5.4, we noticed in Theorem 2.7 that evaluating a kernel $\kappa(\cdot, \cdot)$ on input examples x and x' equals an inner product $\langle \Phi_\kappa(x), \Phi_\kappa(x') \rangle_{\mathcal{H}_\kappa}$ in the corresponding kernel space $\mathcal{H}_\kappa$. For SVM models, the utilization of kernels was directly possible when solving the dual problem given in Eq. (2.73). However, we noticed that solving the (kernelized) dual problem scales with time $\mathcal{O}(N^3)$. In addition, kernel evaluations of any pair of inputs need to be stored, resulting in an asymptotic storage demand of $\mathcal{O}(DN + N^2)$. In consequence, exactly solving the dual optimization problem can be computationally unfeasible for large-scale datasets. For the specific choice of histogram intersection kernels, Maji and Berg (2009) therefore proposed to explicitly compute feature transformations $\tilde{\Phi}_{\kappa_{\text{HIK}}}$ which approximate the exact implicit transformations $\Phi_{\kappa_{\text{HIK}}}$ induced by histogram intersection kernels as introduced in Eq. (2.109). Later on, a general transformation for arbitrary ad-

ditive kernels (see Section 2.1.5.4) was presented by Vedaldi and Zisserman (2010, 2012). After applying the transformation, the explicitly embedded input features can be used for training a linear SVM.

Instead of approximating the feature transformation, one can alternatively approximate the process of model training, *e.g.*, using stochastic gradient descent (see (Shalev-Shwartz et al., 2011) for a compact review). Wang et al. (2012) used κ_{HIK} to derive an efficient version of the stochastic gradient descent technique.

If the SVM dual should be solvable in its kernelized version, computing exact classification decisions afterwards can be speeded up from $\mathcal{O}(DN)$ to $\mathcal{O}(D \log N)$ without any required approximation but just by using computational benefits arising from κ_{HIK} as shown by Maji et al. (2008, 2012). Since their approach still requires expensive training, Wu (2010, 2012) extended the previous results and presented an efficient technique for also solving the dual formulation for κ_{HIK} based on coordinate descent (see (Hsieh et al., 2008) for an informative introduction). On retrospective, our developed techniques for fast exact Gaussian process models equipped with histogram intersection kernels as presented in this chapter where mainly influenced by the work of Maji et al. (2008) and Wu (2010).

3.1.3 Large-scale Learning with Gaussian Process Models

As reviewed in Section 2.1.5.9, two main drawbacks arise from the choice of a Gaussian process model: (1) the asymptotic computation time in training which requires $\mathcal{O}(N^3)$ for inverting or decomposing the kernel matrix, and (2) a memory consumption of $\mathcal{O}(N^2)$ for storing the entire kernel matrix. A naive solution is to neglect all but a small set of $n \ll N$ training examples. According to Rasmussen and Williams (2006, Sect. 8.3.3), this approach is known as “subset of data”.

Different strategies for carefully selecting subsets have been proposed. Two well-known examples are the informative vector machine by Lawrence et al. (2003) or the information theoretic approach by Csató and Oppé (2001). In all techniques, special care needs to be taken to retain the information which would be lost by simply dropping unchosen examples.

While the amount of stored data can thereby be decreased, the set of selected examples does not necessarily be restricted to a subset of the training set itself. In fact, Quiñero Candela and Rasmussen (2005) showed that the majority of sparse GP techniques can be viewed as determining a set of latent variables and assuming that outputs of training and test examples are conditionally independent given these variables. From this perspective, the set of latent variables is often called *inducing variables*. Special cases of sparse GP techniques are the subset of regressors approximation by Smola and Bartlett (2001), the deterministic training conditional approximation by Seeger et al. (2003), or the fully independent training conditional (FITC) approximation by Snelson and Ghahramani (2006).

Sparse GP approximations can roughly be interpreted as approximating a GP model by reducing the number of computationally relevant examples. An orthogonal approach consists of retaining the entire training data, but approximating the model's parameters instead. In our work in (Bodesheim et al., 2013b) and (Bodesheim et al., 2013a), we presented an approximation technique which replaces the full covariance matrix with a diagonal matrix and thereby significantly reduces memory demand and computation times for training and testing. The interested reader is referred to Bodesheim (2016) for a detailed analysis.

Another line of research tackles computational drawbacks of plain GP models by introducing hybrid models. One example is the combination of nearest neighbor selection and GP models as presented by Urtasun and Darrell (2008). Their approach selects a subset of training examples most similar to an unseen input for model approximation at test time. Thereby, exemplar-specific GP models are trained (see also Section 6.3 for exemplar-specific techniques). Thus, their technique can be interpreted as an online variant of sparse GP models. However, an approximation is computed not only once during training but instead on-the-fly for unseen examples.

Finally, note that a supervised division of the training set into smaller subproblems can be obtained by exploiting a random decision forest. For each sub-task consisting of all examples in a leave node, Fröhlich et al. (2012b) suggested training a full GP model which is again computationally feasible and thus related to the SVM counterpart as presented by Chang et al. (2010).

Besides the model approximations reviewed so far, solving $\alpha = \mathbf{K}^{-1}\mathbf{y}$ during training can also be cast as optimization problem by minimizing $\|\mathbf{K}\alpha - \mathbf{y}\|$ using *iterative linear solvers (ILSs)*, e.g., conjugate gradients (Golub and Loan, 1996, Sect. 10.2) or coordinate descent. A successful application requires efficient *matrix vector multiplications* to compute $\mathbf{K}\alpha$. Therefore, Shen et al. (2006) introduced a fast approximation of $\mathbf{K}\alpha$ by exploiting KD-trees. Their approach subdivides the input space and assigns equal kernel values to all examples within each cell. An alternative technique for approximating evaluations of RBF kernels with Taylor expansions was presented by Yang et al. (2005). The computational stability of learning with fast matrix vector multiplications is analyzed by de Freitas et al. (2006). Bo and Sminchisescu (2008) introduced a variant of the block coordinate descend method which greedily selects the subset of variables to be optimized in each iteration. Thereby, the optimization problem is decomposed into smaller problems and therefore does not require storing the full kernel matrix in memory. Our approach for exact large-scale Gaussian process models which we present in the following sections also relies on fast matrix vector multiplications. As we will see, selecting κ_{HIK} or generalizations thereof as covariance functions results in efficient techniques for *exactly* computing $\mathbf{K}\alpha$.

For further introductions and details on large-scale Gaussian process models, we refer the interested reader to (Rasmussen and Williams, 2006, Chapter 8), (Kemmler, 2013, Chapter 4), and (Quiñonero Candela et al., 2007).

3.2 Towards Exact and Scalable GP Models

In this section, we will see how to significantly reduce required resources during training and classification for Gaussian process models with histogram intersection kernels as covariance functions. Consequently, the resulting techniques allow for the application of GP models in large-scale image classification scenarios with tens of thousands of known examples. In these scenarios, the application of plain GP models is not possible anymore due to limitations of memory or computation time. This section is based on our findings published in (Rodner et al., 2012) and (Rodner et al., 2015a).

3.2.1 Histograms and Histogram Kernels

Introducing Histograms: Computer vision applications often require image representations which are based on statistics of contained components in images. A prominent example are Bag-of-visual-words algorithms introduced in Section 2.2.2. There, each histogram bin contains the number of local features assigned to the corresponding prototype. A second example are object detection algorithms, *e.g.*, DPMs as introduced in Section 2.2.4. Here, images are represented by statistics over quantized responses of edge filters, *e.g.*, HOG representations (see Section 2.2.1) which express the frequency of orientated gradients accumulated over subwindows.

According to Oxford’s dictionary (Oxford Dictionaries, 2015), a *histogram* refers to “*a diagram consisting of rectangles whose area is proportional to the frequency of a variable [. . .]*” and is thus a visualization tool in statistics. In this thesis, however, we follow the computer vision notion of histogram features and neglect the visualization aspect. Thus, we instead refer to a histogram as a real-valued vector $\mathbf{d} \in \mathbb{R}_*^D$ with non-negative components. If only relative frequencies are of interest, it is further common to normalize histograms to length one which thereby reduces the number of independent components to $D - 1$. Note further that histograms can also be used to express statistics of absolute occurrences which results in integer-valued components.

Kernels for Histogram Features: To measure similarity between histograms, a suitable distance function should reflect the nature of these representations. In computer vision applications, additive kernels as introduced in Section 2.1.5.4 have been found useful for this task. The presumably most prominent example is the

histogram intersection kernel κ_{HIK} as defined in Eq. (2.109) which compares two histograms by summing smaller values over each dimension. For integer-valued histograms, Barla et al. (2002) have been able to explicitly construct the corresponding feature map. Due to the converse of Mercer's theorem (see Theorem 2.7), they thereby proofed the positive definiteness of κ_{HIK} . Later on, the positive definiteness of κ_{HIK} for real-valued non-negative components was proven by Boughorbel et al. (2005). It should be noted that Maji and Berg (2009) stated a different proof for the conditional positive definiteness of κ_{HIK} which was finally completed by Wu (2010) for the positive definiteness. In consequence, histogram intersection kernels are applicable as covariance functions if image representations offer histogram characteristics.

Generalizations of HIK With Adjustable Hyperparameters: The previously introduced κ_{HIK} operates directly on inputs and can not be adjusted to training data. To allow for improved flexibility, Boughorbel et al. (2005) introduced a generalization of κ_{HIK} by applying arbitrary non-negative functions $g(\cdot)$ on inputs:

$$\kappa_{\text{GHIK}}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^D \min(g(\mathbf{x}(d)), g(\mathbf{x}'(d))) . \quad (3.1)$$

Although they restricted their application to absolute polynomial feature transformations:

$$g_{|\cdot|, \eta}(\mathbf{x}(d)) = |\mathbf{x}(d)|^\eta, \quad \eta \in \mathbb{R}_+, \quad (3.2)$$

the given proof is actually more general. As we will see later, it is beneficial if the applied transformation $g(\cdot)$ is an automorphism, *i.e.*, an order-preserving transformation. Thereby, the relative order of examples within each dimension is not affected by applying $g(\cdot)$, and we will later make use of ordered sequences of training values. As we noted in (Rodner et al., 2012), it follows from the proof given in (Boughorbel et al., 2005) that applying any positive-valued automorphism within κ_{HIK} still leads to a permissible kernel function. In consequence, we followed Maji et al. (2008) and introduced the notion of very general histogram intersection kernels in our work in (Rodner et al., 2012). Besides the powered absolute value in Eq. (3.2), another intuitive choice is exponential transformation:

$$g_{\exp, \eta}(\mathbf{x}(d)) = \frac{\exp(\eta |\mathbf{x}(d)|) - 1}{\exp(\eta) - 1}, \quad \eta \in \mathbb{R}_+. \quad (3.3)$$

We quantitatively evaluate the resulting improvements with respect to classification accuracy obtained by optimizing hyperparameters of generalized histogram intersection kernels in Section 7.2.3.

It should be noted that applying non-negative functions $g(\cdot)$ also includes trans-

formations individually adjustable for each dimension as mentioned in Eq. (2.125):

$$g_{\text{ard},\boldsymbol{\eta}}(\mathbf{x}(d)) = \boldsymbol{\eta}(d) \cdot \mathbf{x}(d), \quad \boldsymbol{\eta} \in \mathbb{R}_*^D. \quad (3.4)$$

Thus, it opens the door for relevance determination techniques to estimate the relative importance of each dimension. An possible application was given by Ablavsky and Sclaroff (2011) where weights were found by optimizing bounds or proxies of leave-one-out estimates to then train an SVM. As probabilistic counterpart, we will show later how to elegantly determine suitable weights by maximizing the marginal likelihood instead. In the remainder of this thesis, we will denote the generalizations of κ_{HIK} resulting from applying the previously introduced transformations $g_{|\cdot|,\beta}$, $g_{\text{exp},\beta}$, and $g_{\text{ard},\omega}$ as $\kappa_{\text{HIK-POLY}}$, $\kappa_{\text{HIK-EXP}}$, and $\kappa_{\text{HIK-ARD}}$, respectively.

We finally adhere that related approaches from the field of image-based kernel functions can also be viewed as generalization of κ_{HIK} , e.g., the spatial pyramid match kernel by Grauman and Darrell (2007), which additionally incorporate spatial information into the construction of the kernel function. However, a more intuitive interpretation is to view these approaches as improved representations which are coupled with a simple κ_{HIK} .

Fast Multiplication Kernels: In the remaining parts of this chapter, we are interested in breaking common computational limitations of representer models. Let us therefore introduce the notion of *fast multiplication kernels (FMKs)*, for referring to any kernel function with the following two properties:

1. computing $\mathbf{k}_*^T \boldsymbol{\alpha}$ scales only sub-linear with the number of known examples for any unseen test input $\mathbf{x}_*$ and
2. computing $\mathbf{K} \mathbf{v}$ requires only sub-quadratic time for any vector $\mathbf{v}$.

If a kernel function fulfills the first requirement of fast weighted kernel evaluations, its application in any representer model will directly result in enormous speed-ups during classification. Similarly, the second property of fast matrix vector multiplications will have dramatic impact on the efficiency of training these models. Since FMKs are only a theoretical construct so far, it remains to find kernel functions which actually exhibit both properties. As we will see in the following sections, several additive kernels – and especially generalizations of κ_{HIK} as introduced previously – indeed fulfill both aspects. Thus, they open the door for applying representer models such as GP also in scenarios where the sheer number of known examples would render the plain formulation as practically infeasible.

3.2.2 Efficient Computations of Classification Decisions

Following the previous two properties of FMKs, we first show that the (generalized) histogram intersection kernel allows for weighted kernel evaluations in sub-linear time. We will start with the basic formulation of the histogram intersection kernel, although all derivations also hold for its generalized counterpart. We will re-use the presented techniques in Section 3.2.3 to also obtain efficient routines for model training.

3.2.2.1 Exact Computations of Classification Decisions with (G)HIKs

For the choice of κ_{HIK} in Kernel-SVM models, Maji et al. (2008) presented significant speed-ups for the computation of classification scores. The key insight of their approach is to decompose the necessary computations once the trained model with its weight vector α is given. In this section, we will transfer their idea to the predictive mean estimate of GP models.

Decomposing Classification Scores: In Section 2.1.5.5, we already noted that due to the representer theorem, classification scores of SVM models and GP models can both be cast as kernel evaluations which are centered on training examples:

$$\mathbf{k}_*^T \alpha = \sum_{i=1}^N \alpha(i) \kappa(\mathbf{x}_i, \mathbf{x}_*) . \quad (3.5)$$

Thus, although Maji et al. (2008) solely focused the observations in Eq. (3.9) on SVM models, they also hold for computing the predictive mean within Gaussian process regression as introduced in Eq. (2.110). Furthermore, let us remind that the additivity of κ_{HIK} leads to evaluating the minimum operation in each dimension separately:

$$\mathbf{k}_*^T \alpha = \sum_{i=1}^N \alpha(i) \sum_{d=1}^D \min(\mathbf{x}_i(d), \mathbf{x}_*(d)) \quad (3.6)$$

$$= \sum_{d=1}^D \sum_{i=1}^N \alpha(i) \min(\mathbf{x}_i(d), \mathbf{x}_*(d)) . \quad (3.7)$$

Since both sums are linear operations, switching their order to obtain Eq. (3.7) is well justifiable.

When we inspect Eq. (3.7), we note that all training examples with values larger than $\mathbf{x}_*$ in dimension d will lead to identical contributions to the final classification

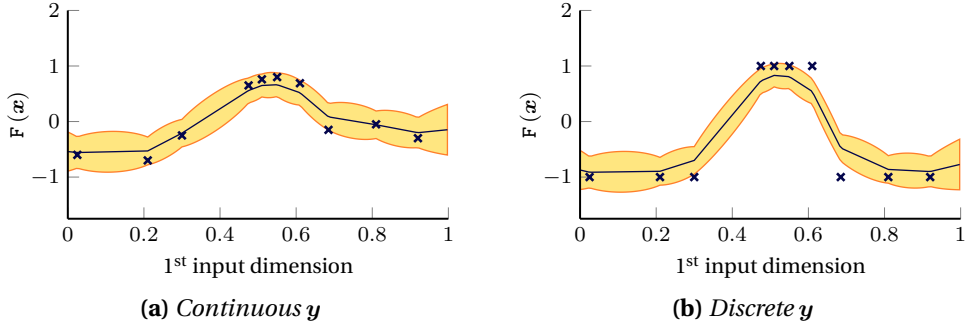


Figure 3.1: Visualization of the piecewise linearity of predictive mean estimates for GP regression models equipped with κ_{HIK} . Left: a regression scenario with continuous targets. Right: a classification task with discrete labels. Note that the underlying dimensionality is 2, however, we only show the first dimension due to the redundancy added by histogram normalization $\|\mathbf{x}\|_1 = 1$. Thus, we only display predictive mean estimates (blue line) and confidence areas (in orange) obtained from the standard deviation, i.e., the square root of the predictive variance, with respect to the first dimension. Training examples are denoted with blue crosses. Results are obtained by assuming a noise level of $\sigma_n^2 = 0.1$.

score. More precisely, classification scores are entirely defined by $\mathbf{x}_*(d)$:

$$\mathbf{k}_*^T \boldsymbol{\alpha} = \sum_{i=1}^N \alpha(i) \sum_{d=1}^D \min(\mathbf{x}_i(d), \mathbf{x}_*(d)) \quad (3.8)$$

$$= \sum_{d=1}^D \left(\sum_{\{i : \mathbf{x}_i(d) < \mathbf{x}_*(d)\}} \alpha(i) \mathbf{x}_i(d) + \mathbf{x}_*(d) \sum_{\{j : \mathbf{x}_j(d) \geq \mathbf{x}_*(d)\}} \alpha(j) \right). \quad (3.9)$$

It is interesting to note that the resulting decision function is then piecewise linear. A visualization is given in Fig. 3.1.

How to Partially Precompute Classification Scores: The decomposition property of kernel evaluations given in Eq. (3.9) will be the building block for efficient training, optimization of hyperparameters, and testing routines. Let us therefore assume that we computed permutations π_d specific for each dimension which re-arrange training examples in ascending order with respect to input dimension d .

Given π_d , we can rewrite the previous decomposition as follows:

$$\mathbf{k}_*^T \boldsymbol{\alpha} = \sum_{d=1}^D \left(\underbrace{\sum_{i=1}^{r_d} \alpha(\pi_d^{-1}(i)) \mathbf{x}_{\pi_d^{-1}(i)}(d)}_{\dot{=} \mathbf{A}(d, r_d)} + \mathbf{x}_*(d) \underbrace{\sum_{i=r_d+1}^N \alpha(\pi_d^{-1}(i))}_{\dot{=} \mathbf{B}(d, r_d)} \right) \quad (3.10)$$

$$= \sum_{d=1}^D (\mathbf{A}(d, r_d) + \mathbf{x}_*(d) \cdot \mathbf{B}(d, r_d)) \quad (3.11)$$

where r_d is the number of examples with entries smaller than $\mathbf{x}_*$ in dimension d . If we already obtained $\boldsymbol{\alpha}$, we can also precompute matrices $\mathbf{A}$ and $\mathbf{B}$ during training. For an unseen test example, we then only have to find r_d for each dimension which is possible with a simple binary search in $\mathcal{O}(\log N)$ since training examples are sorted via permutations π_d . Thus, Eq. (3.11) allows us to evaluate the inner product $\mathbf{k}_*^T \boldsymbol{\alpha}$ in $\mathcal{O}(D \log N)$.

The previous derivations also hold for the generalization of κ_{HIK} as introduced in Eq. (3.1) if the transformation function $g(\cdot)$ is an automorphism, since permutations π_d remain unchanged and matrices $\mathbf{A}$ and $\mathbf{B}$ can be computed accordingly. From a practical point of view, we further note that for non-negative valued features, we can nicely exploit sparsity in Eq. (3.11), *i.e.*, we can skip input dimensions with $\mathbf{x}_*(d) = 0$. This aspect is especially relevant for extremely high-dimensional representations.

In summary of the previous derivations, we conclude that κ_{HIK} and κ_{GHIK} fulfill the first property of FMKs. We now show how to approximate classification decisions to further reduce the number of required computations.

3.2.2.2 Approximating Classification Decisions with (G)HIKs – the Quantization Idea

In Eq. (3.11), we noticed that unseen inputs can only result in $N + 1$ different positions relative to training examples in each dimension, which allowed for precomputing matrices $\mathbf{A}$ and $\mathbf{B}$. In (Maji et al., 2008), the authors proposed to further quantize test inputs in each dimension. Let us therefore assume that feature values of dimension d are bounded within $[l_d, u_d]$. This assumption trivially holds for normalized histograms and we can think of $[l_d, u_d]$ as $[0, 1]$ in the following derivations.

Quantizing Features using Equidistant Interval Borders: The simplest quantization of the specified interval into q smaller intervals is an equidistant placement

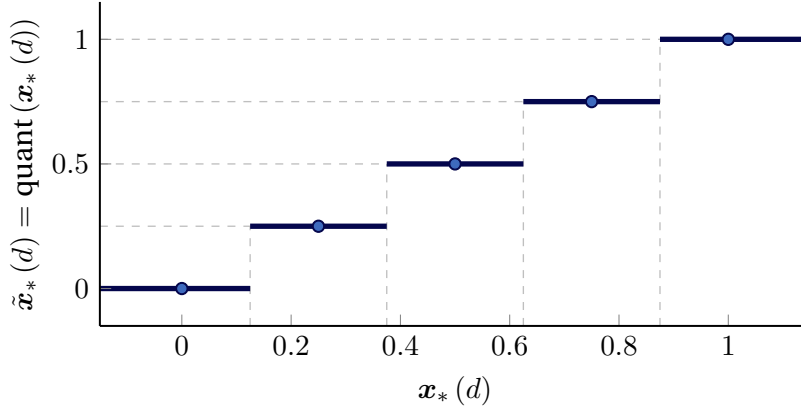


Figure 3.2: Quantization of inputs with equidistant placement of interval borders as defined in Eq. (3.12) can be computed in $\mathcal{O}(1)$ operations by mapping the evaluated step function to the desired interval as depicted in Eq. (3.13). We visualized the resulting step function computed with $q = 5$ bins and interval borders of $l_d = 0.0$ and $u_d = 1.0$, respectively.

of interval borders and centers:

$$\text{binning}(x_*(d)) = \begin{cases} 1 & \text{if } x_*(d) \leq l_d \\ \left\lfloor \frac{(x_*(d) - l_d)}{u_d - l_d} \cdot q + \frac{1}{2} \right\rfloor & \text{if } l_d < x_*(d) < u_d \\ q & \text{if } u_d \leq x_*(d) \end{cases}, \quad (3.12)$$

where $\lfloor x \rfloor$ denotes the floor operation, *i.e.*, rounding to the largest integer smaller than x . Thus, $\lfloor x + \frac{1}{2} \rfloor$ is a computational convenient way for rounding x . Quantizing a new value can then be done in constant time by evaluating the step function and mapping results back to the desired interval:

$$\text{quant}(x_*(d)) = \frac{\text{binning}(x_*(d)) - 1}{q - 1} (u_d - l_d) + l_d. \quad (3.13)$$

A visualization for the quantization defined by Eq. (3.13) is given in Fig. 3.2.

Quantization Allows for Extended Pre-Computations: Due to the finite number of possible values after quantization, kernel evaluations in Eq. (3.11) can also lead to only a finite number of different values. In consequence, we can additionally precompute all possible values and store results in a single *look-up table (LUT)* denoted with $T \in \mathbb{R}^{D \times q}$. For unseen test inputs, we then only need to quantize the value of each dimension, index T accordingly, and add the obtained D entries. Thereby, we can reduce the asymptotic computation time to $\mathcal{O}(D)$ if the

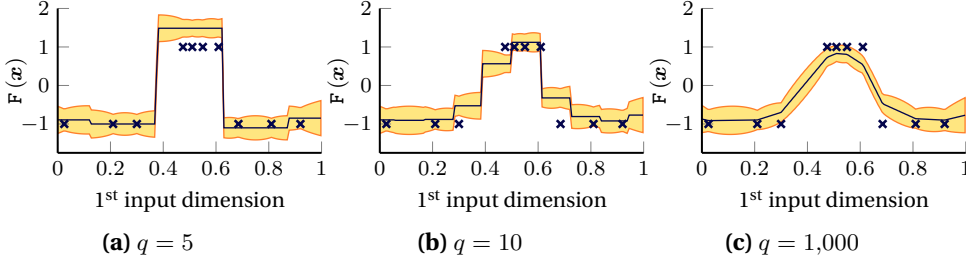


Figure 3.3: Predictive mean estimates $\tilde{\mu}_*$ with quantized test inputs for different number of intervals q . Remaining settings are identical to those used in Fig. 3.1, i.e., 2-dimensional histogram features are used and a noise level $\sigma_n^2 = 0.1$ is assumed. A comparison to the results without quantization shown in Fig. 3.1b is insightful.

computation of $\text{quant}(x_*(d))$ can be done in $\mathcal{O}(1)$. As an example, the previously introduced equidistant placement of interval borders in Eq. (3.12) fulfills this condition. Thus, we can overcome complexity limitations of common kernel-based methods and are thus competitive with linear models while keeping the flexibility of histogram kernel functions.

For gaining further insights, we visualized the obtained predictive distribution obtained with a quantization defined by Eq. (3.13) and with different quantization levels in Fig. 3.3. The setup is identical to the one in Fig. 3.1. When inspecting Fig. 3.3, we nicely observe the discontinuities at interval borders. While the first two examples serve for visualization only, the setting with 1,000 intervals is indeed realistic and is used within our experimental evaluation in Section 7.2. Notably, we can thereby rebuild the original decision function with negligible error.

3.2.2.3 Adaptive Quantization

The previous quantization was motivated for histogram features which are commonly L_1 -normalized. Nonetheless, the normalization property is not necessary for the application of κ_{GHIK} . Instead, we only require non-negativity of feature components. In consequence, recent image representations based on CNN activation are applicable as well after being squashed through rectified linear units (see (Krizhevsky et al., 2012)).

However, there is no theoretical motivation for normalizing CNN activations. Furthermore, the activation strength typically varies heavily between different filters. Therefore, we suggest using an adaptive quantization for these feature types by estimating $[l_d, u_d]$ from training data. Specifically, we still assume $l_d \equiv 0$ for all dimensions, but set u_d to the largest activation of filter d among all training examples. In Section 7.2.2.7, we will analyze the resulting benefit in practical applications.

3.2.2.4 Analysis of Quantization Errors

The previously derived computation time turned out to be independent of the actual number of training examples. Thus, they sound like the holy grail for kernel-based methods. However, the involved quantization leads inevitably to errors. In consequence, it is important to analyze which differences we can expect between correct estimates and approximated results obtained by applying the quantization trick.

Bounds on the Deviation of Classification Scores for Quantized Test Data: In the following, we will present an upper bound on the resulting error. A proof is given in Appendix B.1. Note that the presented result applies to a wide range of models including support vector machines. Nonetheless, we motivate our findings for the choice of predictive mean estimates obtained from a GP regression model.

Theorem 3.1 (Worst-case analysis for the error in the predictive mean with quantized inputs). *If a quantization with a maximum quantization error of ε_q is used, the error of the quantized predictive mean $\tilde{\mu}_*$ can be bounded as follows:*

$$|\mu_* - \tilde{\mu}_*| \leq D \cdot \varepsilon_q \cdot \|\alpha\|_1. \quad (3.14)$$

It is noteworthy that we can derive even finer bounds when taking the normalization property of histogram features into account. In general, it is not possible for normalized vectors with $\|x\|_1 = 1$ that an error of ε_q occurs in every dimension if $D \cdot \varepsilon_q > 1$. As an example, the error ε_q for the quantization in Eq. (3.13) is $\varepsilon_q = \frac{1}{2(q-1)}$ with $[l_d, u_d] = [0, 1]$, $d \in \{1, \dots, D\}$. The reason is that a quantized value can differ from its original input by not more than half of the bin size. For the extreme case of $q = 2$, *i.e.*, quantizing inputs to either zero or one, we get at most an error of $\varepsilon_q = \frac{1}{2}$ in each dimension. Obviously, this error can not occur more than twice for normalized histograms (a similar argumentation holds for splitting the error over multiple dimensions).

In the following, we state a sharper bound by following this observation. The provided bound requires the additional assumption that inputs are still L_1 -normalized after being quantized. Note that this assumption is reasonable if the distribution of quantization errors is with zero mean, *i.e.*, feature values are not systematically rounded to only larger or only smaller values. According to our experience, such scenarios occur for normalized high-dimensional histogram features. A tighter bound without this additional requirement, however, is not yet found and still requires future research. The proof for the following theorem is given in Appendix B.2.

Theorem 3.2 (A sharper worst-case analysis for the error in the predictive mean with quantized inputs). *Let ε_q be the maximum error during quantization. Furthermore, let inputs x_* and their quantized counterparts $\tilde{x}_*$ be normalized such that*

$\|\mathbf{x}_*\|_1 = \|\tilde{\mathbf{x}}_*\|_1 = 1$ and, in consequence, let $[l_d, u_d] = [0, 1]$, $d \in \{1, \dots, D\}$. Then, the error of the quantized predictive mean $\tilde{\mu}_*$ can be bounded as follows:

$$|\mu_* - \tilde{\mu}_*| \leq \frac{D \cdot \varepsilon_q}{2} \cdot \|\boldsymbol{\alpha}\|_1 . \quad (3.15)$$

When inspecting the result of Theorem 3.1 and Theorem 3.2, we note that the term $\|\boldsymbol{\alpha}\|_1$ is independent of the chosen quantization. Instead, the entire quantization is performed only for test inputs to quickly compute positions within the sorted permutations for each dimension. Thus, the quantization does not affect the weight vector which has been computed previously.

Even Sharper Bounds for SVM and GP Models: For certain models, we can further bound this term to obtain a more insightful error analysis. For SVM models, we notice in Eq. (2.70) that each Lagrangian multiplier λ_i is bounded by C from above where C denotes the trade-off parameter for the soft margin formulation. With $\alpha_i = \lambda_i y_i$, we obtain the connection $\|\boldsymbol{\alpha}\|_1 \leq C \cdot N$ (see also (Schölkopf and Smola, 2002)). As a result, we can rewrite the error bound as follows:

$$|\mu_* - \tilde{\mu}_*| \leq \frac{D \cdot \varepsilon_q}{2} \cdot C \cdot N . \quad (3.16)$$

Note that we used the symbols μ_* and $\tilde{\mu}_*$ to refer to SVM classification scores which are obtained without and with applying the quantization to test inputs.

For the choice of GP label regression models, *i.e.*, $y \in \{-1, 1\}$, Rodner (2011, p. 64) derived a similar bound regarding the norm of the weight vector $\boldsymbol{\alpha}$. When $\lambda_{\min}(\mathbf{K})$ denotes the smallest eigenvalue of the kernel matrix $\mathbf{K}$, the norm can be bounded by $\|\boldsymbol{\alpha}\|_2 \leq \frac{\sqrt{N}}{\lambda_{\min}(\mathbf{K}) + \sigma_n^2}$. Since $\mathbf{K}$ is a covariance matrix, it further holds that $\lambda_{\min}(\mathbf{K}) \geq 0$. Furthermore, L_1 -norm and L_2 -norm are related by the inequality $\|\mathbf{v}\|_1 \leq \sqrt{N} \|\mathbf{v}\|_2$ for $\mathbf{v} \in \mathbb{R}^N$ (Golub and Loan, 1996, p. 56). By combining the three aspects, we finally obtain the following upper bound for quantization errors:

$$|\mu_* - \tilde{\mu}_*| \leq \frac{D \cdot \varepsilon_q}{2} \cdot \frac{N}{\sigma_n^2} . \quad (3.17)$$

Implications of the Error Analysis: From the obtained bounds, we can derive several conclusions. First of all, the worst-case error increases linearly with the dimensionality and the number of training examples. This intuitively reflects the fact that the quantization is applied to each dimension and each entry of $\boldsymbol{\alpha}$ can be affected when computing the inner product. We can further notice in Eq. (3.16) and Eq. (3.17) that also the regularization directly affects the quantization error. A strong regularization, *e.g.*, a small C for SVM models (see Eq. (2.69)) or a large σ_n^2 for GP regression models, leads to smaller errors. This again reflects the intuition

that a strongly regularized decision function is less affected by small variations of inputs than a more complex function.

3.2.3 Efficient Model Training with (Generalized) Histogram Intersection Kernels

We have already seen in Section 2.1.5 that training of a Gaussian process model requires inverting the covariance matrix K to then solve for the weight vector $\alpha = K^{-1}y$. It can be easily seen that computing α is equivalent to optimizing

$$\operatorname{argmin}_{\alpha \in \mathbb{R}^N} \|K\alpha - y\| . \quad (3.18)$$

The latter one can be obtained by minimizing the following quadratic program (Bo and Sminchisescu, 2008):

$$\operatorname{argmin}_{\alpha \in \mathbb{R}^N} \alpha^T K \alpha - \alpha^T y , \quad (3.19)$$

which is similar to the SVM optimization problem given in Section 2.1.4.3 and can be solved with established toolboxes for quadratic programs. Alternatively, we can apply iterative linear solvers that directly optimize Eq. (3.18). In the following, we provide a brief overview on these techniques and then show how to combine them with κ_{HIK} for fast training of representer models.

3.2.3.1 A Short Introduction to Iterative Linear Solvers

Iteratively Solving Linear Equation Systems: As noted previously, the application of iterative linear solvers allows for solving Eq. (3.18) or Eq. (3.19) to obtain α . According to the name *iterative* solver, an initial guess v_0 is thereby incrementally updated which results in a series of estimates $v_0, v_1, \dots$. In each iteration i , the current solution v_i is computed from the previous one v_{i-1} by optimizing a defined criterion. This criterion depends on the specific solving routine and should ideally be related to reducing the difference of solutions $r_i = y - K v_i$. Note that the difference r_i is also known as *residual*.

Updating v_i then reduces to estimating a useful direction of change d_i and computing $v_i = v_{i-1} + \alpha d_i$ for a sufficiently chosen $\alpha \in \mathbb{R}$. An intuitive example is to select the negative gradient of Eq. (3.19) at the current solution as change direction, *i.e.*, $d_i = r_{i-1}$, which is known as *gradient descent* (Boyd and Vandenberghe, 2004, Sect. 9.3). A generalization to arbitrary norms is the steepest descent technique (see (Golub and Loan, 1996, Sect. 10.2.1) and (Boyd and Vandenberghe, 2004, Sect. 9.4) for further details). Unfortunately, steepest descent (and thus also gradient descent)

can result in an extremely slow convergence of v_i against the true solution v_* (Golub and Loan, 1996, Sect. 10.2.1).

Beyond Steepest Descent: Therefore, a variety of further techniques has been developed for computing search directions different from the residual vectors. One extreme is the technique of *coordinate descent*, where search directions d are instantiated with N basis vectors. Thus, in each iteration a single coordinate is optimized and a (locally) optimal solution is found if an entire cycle through all basis vectors did not change the current solution. In this chapter, however, we will focus on the *conjugate gradients* (CG) method introduced by Hestenes and Stiefel (1952).

In contrast to steepest descent and coordinate descent, the CG technique computes *conjugate* search directions, i.e., $d_i^T K d_j = 0$, $i \neq j$, which gives rise to the technique's name. When assuming computations without round-off errors, it can be shown that the exact solution v_* is guaranteed to be found in only N iterations for $y \in \mathbb{R}^N$ (Hestenes and Stiefel, 1952). Although this assumption might easily fail in practice, the sequence $v_1, v_2, \dots$ of approximated solutions is guaranteed to increase in accuracy (Hestenes and Stiefel, 1952, Theorem 6:3). Thus, the length of $v_i - v_*$ monotonically decreases.

Early Stopping and Pre-conditioning: The property of increasing accuracies justifies the interruption of the iterative process even before the exact solution is found. For intuitive reasons, this is known as *early stopping* (see Section 7.2.2). As an example, one commonly interrupts the computation after a maximum number of iterations T_1 is reached and still obtains a close approximation. Similarly, computations can be stopped when the norm of the current residual drops below a given threshold. Furthermore, from the perspective of model regularization in machine learning, it might even not be desirable to obtain the exact solution but only an approximation thereof. As mentioned by Perronnin et al. (2012), early stopping of iterative solvers results in faster training times while increasing generalization abilities especially for large numbers of training examples and known categories.

Finally, *preconditioning* of the iterative solver is beneficial if the kernel matrix is not well conditioned or exhibits a variety of different eigenvalues. For details of preconditioning, the interested reader is referred to (Golub and Loan, 1996, Sect. 10.3) and (Saad, 2003, Chapter 9). A great overview on various iterative linear solvers is given by Saad (2003). Further details can also be found in (Golub and Loan, 1996, Chapter 10).

3.2.3.2 Iterative Linear Solvers for Efficient Training

We have seen that the prominent CG techniques introduced previously converges in only $\mathcal{O}(N)$ iterations. However, each iteration still requires the evaluation of Kv which in turn demands $\mathcal{O}(N^2)$ operations in its plain formulation. Therefore, Quiñero Candela et al. (2007) pointed to the fact that the direct application of CG with explicit matrix vector multiplications is still not practical for large-scale scenarios. Even with early stopping after $k < N$ iterations, the resulting complexity is $\mathcal{O}(kN^2)$ and hence quadratic in the number of training examples.

As reviewed earlier, different solutions for reducing this computational burden have been developed, *e.g.*, approximated matrix vector products using KD trees as introduced by Shen et al. (2006). By definition, we can instead obtain exact solutions with FMKs due to the resulting possibility of efficient multiplications of vectors with kernel matrices in sub-quadratic time. As one example, we will see in the following that our technique based on (generalized) histogram intersection kernels allows for exact solutions due to the inherent properties of κ_{HIK} .

3.2.3.3 Fast Evaluations of Kv with κ_{HIK} for Iterative Linear Solvers

As noted previously, the successful application of iterative linear solvers heavily depends on the possibility for rapid matrix vector multiplications Kv for any given vector v . Fortunately, for the choice of κ_{HIK} , we can apply the same idea as for classifying unseen examples in Section 3.2.2 and rewrite the product Kv :

$$(Kv)(i) = \sum_{j=1}^N v(j) \cdot \kappa_{\text{HIK}}(x_i, x_j) \quad (3.20)$$

$$= \sum_{j=1}^N v(j) \sum_{d=1}^D \min(x_i, x_j) \quad (3.21)$$

$$= \sum_{d=1}^D \left(\sum_{\{j : x_j(d) < x_i(d)\}} v(j) x_j(d) + x_i(d) \sum_{\{k : x_k(d) \geq x_i(d)\}} v(k) \right). \quad (3.22)$$

As previously, the above derivations also hold for the generalized histogram intersection kernel κ_{GHIK} if $g(\cdot)$ is an automorphism.

In Section 3.2.2, we required permutations to sort training examples in each dimension for a fast classification. Although computing these permutations requires $\mathcal{O}(DN \log N)$ operations with standard sorting algorithms, this computational overhead needs to be done only once during training. As a result, we can now re-use them to also build look-up tables A and B for a given vector v as defined in Eq. (3.10). Since both matrices are of size $D \times N$, this step requires $\mathcal{O}(DN)$

operations.

Each entry $(Kv)(i)$ can then be obtained by indexing both matrices and summing all D values as already introduced for exact classification. One important difference - though - is the fact that the position of each example x_i does not need to be determined again. Instead, we can store the index for each example when computing the permutations and can thereby index A and B in $\mathcal{O}(1)$.

In summary, computing a single entry of Kv results in $\mathcal{O}(D)$ operations. Thereby, we can finally compute the entire matrix vector product in $\mathcal{O}(DN)$. It should be noted that we can again speed up the required computations by skipping zero-valued entries as already mentioned in Section 3.2.2. Furthermore, the memory demand is dominated by matrices A and B as well as permutations π_d and thereby results in $\mathcal{O}(DN)$. In consequence, the time spent for computing a single matrix vector multiplication does only scale linear and not quadratic in number of training examples. As noted previously, this allows for an efficient implementation within iterative linear solvers (see also (Léon Bottou, 2007, Sect. 9.4)).

3.2.3.4 Summary and Asymptotic Computation Times for Training

The previously introduced techniques for efficient training can be roughly split into four main parts:

- (i) sort features in each dimension (see Eq. (3.10) in Section 3.2.2.1),
- (ii) estimate weight vectors α using an iterative linear solver (see Section 3.2.3.3),
- (iii) compute matrices A and B (see Eq. (3.11) in Section 3.2.2.1), and
- (iv) optionally compute look-up table T for applying the quantization trick introduced in Section 3.2.2.2 .

Let us now summarize the asymptotic computation time required for each step to analyze the overall computational complexity. With standard sorting routines, step (i) can be achieved in $\mathcal{O}(DN \log N)$ operations. As noted previously, we often observe sparse features in practice which significantly reduces the computational burden during sorting. Solving the linear system in step (ii) is dominated by T_1 matrix vector multiplications, where T_1 is number of iterations of the ILS method. While T_1 can be set to the maximum number of desired iterations, the ILS can stop earlier depending on the condition number of the kernel matrix. We have seen in Section 3.2.3.3 that computing Kv for any vector v can be done in $\mathcal{O}(DN)$ time, thus, step (ii) scales with $\mathcal{O}(T_1 DN)$. Step (iii) is already performed as sub-routine in step (ii) and thus comes with no extra cost. To build the LUT T in step (iv), we can benefit from the available permutations π_d computed in step (i). Thereby, step (iv) requires additionally $\mathcal{O}(D \max(q, N))$ operations.

For multiple known categories, it is important to note that the first step is independent of class labels. In contrast, the remaining steps (ii) to (iv) are class-dependent and need to be carried out for each class separately when following the one-vs-all approach (see Eq. (2.132) in Section 2.1.5.8). In consequence, we thereby obtain an overall asymptotic computation time of $\mathcal{O}(DN(\log N + T_1 M))$ for learning.

From the previous analysis, we can draw three important conclusions. First of all, we can reduce the cubic complexity common for kernel methods to scale only with $N \log N$. Besides, the relative speed-up of our efficient techniques during training is higher for low-dimensional features but unaffected at test time. As final note, we observe that our method also scales linear with the number of classes M which allows for scalability towards large-scale scenarios with hundreds or even thousands of classes and gains even more importance in the context of lifelong learning systems.

3.3 Efficient Optimization of Hyperparameters

In the previous section, we were interested in efficient routines for training and classification with GP models when equipped with histogram intersection kernels. However, the probabilistic formulation of GPs as introduced in Section 2.1.5 offers far more than pure classification. One such aspect is the optimization of involved hyperparameters as introduced in Section 2.1.5.7 to optimally adapt a model to a specific task or dataset. The main findings of efficient hyperparameter optimization with GP and κ_{HIK} as presented in the following are published in (Rodner et al., 2012) and (Rodner et al., 2015a).

3.3.1 An Efficiently Computable Upper Bound of the Negative Log Likelihood

In Section 2.1.5.7, we introduced two approaches for optimizing the marginal likelihood $p(\mathbf{y} | \mathcal{X}, \boldsymbol{\eta})$: either by minimizing the negative log likelihood as given in Eq. (2.123) using any black-box optimization technique of our choice, or alternatively by computing roots of the likelihood's partial derivative as stated in Eq. (2.127). The latter one, though, is only possible if the covariance function κ is differentiable with respect to each hyperparameter which unfortunately is not the case for the histogram intersection kernel κ_{HIK} . We will therefore follow the first path by directly optimizing the marginal likelihood.

According to Eq. (2.123), computing the negative log likelihood mainly requires the evaluation of two terms: a quadratic data term $\mathbf{y}^T \mathbf{K}_{\boldsymbol{\eta}}^{-1} \mathbf{y}$ and a complexity term $\log \det(\mathbf{K}_{\boldsymbol{\eta}})$. With FMKs and the techniques for efficient training presented in the

previous Section 3.2.3, evaluating the first term can be easily achieved. However, the second term requires the kernel matrix's determinant, which is expensive to compute exactly (Yuster, 2008). We will therefore provide an efficiently computable upper bound for the negative log likelihood. Thereby, optimizing hyperparameters is reduced to minimizing this upper bound instead of the exact negative log likelihood.

Approximating the Determinant of K_η : Let us start with an upper bound on the determinant of a matrix which was provided by Bai and Golub (1997, Theorem 2) and which turns out to be efficiently computable with FMKs. For that, let us recall that the *Frobenius norm* of a matrix $M \in \mathbb{R}^{N \times N}$ is defined as

$$\|M\|_F = \sqrt{\sum_{i,j=1}^N M(i,j)^2} \quad , \quad (3.23)$$

which will play an important role for obtaining the upper bound.

Lemma 3.3 (An upper bound for a matrix's determinant by Bai and Golub (1997)).

Let M be a given matrix with $M \in \mathbb{R}^{N \times N}$. We further assume all eigenvalues λ_i of M to be bounded by $0 < \lambda_i \leq \beta$ with a positive constant $\beta \in \mathbb{R}_+$. In addition, we introduce the variables μ_1 and μ_2 which refer to the trace and the squared Frobenius norm of M , i.e., $\mu_1 = \text{tr}(M) = \sum_{i=1}^N \lambda_i$, and $\mu_2 = \|M\|_F^2 = \sum_{i,j=1}^N M(i,j)^2$. Finally, let $\bar{t}$ be defined as the following ratio:

$$\bar{t} = \frac{\beta\mu_1 - \mu_2}{\beta N - \mu_1} \quad . \quad (3.24)$$

If $\beta \neq \bar{t}$, the logarithm of M 's determinant can be bounded from above as follows

$$\log \det(M) \leq [\log \beta, \log \bar{t}] \begin{bmatrix} \beta & \bar{t} \\ \beta^2 & \bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (3.25)$$

$$\doteq \text{ub}(\beta, \mu_1, \mu_2) \quad . \quad (3.26)$$

For its justifiable application, it is essential that the bound is sufficiently close to the exact solution. Under this point of view, it is important to note that the bound is indeed tight for regularized rank-1 matrices $M = vv^T + \sigma I_N$, $v \in \mathbb{R}^N$, $\sigma \in \mathbb{R}_+$, which can be easily derived from the result provided by Bai and Golub (1997, Example 1) on so-called Pei matrices. In addition, kernel matrices ideally obtain a clear block structure in multi-class classification scenarios since examples of each category are often clustered in space. Thus, K is often of low rank which is supported by the observation by Williams and Seeger (2000). Thus, the bound

should be well suited for visual recognition task as considered within this thesis.

It now remains to compute β , μ_1 , and μ_2 based on the previously introduced techniques for fast multiplication kernels such as κ_{HIK} . We therefore note that the smallest possible value for β is the largest eigenvalue $\lambda_{\max}$ itself. The largest eigenvalues can be obtained in affordable time using iterative decomposition techniques, *e.g.*, the *Arnoldi iteration*¹ (Golub and Loan, 1996, Sect. 9.4). Since the Arnoldi iteration solely requires a matrix through matrix vector multiplications, we can again easily re-use our previously derived techniques for κ_{HIK} . We empirically found that convergence usually occurs after only a small number of iterations.

Computing the Trace of K_η : With respect to the trace $\text{tr}(K)$, we remember from linear algebra that we can compute $\text{tr}(K)$ by summing all entries of M 's main diagonal. For a kernel matrix, this equals adding all self-similarity values $\kappa(x_i, x_i)$. It should be noted that self-similarity values are constant for many popular kernel functions and thus lead to an easy computation of the trace. For the choice of κ_{HIK} , we can even further note that the trace of the kernel matrix equals the summation of values in all dimensions of all training examples augmented by the noise values added for model regularization (see also the combined covariance function for GP models introduced in Eq. (2.124)):

$$\text{tr}(K_\eta) = \sum_{i=1}^N (\kappa_{\text{HIK}}(x_i, x_i) + \sigma_n^2) \quad (3.27)$$

$$= \left(\sum_{i=1}^N \sum_{d=1}^D x_i(d) \right) + N\sigma_n^2. \quad (3.28)$$

For histogram features normalized to length one under L_1 -norm, this finally translates to:

$$\text{tr}(K_\eta) = N(1 + \sigma_n^2). \quad (3.29)$$

Note that similar derivations hold for κ_{GHIK} .

Computing the Frobenius Norm of K_η : The last missing part for evaluating the upper bound in Eq. (3.26) is the squared Frobenius norm $|K|_F^2$. According to its definition as squared sum of all matrix entries, we immediately observe that $|K|_F^2$ is not directly available since we spent major efforts in Section 3.2 into techniques which circumvent the explicit computation of K . However, the definition of $|K|_F^2$

¹It should be noted that for symmetric matrices, the Arnoldi technique reduces to the Lanczos iteration technique, for which well-established theoretical results on convergence properties exist. See (Golub and Loan, 1996, Sect. 9.4) and references therein for further details and valuable hints on practical implementations.

is equivalent to the sum of all squared eigenvalues, if $\mathbf{K}$ is symmetric - which holds for kernel matrices:

$$|\mathbf{K}|_F^2 = \sum_{i=1}^N \lambda_i^2 . \quad (3.30)$$

As noted previously, we often observe $\mathbf{K}$ of low rank for visual recognition tasks, and thus the sum of all eigenvalues is clearly dominated by the first M elements:

$$|\mathbf{K}|_F^2 \lesssim \sum_{i=1}^M \lambda_i^2 . \quad (3.31)$$

Again, eigenvalues λ_i are to be understood in decreasing order for notational convenience. Further note that it is not obvious that a lower bound approximation of $|\mathbf{K}|_F^2$ applied within Lemma 3.3 still leads to a valid upper bound for the log-determinant. In Appendix B.3, we therefore give a proof consisting of two lemmas and one theorem showing that we indeed maintain a valid upper bound.

The Resulting Bound: When combining the previous aspects, we can compute the general upper bound as:

$$\log \det (\mathbf{K}_\eta) \leq \text{ub} \left(\lambda_{\max}, \text{tr} (\mathbf{K}_\eta), \sum_{i=1}^M \lambda_i^2 \right) . \quad (3.32)$$

Finally, for the choice of κ_{HIK} as kernel function and L_1 -normalized features we obtain the following upper bound :

$$\log \det (\mathbf{K}_\eta) \leq \text{ub} \left(\lambda_{\max}, N (1 + \sigma_n^2), \sum_{i=1}^M \lambda_i^2 \right) . \quad (3.33)$$

In summary, we have efficient techniques for computing or bounding both terms required for computing the negative log-likelihood. Thus, we can apply established techniques from optimization theory to optimally adjust hyperparameters to given training data, and we give a short overview in the following.

3.3.2 Gradient-free Optimization

With the previously presented techniques, we can efficiently approximate the log likelihood associated with a certain setting of involved hyperparameters. Unfortunately, the presented techniques do not apply to the likelihood's derivative. Thus, we can not directly obtain the optimal solution η_* .

Instead, we need to iteratively improve a current solution – similar to the itera-

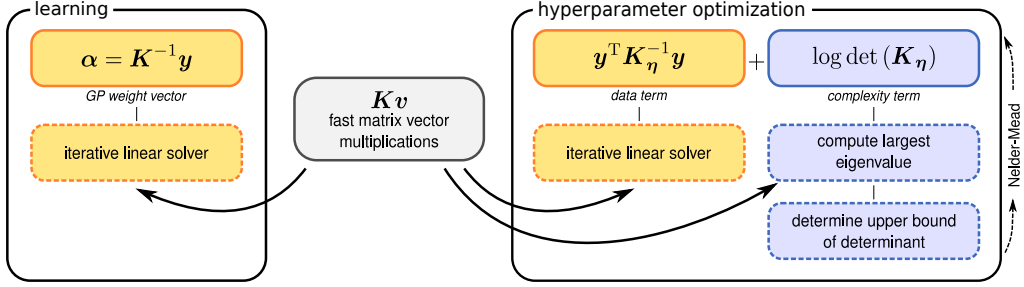


Figure 3.4: Visualization of the interplay of key ingredients for training and optimization of hyperparameters which rely on efficient computations with fast multiplication kernels such as κ_{HIK} . Boxes with solid lines indicate required terms whereas dashed lines represent corresponding techniques for their efficient computation. We additionally colored boxes in orange which have been introduced in Section 3.2.3 for model training. Instead, boxes colored in blue correspond to techniques specifically introduced in Section 3.3 for optimization of hyperparameters.

tive linear solver scheme in Section 3.2.3.1. In other words: we seek for the vector η_* with largest likelihood as fast as possible, *i.e.*, with as few likelihood evaluations for different vectors $\eta_1, \eta_2, \dots$ as possible. While any black-box algorithm can be applied to achieve this goal, we will rely on the downhill-simplex algorithm by Nelder and Mead (1965), which we experimentally proved to be useful within image categorization scenarios.

The downhill-simplex algorithm can be roughly described as technique which iteratively moves the corner points of a polytope of initially large size such that the optimal solution is likely covered by the polytope’s volume. The underlying idea is to move corner points such that the volume is increased into the directions of currently high-scored corners, and decreased in low-scored dimensions. Several parameters can be adjusted, *e.g.*, for tuning the speed of expansion and shrinking, the relative importance of moving corners away from the current polytope, or the largest number of desired iterations. For staying self-contained, we will skip a more detailed introduction to the Nelder-Mead technique here and instead refer to (Press et al., 1999) and (Nelder and Mead, 1965).

With all these techniques combined, we can now efficiently train a GP model with κ_{HIK} as covariance function, we can optimize involved hyperparameters specifically for given training data, and we can apply our model with fast evaluations on previously unseen test data. Fig. 3.4 gives an overview on the interplay of key ingredients of training and optimization which rely on efficient computations with κ_{HIK} and relevant steps are put into perspective.

We additionally summarized all asymptotic computation times as well as memory demand in Table 3.1 and compared it with the required resources of GP in its plain formulation as derived in Section 2.1.5.9. A compact overview focusing only

Table 3.1: Overview of asymptotic computation times and memory demand for GP label regression in its plain formulation, with (G)HIKs, and when quantizing test inputs. See also text for detailed explanations. T_1 and T_2 denote the number of iterations required by the iterative linear solver and the optimization routine, respectively. A more compact overview is given in Table 3.2.

Step	Asymptotic computation time		
	GP baseline	GP/HIK	GP/HIK + Quant.
Learning	$\mathcal{O}(N^3 + (D + M)N^2)$	$\mathcal{O}(DN(\log N + T_1M))$	$\mathcal{O}(DN(\log N + T_1M))$
Hyperp. opt.	$\mathcal{O}(T_2(N^3 + MN^2))$	$\mathcal{O}(T_2NMDT_1)$	$\mathcal{O}(T_2NMDT_1)$
Testing	$\mathcal{O}(MDN)$	$\mathcal{O}(MD \log N)$	$\mathcal{O}(MD)$
Memory	$\mathcal{O}(N^2 + N(D + M))$	$\mathcal{O}(DN)$	$\mathcal{O}(D \max(q, N))$

Table 3.2: A compact overview of asymptotic computation times and memory demand for GP label regression in its plain formulation, with (G)HIKs, and when quantizing test inputs with focus on number of training examples N only. Note that complexity constants which are hidden in the $\mathcal{O}$ -notation can differ.

Step	Asymptotic computation time		
	GP baseline	GP/HIK	GP/HIK + Quant.
Learning	$\mathcal{O}(N^3)$	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$
Hyperp. opt.	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Testing	$\mathcal{O}(N)$	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$
Memory	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$

on the dependence on N can finally be found in Table 3.2.

3.4 Efficient Variance Computation with Selectable Precision

In the previous two sections, we presented techniques to train Gaussian process label regression models from large and high-dimensional data, to optimize involved hyperparameters based on likelihood maximization, and to classify unseen inputs.

Our techniques have only limited by the choice of fast multiplication kernels (FMKs) as covariance function, *e.g.*, the histogram intersection kernel κ_{HIK} . For all involved aspects, we have seen that our techniques reduce involved computation times and memory demand by orders of magnitude. However, we already highlighted an important property of Gaussian process models in Section 2.1.5. In contrast to many other approaches, *e.g.*, nearest neighbor techniques (see Section 2.1.4.1) or SVM models (see Section 2.1.4.3), GPs allow for obtaining statements with respect to the uncertainty of decisions rather than classification results only. Intuitively, this information is highly relevant in several application areas, *e.g.*, medical assistance, where a wrong decision can easily result in severe health issues, or in automotive applications, where a wrong decision can be harmful for remaining vehicles and drivers. In these cases, an interpretable and reliable estimator for uncertainty can indicate whether or not the decision is trustworthy, and can eventually launch additional security checks or ask for an expert's guidance. Furthermore, in lifelong learning scenarios, highly uncertain decisions can hint to examples with yet unknown content. Thus, also novelty detection and active learning (see Chapter 5) can benefit from this additional information to discover informative examples. In this section, we therefore present developed techniques for efficiently computing the Gaussian process predictive variance as defined in Eq. (2.100) even for large numbers of training examples and input dimensions.

Therefore, we recall that the predictive variance is defined as (see Eq. (2.100)):

$$\sigma_*^2 = k_{**} - \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{k}_* . \quad (3.34)$$

Let us stress again that we implicitly include an additive Gaussian noise model for regularization by applying a combined covariance function as introduced in Eq. (2.124). Furthermore, we notice that the predictive variance is independent of class labels and solely depends on kernel values evaluated between training and test examples. Thus, σ_*^2 is shared among all categories and efficient computations as introduced in the following scale even to scenarios with large number of classes. Our findings are published in (Freytag et al., 2012b), which received a best paper honorable mention award. All techniques are additionally summarized in (Rodner et al., 2015a).

3.4.1 Exact Computations of the Predictive Variance

Let us start with an exact technique for computing the predictive variance with fast multiplication kernels. In Section 3.2, we already introduced techniques for efficiently computing products $\mathbf{K}\mathbf{v}$ and $\mathbf{k}_*^T \mathbf{v}$ for vectors $\mathbf{v} \in \mathbb{R}^N$. As a first step, we will now adapt these ideas to also compute σ_*^2 . The solution of this technique will be exact, *i.e.*, within the accuracy of the iterative linear solver. Therefore, we will refer to

this strategy as *precise uncertainty prediction (PUP)*. Note that we actually compute the predictive variance rather than the closely related classification uncertainty, however, we stick to the term *uncertainty* for staying consistent with our previous publications (Freytag et al., 2012b) and (Rodner et al., 2015a).

To compute the quadratic term in Eq. (3.34), we require three steps:

- (i) compute $\mathbf{k}_*$ explicitly,
- (ii) solve $\boldsymbol{\alpha}_* = \mathbf{K}^{-1}\mathbf{k}_*$ with ILS, and
- (iii) compute $\mathbf{k}_*^T \boldsymbol{\alpha}_*$ explicitly.

The first step of explicitly computing $\mathbf{k}_*$ for an unseen test input $\mathbf{x}_*$ is indeed an expensive operation requiring $\mathcal{O}(ND)$ operations. However, we inevitably need $\mathbf{k}_*$ to apply the previously presented techniques. With $\mathbf{k}_*$ at hand, we can run the iterative linear solver introduced in Section 3.2.3 to obtain $\boldsymbol{\alpha}_* = \mathbf{K}^{-1}\mathbf{k}_*$. We should stress that this needs to be done for each test input $\mathbf{x}_*$ and requires $\mathcal{O}(DNT_1)$ operations. In experimental evaluations, we observed that T_1 , the required number of iterations until convergence, depends on the condition of the kernel matrix and can thus be influenced by model regularization. Once we found a solution for the linear equation system, we can finally compute $\mathbf{k}_*^T \boldsymbol{\alpha}_*$, which additionally requires $\mathcal{O}(N)$ operations².

In summary, we need to spent $\mathcal{O}(DNT_1)$ operations to compute the exact predictive variance by following the three steps. As mentioned before, though, the number T_1 of iterations until convergence is implicitly connected to N . In consequence, we often observe a significant increase of computation time for increasing number of known examples which renders PUP as too slow for many large-scale scenarios. Additionally, we are sometimes only interested in the relative order of uncertainties with respect to several examples rather than in the exact scores (e.g., in active learning, we could be interested in selecting the most uncertain point, no matter *how* uncertain). We therefore introduce efficient approximations in the following sections.

3.4.2 Approximating the Predictive Variance with Adjustable Precision

The approximations introduced in this section will mainly be based on results from matrix algebra. First of all, we note that $\mathbf{K}$ is symmetric and positive definite, thus, both properties also hold for $\mathbf{K}^{-1}$. In Appendix B.4, we derived bounds for

²Note that it would be unwise to apply techniques for efficient evaluations of $\mathbf{k}_*^T \boldsymbol{\alpha}_*$ as presented in Section 3.2.2 here. This would require to additionally build look-up tables $\mathbf{A}$ and $\mathbf{B}$ specifically for $\boldsymbol{\alpha}_*$ which would already result in $\mathcal{O}(ND)$ operations.

quadratic forms on these types of matrices. Of major importance will be a lower bound for symmetric and positive definite matrices $M \in \mathbb{R}^{D \times D}$ and vectors $x \in \mathbb{R}^D$. Let therefore denote $\lambda_1 \geq \dots \geq \lambda_D$ the eigenvalues of M in decreasing order and let further $\tilde{x}(i)$ be the signed length of x 's projection³ onto the i^{th} eigenvector of M . As derived in Eq. (B.62), Appendix B.4, we can bound the quadratic form as follows:

$$x^T M x \geq \left(\sum_{i=D-k+1}^D \lambda_i \tilde{x}(i)^2 \right) + \lambda_{D-k} \left(\|x\|^2 - \sum_{j=D-k+1}^D \tilde{x}(j)^2 \right). \quad (3.35)$$

We can now use the obtained bound to approximate the predictive variance as given in Eq. (3.34) from above. In detail, let us replace the exact quadratic form $k_*^T K^{-1} k_*$ with the lower bound in Eq. (3.35):

$$\sigma_*^2 = k_{**} - k_*^T K^{-1} k_* \quad (3.36)$$

$$\leq k_{**} - \left(\sum_{j=N-k+1}^N \lambda_j \tilde{k}_*(j)^2 \right) - \lambda_{N-k} \left(\|k_*\|^2 - \sum_{j=N-k+1}^N \tilde{k}_*(j)^2 \right), \quad (3.37)$$

where we followed the previous notation and introduced $\tilde{k}_*(i)$ as the projection of k_* on the i^{th} eigenvector of K^{-1} . It now remains to compute the $k+1$ smallest eigenvalues of K^{-1} along with the k corresponding eigenvectors. Unfortunately, computing the smallest eigenvalues is known to suffer from numerical instability. Furthermore, we can not directly access K^{-1} when applying the techniques for efficient training and testing introduced in the previous chapters. To circumvent these drawbacks, let us reformulate the previous bound by solely referring to K instead of its inverse. Therefore, let ξ_i denote the i^{th} largest eigenvalue of K , which consequently gives $\xi_i = \frac{1}{\lambda_{N-i+1}}$. Furthermore, let $\nu(i)$ denote the projection of k_* on the i^{th} eigenvector of K . Similar to the relations between original and inverse matrix with respect to eigenvalues, eigenvectors of K are identical to those of K^{-1} but in inverse order since K is symmetric and positive-definite. Therefore, we obtain the relation $\nu(i) = \tilde{k}_*(N-i+1)$. In consequence, we can compute an upper bound approximation for the predictive variance as follows:

$$\sigma_*^2 \leq k_{**} - \left(\sum_{i=1}^k \frac{1}{\xi_i} \nu(i)^2 \right) - \frac{1}{\xi_{k+1}} \left(\|k_*\|^2 - \sum_{i=1}^k \nu(i)^2 \right). \quad (3.38)$$

³For an improved readability, we will denote the “signed length of the projection of vector a on vector b ” as “projection of a on b ” in the remaining part of this thesis.

Depending on the chosen number of eigenvalues and eigenvectors, the resulting approximation will vary in accuracy.

Let us shortly inspect the resulting computation time. In each Arnoldi iteration, $k + 1$ matrix vector products need to be computed where each takes $\mathcal{O}(DN)$ operations with our techniques introduced earlier. Thus, we can obtain $k + 1$ eigenvalues and corresponding eigenvectors in $\mathcal{O}(T_3 k DN)$ where T_3 denotes the number of Arnoldi iterations needed to compute eigenvalues and eigenvectors. As noted previously, we empirically found that T_3 stays surprisingly small with $T_3 \approx 10$. Computing $\mathbf{k}_*$ requires $\mathcal{O}(DN)$ as for the PUP method. Projecting $\mathbf{k}_*$ on k eigenvectors and computing $\|\mathbf{k}_*\|$ can finally be done in $\mathcal{O}(N)$ operations. In summary, we require an asymptotic computation time of $\mathcal{O}(T_3 k DN)$. By adjusting k we can thereby find a suitable trade-off between desired precision and affordable computation time. We therefore refer to Eq. (3.38) as *fine approximation of the predictive uncertainty (FAPU)*.

A visualization of the resulting approximation is given in Figs. 3.5a to 3.5c using the previous $2D$ example of Fig. 3.1 and Fig. 3.3. We can nicely observe that with smaller numbers of eigenvectors used, the approximation error gets larger and the original quadratic form converges to a linear function.

3.4.3 A Coarse Approximation of the Predictive Variance with Histogram Intersection Kernels

Taking the previous approximation in Eq. (3.38) to the extreme case with $k = 0$ would lead to:

$$\sigma_*^2 \leq k_{**} - \frac{1}{\mu_1} \|\mathbf{k}_*\|^2. \quad (3.39)$$

However, even this fastest version still requires explicitly computing $\mathbf{k}_*$ and thus demands $\mathcal{O}(DN)$ operations even with the efficient techniques introduced earlier. When using a (generalized) histogram intersection kernel, though, we can obtain an approximation which can be computed in sub-linear time with respect to N . Let us therefore notice that term $\|\mathbf{k}_*\|^2$ can be written as:

$$\|\mathbf{k}_*\|^2 = \mathbf{k}_*^T \cdot \mathbf{k}_* = \sum_{i=1}^N \left(\sum_{d=1}^D \min(\mathbf{x}_i(d), \mathbf{x}_*(d)) \right)^2 \quad (3.40)$$

$$= \sum_{i=1}^N \left(\sum_{d=1}^D \left(\min(\mathbf{x}_i(d), \mathbf{x}_*(d))^2 \right) + \sum_{\substack{d, d'=1 \\ d \neq d'}}^D \left(\min(\mathbf{x}_i(d), \mathbf{x}_*(d)) \min(\mathbf{x}_i(d'), \mathbf{x}_*(d')) \right) \right). \quad (3.41)$$

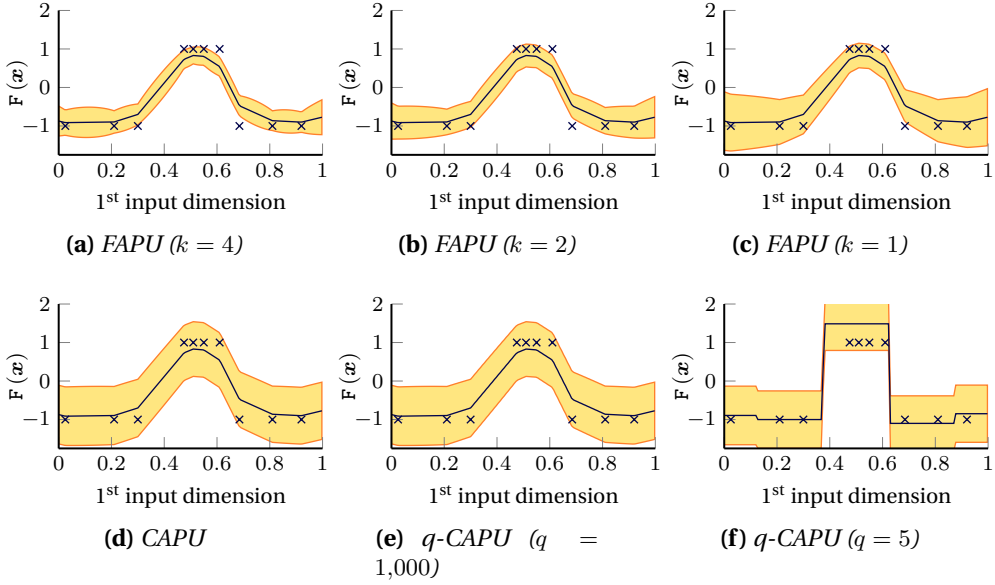


Figure 3.5: Different approximations of predictive variance estimates using GP regression models equipped with κ_{HIK} . Top row: results are obtained applying the fine approximation technique introduced in Section 3.4.2 with varying numbers of eigenvectors. Bottom row: using the coarse approximation as introduced in Section 3.4.3 without and with additional quantization of test inputs. The setup is identical to Fig. 3.1 and Fig. 3.3 where no approximation was applied to compute the predictive variance.

Especially if features are sparse, which is common for high-dimensional features, most of the mixed terms

$$\min(x_i(d), x_*(d)) \min(x_i(d'), x_*(d'))$$

in the second sum of Eq. (3.41) are equal to zero. The remaining terms will also be mostly small in value due to the minimum operation. As a numerical example, a sparsity ratio of 10% already results in 99% of the mixed terms which vanish. Depending on the application and the chosen feature type, it can therefore be well justifiable to neglect the sum over mixed terms and approximate $\|k_*\|^2$ with the

following lower bound:

$$||\mathbf{k}_*||^2 \geq \sum_{i=1}^N \sum_{d=1}^D \min\left(\mathbf{x}_i(d), \mathbf{x}_*(d)\right)^2 \quad (3.42)$$

$$= \sum_{i=1}^N \sum_{d=1}^D \min\left(\mathbf{x}_i(d)^2, \mathbf{x}_*(d)^2\right). \quad (3.43)$$

Note that Eq. (3.43) only follows for non-negative features which – however – is the field of application for κ_{HK} . The resulting upper bound for σ_*^2 is:

$$\sigma_*^2 \leq k_{**} - \sum_{i=1}^N \sum_{d=1}^D \min\left(\mathbf{x}_i(d)^2, \mathbf{x}_*(d)^2\right). \quad (3.44)$$

In comparison with our previously introduced techniques for fast classifications of unseen test inputs (see Section 3.2.2.1), we notice that Eq. (3.43) can be obtained from Eq. (3.10) with squared feature values and $\alpha = 1$. We can thereby apply these techniques and even re-use the permutations π_d already computed during training. The only additional burden is to compute the look-up table $\mathbf{A}_{\sigma_*^2}$ which contains cumulative sums of squared feature values similar to $\mathbf{A}$ in Eq. (3.11). Note that we do not need the second table $\mathbf{B}$ for $\alpha = 1$. Since we can not influence the accuracy of the approximation anymore and the bound is less accurate than the FAPU approximation, we refer to the approximation obtained with Eq. (3.43) as *coarse approximation of the predictive uncertainty (CAPU)*.

Similar to classification, we can again apply the quantization idea to unseen test inputs to further speed up involved computations. Consequently, we refer to it as *coarse approximation of the predictive uncertainty with quantized inputs (q-CAPU)*. During training, we thus need to compute an additional table $\mathbf{T}_{\sigma_*^2}$ which however does not change asymptotic scaling of the entire training process. To compute the predictive variance approximations for unseen test inputs, required computations scale identical to the classification counterpart (see Table 3.1) except that we drop the dependency on the number of classes M . Thus, we obtain asymptotic computations times of $\mathcal{O}(D \log N)$ and $\mathcal{O}(D)$ for CAPU and q-CAPU, respectively.

A visualization of the approximation with and without quantized test inputs is given in Figs. 3.5d to 3.5f. In Fig. 3.5d, we notice that the predictive variance collapsed from a quadratic to a linear function with respect to $\mathbf{k}_*$. This can also be seen in Eq. (3.44). When additionally applying the quantization trick, we can further trade speed against accuracy depending on the quantization's coarseness (Fig. 3.5e and Fig. 3.5f). In comparison with Fig. 3.3 we again observe the same discontinuities in the confidence areas but with a piecewise linear instead of a piecewise quadratic boundary.

Table 3.3: Summary of different techniques for computing or approximating the predictive variance σ_*^2 . See text for detailed derivations and explanations. k denotes the selected number of eigenvectors to approximate K . Variables T_1 and T_3 denote the number of iterations needed by the iterative linear solver and the Arnoldi technique, respectively. Dependency with respect to number of training examples is additionally highlighted in **bold font**.

Method	Asymptotic computation time	Resulting score	Approximation
GP-standard	$\mathcal{O}(N^2 + ND)$	Eq. (3.34)	none (exact)
PUP	$\mathcal{O}(DNT_1)$	Eq. (3.34)	none (exact)
FAPU	$\mathcal{O}(T_3kDN)$	Eq. (3.38)	rank k approx. of K
CAPU	$\mathcal{O}(D \log N)$	Eq. (3.44)	rank 1 approx. of K , sparse feature assumption
q-CAPU	$\mathcal{O}(D)$	Eq. (3.44)	rank 1 approx. of K , sparse feature assumption, quantized test inputs

In summary, we introduced four techniques to compute or approximate the predictive variance σ_*^2 for unseen test inputs x_* . The techniques differ in the precision of the approximation as well as computational complexity. An overview over all techniques is given in Table 3.3.

3.5 Semantic Segmentation with GP Models

In this section, we will introduce an application scenario where the scalability of Gaussian process models to large-scale data is highly valuable. To this end, we will increase the level of investigation from image classification to a more localized analysis, which will allow us to obtain a semantic label for each pixel in a given image. Since the number of pixels in images is vast, we immediately observe large numbers of examples during both training and testing. Thus, the applicability of sophisticated classification models heavily depends on the availability of scalable algorithms as the previously introduced techniques for Gaussian process regression models with fast multiplication kernels, *e.g.*, the histogram intersection kernel κ_{HIK} . Our findings are published in (Freytag et al., 2012a) and can additionally be found in (Rodner et al., 2015a).

3.5.1 A Brief Introduction to Semantic Segmentation

Dividing an image into meaningful parts is often a crucial step for a detailed analysis thereof. Within the computer vision community, this process of partitioning is

referred to as segmentation. Each segment then covers connected pixels which fulfill certain similarity criteria, *e.g.*, similar color distributions, a joint location inside strong borders, or the same category all selected pixels belong to. Depending on the chosen similarity criterion, we can distinguish between two general classes of segmentation techniques: unsupervised and supervised segmentation.

Unsupervised segmentation are designed or learned to images into consistent parts, but without explicitly requiring class label annotations to define class consistencies. Thus, resulting techniques commonly rely only on visual cues to decide upon which pixels to group into segments. Established approaches are the graph-based segmentation by Felzenszwalb and Huttenlocher (2004), MeanShift by Comaniciu and Meer (2002), or SLIC by (Achanta et al., 2012). Applying unsupervised segmentation techniques to new images results in a set of regions which ideally are of high visual consistency. However, for a detailed analysis of the image, a semantic interpretation of each region is yet missing.

In contrast, *semantic segmentation* refers to techniques which aim at additionally estimating the category each pixel belongs to. Thus, a model trained from additional annotated data is applied to add a *semantic* interpretation to regions and to allow for a semantically rich analysis. Note that we can distinguish between binary segmentation tasks which separate foreground from background and multi-class segmentation tasks which assign each pixel to one out of M specified categories (*e.g.*). Two examples for the first class of algorithms are (Rother et al., 2004) for interactive foreground segmentation and (Budai et al., 2013) for a medical application scenario. In contrast to these approaches, we are interested in multi-class semantic segmentation in the following.

To tackle this task, several general pipelines have been developed which mainly differ in the amount of domain knowledge to be added. Here, we follow an approach which was introduced by Csurka and Perronnin (2010), which was further used by Fröhlich et al. (2010), and which consists of five steps:

- (i) unsupervised segmentation,
- (ii) extraction of local features for individual pixels or patches,
- (iii) an optional feature enhancement step,
- (iv) classification of all features, and
- (v) combination of classification results and segmentation output.

The five steps are additionally visualized in Fig. 3.6. In contrast to other techniques, this pipeline does not rely on manually defined prior knowledge and instead learns semantic relations entirely from provided training data. Further techniques additionally incorporate context information, *e.g.*, AutoContext (*e.g.*, (Tu and Bai, 2010)),

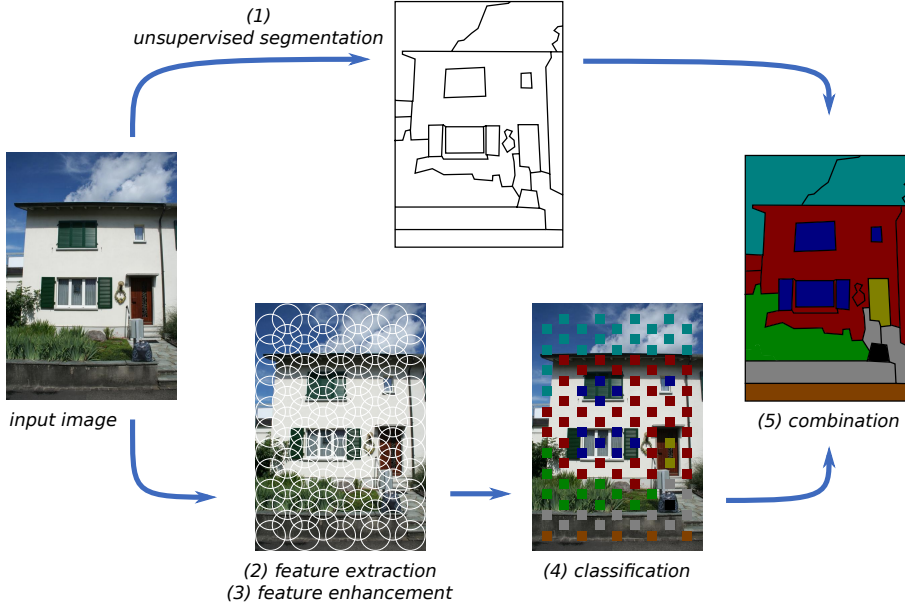


Figure 3.6: A general pipeline for semantic segmentation consisting of five steps: (1) unsupervised segmentation, (2) extraction of local features for individual pixels or patches, (3) an optional feature enhancement step, (4) classification of all features, and (5) combination of classification results and segmentation output.

iterative context forests (e.g., (Fröhlich et al., 2012; Fröhlich et al., 2012a)), conditional random fields (e.g., (He et al., 2004; Shotton et al., 2009)), and even latest results using CNNs can be viewed from this perspective (e.g., (Brust et al., 2015; Long et al., 2015)). Following the scope of this thesis, however, we will restrict us to the aforementioned pipeline without additional contextual information. Instead, we will focus on how to apply GP models to the task of semantic segmentation – a task which would not have been possible with plain GP models due to their excessive demand of memory and computation time.

3.5.2 Efficient Semantic Segmentation with GP Models

To apply our previously introduced techniques for scalable GP models, we need to ensure that inputs are histogram features to justify the utilization of κ_{HIK} or κ_{GHIK} as covariance function. We therefore follow Csurka and Perronnin (2010) and Fröhlich et al. (2010) and convert local features into BoW-like response vectors. Therefore, a Gaussian mixture model is trained with local features extracted from training images and assignment probabilities for each cluster are used to build the final representation for each local patch. Since the resulting representations are known to be highly sparse (Fröhlich, 2013, Sect. 3.1.1), they are perfectly suitable as

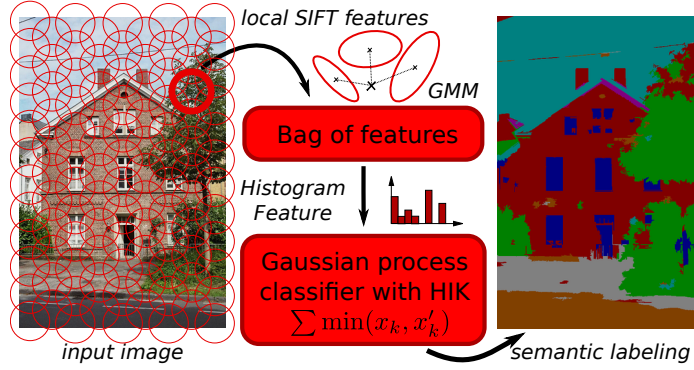


Figure 3.7: Semantic segmentation with scalable Gaussian process models using κ_{HIK} . Following the previously introduced pipeline for semantic segmentation (see also Fig. 3.6), local features are enhanced by computing BoW-like cluster assignments using GMMs to form histogram features which are then classified using the techniques presented earlier in this chapter. Classification results are pooled over regions obtained from an unsupervised segmentation to compute the final semantic segmentation estimate.

inputs for our efficient GP techniques. For a more detailed description, we refer to (Fröhlich, 2013, Sect. 3.1).

Given an annotated dataset with class labels for each pixel, we can thereby extract features on a regular grid to train a GP in a one-vs-all manner. For unseen test images, we then apply the same feature extraction and classify each image patch separately. Classification scores for each class are further smoothed over a local neighborhood to leverage classification results from the regularly spaced grid to all image pixels. For each region obtained from an unsupervised segmentation, we then average smoothed classification scores over all pixels located in that region which yields a single feature vector in $\mathbb{R}^M$ as representation of the entire region. A final decision is obtained by assigning each region with the category label corresponding to the highest averaged score. An overview of the resulting framework is given in Fig. 3.7.

In Fig. 3.8, we present an example of applying the introduced pipeline for semantic segmentation to unseen test images to provide a first idea about the quality of obtainable results. First of all, we note that the overall impression of the labeling result is visually close to the human annotations. This is indeed remarkable given the fact that only local patches are individually classified. In consequence, subtle errors due to missing contextual information can occur, *e.g.*, regions labeled as car on top of the building, which can be removed by adding a context-oriented post-processing. Furthermore, also human annotations are error-prone, *e.g.*, the lower left window was not labeled appropriately by humans, but correctly recognized by our segmentation algorithm. Further results are provided in Section 7.2.5.

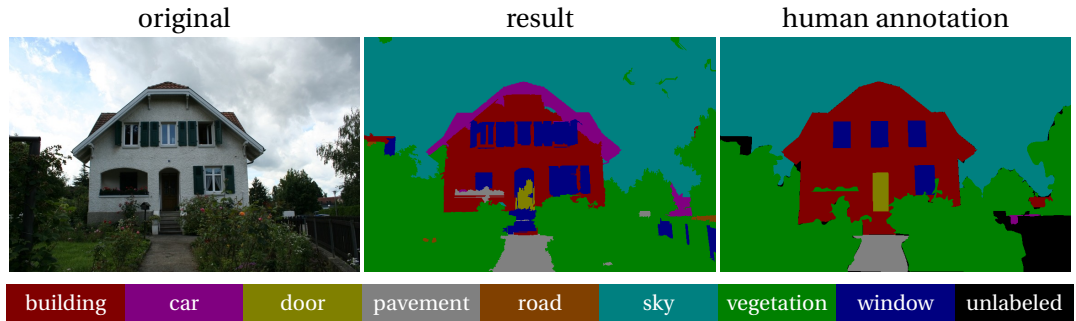


Figure 3.8: A qualitative result for semantic segmentation with Gaussian process models and κ_{HIK} . An unseen test image (left) is propagated through the pipeline for semantic segmentation as introduced in Section 3.5.1. The resulting labeling with respect to each pixel's category as one of eight distinct classes is color coded (middle). Human annotations serving as ground truth are shown right. See text for further explanations.

4 | Incremental Learning with GP Models

“How to deal with new objects and classes?”

In the previous chapter, we tackled techniques to train Gaussian process models with label regression even from tens of thousands of examples. Due to memory and computation time requirements, this has not been feasible in their plain formulation as introduced in Section 2.1.5 . Given our presented techniques, we were able to reduce the consumption of resources model training, optimization of hyperparameters, and inference to unseen test inputs by orders of magnitude. However, even with fast techniques at hand, it is yet unclear how to proceed if we encounter changes in the labeled training dataset $\mathcal{L}$. A naive solution would remove the previously trained model and start training from scratch using the new dataset. However, in many cases we can draw advantage from previously found model parameters to speed up re-learning. Since we expect only parts of the data to be changed, we will also refer to the process of re-learning as *model update*.

Three Types of Changes in Training Data: Let us therefore inspect which changes on $\mathcal{L}$ we can expect that would require an update. From the most general point of view, we can distinguish three scenarios:

- (S1) *incremental learning*: new categories or examples emerge, *i.e.*, data is added,
- (S2) *decremental learning*: known categories or examples become invalid, *i.e.*, data is deleted, or
- (S3) *data update*: representations of existing training examples change over time, *i.e.*, data is altered.

From the perspective of human learning, (S1) represents the process of knowledge expansion while learning something novel, (S2) corresponds to forgetting something known, and (S3) stands for correcting known aspects. Let us investigate a typical example for all three aspects to better familiarize ourselves with the current topic.

So far, we mainly motivated lifelong learning applications in the context of incremental learning. An application could be the search for unknown species in biological data. After a successful discovery, corresponding information needs to be added to the dataset of known animals. In contrast, decremental learning happens when an example or even a whole category can be deleted from the current knowledge set. An example from biological applications would be that a certain species became extinct. Alternatively, we might notice sensor errors in the representation of an existing example. Thus, we could decide to refuse the example for further treatment, *i.e.*, to remove it from collected labeled data. The third aspect becomes relevant if changes on representations of existing examples need to be carried out. As an example, a systematic sensor shift could have been detected and a post-processing step would correct existing errors in representations.

Contributions of This Chapter: In this chapter, we will present several techniques for GP models to tackle the previously introduced scenarios. We will mainly focus on incremental learning. Nonetheless, the presented techniques are easily extendable to the tasks of decremental learning and data updates. After a short review on existing work for incremental learning in Section 4.1, we will derive different strategies to update GP models without further assumptions on the covariance function in Section 4.2. Finally, we present in Section 4.3 how our efficient techniques for GP models with histogram intersection kernels can be extended for incremental learning. We published parts of Section 4.2 in (Lütz et al., 2011), (Lütz et al., 2013), and (Freytag et al., 2013) for the task of incremental learning for visual object categorization. Furthermore, the techniques for incremental learning of GP/HIK have been published in (Freytag et al., 2012b) and are additionally contained in the corresponding journal article (Rodner et al., 2015a).

4.1 Related Work on Incremental Learning

Techniques for efficiently updating current model parameters upon the availability of new data are required in several applications and research areas. Not surprisingly, a variety of techniques emerged over time. Solutions for a specific model family mainly differ in required assumptions on feature space Ω , output space $\mathcal{Y}$, and (hyper)parameters η . In the following, we give a short overview on popular techniques which overlap with the focus of the current thesis.

Note that incremental learning is also referred to as *online learning* when dealing with a single example at each time step. Note further that we do not cover the area of knowledge transfer in this chapter (see Section 2.3.1 for an overview). Transfer learning tackles the problem of optimally transferring parameter estimates from models for known categories to models which need to be trained for newly observed categories. Instead, we are interested in reducing the computational burden of model re-training when new data arrives continuously over time. This includes computations for models of known categories and models of newly occurring categories. However, it does not include techniques to train models of new categories more robustly. A short overview to the field of transfer learning is given in Section 2.3.1.

4.1.1 Incremental Learning for Various Models

Efficient incremental and decremental learning of exact solutions for SVM models with a fixed number of categories and a single new or deleted example was presented by Cauwenberghs and Poggio (2001). Based on the ideas of Cauwenberghs and Poggio (2001), Tax and Laskov (2003) presented incremental learning extension for the popular *support vector data description* (SVDD) model. SVDD aims at finding an enclosing hypersphere of training examples with smallest radius, and is closely related to SVMs. An extension of the well-known SVM solver SMO by Platt (1998) to incremental and decremental learning scenarios was introduced by Yeh and Darrell (2008) for one-vs-all SVM formulations where even entire classes are allowed to be added or deleted. Mensink et al. (2013) propose a nearest class mean classifier, which essentially is a nearest neighbor model (see Section 2.1.4.1) where all examples of a single category are averaged into a single data point. As shown in Mensink et al. (2013), incrementally adding new classes is possible with virtually no computations involved and thus feasible for large-scale scenarios with increasing numbers of categories. A hybrid of nearest class mean classifier and random decision forests (Breiman, 2001) was presented in (Ristin et al., 2014) together with techniques for incrementally adding new categories over time. Update rules for GMM models (see Section 2.1.4.2) have been presented by Sillito and Fisher (2007) and later used by Hospedales et al. (2013) for the task of incremental class discovery.

4.1.2 Incremental Learning for Gaussian Process Models

To apply GP models in incremental learning scenarios, drawbacks of standard GP models with respect to required memory and computation time need to be handled, *e.g.*, by computing sparse approximation of the correct model (see Section 3.1). For a specific sparse approximation, Csató and Opper (2001) presented efficient model

updates when new data becomes available. Update techniques for decompositions of GP covariance matrices have been presented by Seeger (2007) and Nguyen-Tuong et al. (2009) which are further reviewed in Section 4.2.2. Furthermore, updates for the closely related least-square regression models have been introduced by Gauss (1821, Sect. 35) for the case of adding a single examples. Extensions for adding multiple examples have been presented by Plackett (1950, Sect. 7–9).

In the following sections, we first present different techniques for model updates of GPs in their plain formulation, both for updating the inverse kernel matrix, its decomposition, or the weight vector only. Finally, we derive update rules for involved data structures of Gaussian process models with (generalized) histogram intersection kernels to allow for incremental learning even in scenarios with tens of thousands of examples and feature dimensions.

4.2 Incremental Learning with Plain GP Models

In Section 2.1.5, we have already seen that learning a Gaussian process model is dominated by computing and inverting the covariance matrix $\mathbf{K}$ from labeled data. Since required computation time and memory scale cubically and quadratically in the number of training examples, learning is an expensive process. Fortunately, this process is commonly only needed once. In the light of lifelong learning, however, new examples are likely to become available over time. Thus, expensive training would be required frequently. In this section, we will present several techniques for reducing the computational burden compared with simple training from scratch.

In contrast to the techniques presented later in Section 4.3, no assumptions on the covariance function need to be made in advance and computations are always exact. In particular, we will first present a general scheme for directly updating the inverted covariance matrix in Section 4.2.1. We then show how to alternatively update the Cholesky decomposition if accessible in Section 4.2.2. Finally, we show in Section 4.2.3 how to compute efficient updates of the weight vector α for scenarios where not the full model is required but only the predictive mean estimate,

4.2.1 Efficient Updates of the Kernel Matrix

In the most general scenario, learning a Gaussian process model happens as black box, *i.e.*, without any knowledge about the specific implementation. Consequently, the most general output of the learning stage is the inverse of the covariance matrix $\mathbf{K}^{-1}$ (how to deal with more efficient representations of the kernel matrix will be tackled in Section 4.2.2). Inverting matrices of size $N \times N$ scales with $\mathcal{O}(N^3)$. Thus, computing $\mathbf{K}^{-1}$ from scratch every time $\mathbf{K}$ changes is obviously a waste of

resources. Fortunately, the inverse of an updated matrix K' can be traced back to the old inverse K^{-1} by applying basic algebra. Let us therefore start by inspecting the matrix inversion lemma, which is also known as *Woodbury's formula* (Hager, 1989).

Lemma 4.1 (Recursive Matrix Inversion with Woodbury's Formula (Hager, 1989)).

Let A be an invertible $n \times n$ -Matrix and C an invertible $p \times p$ -Matrix with $n, p \in \mathbb{N}_+$. Further, let U and V be a $n \times p$ -Matrix and $p \times n$ -Matrix, respectively. Let us finally assume that A is transformed into a matrix A' of same size by adding UCV , i.e., $A' = A + UCV$. Then, the inverse of A' can be computed from given A^{-1} as follows:

$$A'^{-1} = (A + UCV)^{-1} \quad (4.1)$$

$$= A^{-1} - A^{-1}U (C^{-1} + VA^{-1}U)^{-1} VA^{-1} . \quad (4.2)$$

In direct consequence, we can compute the update efficiently if $p \ll n$. Note that due to the linearity of matrix-scalar-multiplications, this also holds for switched signs. Thus, the inverse can be updated similarly for $A' = A - UCV$:

$$A'^{-1} = (A - U'CV)^{-1} \quad (4.3)$$

$$= A^{-1} + A^{-1}U (C^{-1} - VA^{-1}U)^{-1} VA^{-1} . \quad (4.4)$$

The interested reader is referred to (Bernstein, 2009, pp. 117) for further derivations and analysis of special cases,

In the following derivations, C will be the identity matrix I_p which additionally makes computing C^{-1} trivial. We will now see how to apply Lemma 4.1 to all three scenarios which require model updates as introduced earlier. Note that in the entire section, we again skip explicitly adding σ_n^2 to the main diagonal of K for the sake of clear notations (see also Eq. (2.124) in Section 2.1.5.7 for an explanation).

4.2.1.1 An Update of a Single Example

Let us start with the scenario (S3) of data updates – we will see later how to transfer introduced techniques to incremental and decremental learning. Thus, we are confronted with changes in representations of known examples, *e.g.*, because we noticed a measuring error for some dimension, or the example was post-processed differently. For simplicity, we start with only a single affected example, *i.e.*, x_i changes to x'_i . An extension to the general setting with multiple examples will be presented in the subsequent Section 4.2.1.2.

With respect to a previously trained GP model, changes in one example affect all kernel values where x_i is involved. Thus, every i^{th} row and column of K needs

to be updated. For intuitive derivations, let us introduce the following variables. We will denote with e_i the i^{th} unit vector. Let further d_i be a vector defined by

$$d_i(j) = \begin{cases} \kappa(x'_i, x_j) - \kappa(x_i, x_j) & \text{if } j \neq i \\ \frac{1}{2}(\kappa(x'_i, x_j) - \kappa(x_i, x_j)) & \text{if } j = i \end{cases}, \quad (4.5)$$

with $j \in \{1, \dots, N\}$, which contains differences of kernel values for old and new representation of x_i . In addition, let us define two matrices

$$U = [d_i \ e_i] \text{ and } V = \begin{bmatrix} e_i^T \\ d_i^T \end{bmatrix}. \quad (4.6)$$

Given U and V , we can express the change from K to K' as:

$$K' = (K + UV). \quad (4.7)$$

Furthermore, the product UV offers the elegant pattern:

$$UV = \begin{bmatrix} & & d_i(1) & & & \\ & 0 & \vdots & & 0 & \\ & & d_i(i-1) & & & \\ d_i(1) & \cdots & d_i(i-1) & 2d_i(i) & d_i(i+1) & \cdots & d_i(N) \\ & & d_i(i+1) & & & & \\ & 0 & \vdots & & 0 & & \\ & & d_i(N) & & & & \end{bmatrix} \quad (4.8)$$

where 0 is a matrix of appropriate size which contains only zeros. As we can see from Eq. (4.8), $\text{rk}(UV) = 2$. Thus, we call K' a rank-2-update of K . Given matrices U and V as well as the inverse of the old covariance matrix K^{-1} , we can directly apply Eq. (4.2) to compute K'^{-1} :

$$K'^{-1} = (K + UV)^{-1} \quad (4.9)$$

$$\stackrel{(4.2)}{=} K^{-1} - K^{-1}U(I^{-1} + VK^{-1}U)^{-1}VK^{-1} \quad (4.10)$$

$$= K^{-1} - BF^{-1}VK^{-1}, \quad (4.11)$$

where we introduced $B = K^{-1}U$ and $F = I^{-1} + VK^{-1}U$. Note that F is also known as *Schur complement*. Since F is of size 2×2 , inversion can be done in constant time. Thus, the computation time is dominated by multiplying with K^{-1} and thus scales with $\mathcal{O}(N^2)$. A visualization is given in Fig. 4.1, where outputs of the initial model in Fig. 4.1a change after updating a single training example (Fig. 4.1b).

4.2.1.2 Updates of Multiple Examples

With the previous results for rank-2-updates at hand, we can generalize the setting to the scenario with updates for not only one but N_* examples. As naive baseline, we can perform an update for each affected example separately by iteratively applying the technique introduced in the previous Section 4.2.1.1. Hence, we call this an *iterative update* which requires $\mathcal{O}(N_*N^2)$ operations. However, we can also treat all affected examples simultaneously as we will see in the following. In contrast to the iterative procedure, we will refer to a joint processing as *direct update*.

Therefore, let $\mathcal{S} = \{\mathbf{x}_{s_1}, \dots, \mathbf{x}_{s_{N_*}}\} \subset \mathcal{X}$ be the set of affected examples. For each element in $\mathcal{S}$, we again define a vector with differences between old and new kernel values similar to Eq. (4.5):

$$\mathbf{d}_i(j) = \begin{cases} \kappa(\mathbf{x}'_i, \mathbf{x}_j) - \kappa(\mathbf{x}_i, \mathbf{x}_j) & \text{if } \mathbf{x}_j \notin \mathcal{S} \\ \frac{1}{2}(\kappa(\mathbf{x}'_i, \mathbf{x}_j) - \kappa(\mathbf{x}_i, \mathbf{x}_j)) & \text{if } \mathbf{x}_j \in \mathcal{S} \end{cases}, \quad (4.12)$$

with $i \in \{s_1, \dots, s_{N_*}\}$ and $j \in \{1, \dots, N\}$. In addition, we again define matrices $\mathbf{U}$ and $\mathbf{V}$:

$$\mathbf{U} = [\mathbf{d}_{s_1} \dots \mathbf{d}_{s_{N_*}} \mathbf{e}_{s_1} \dots \mathbf{e}_{s_{N_*}}] \text{ and} \quad (4.13)$$

$$\mathbf{V} = [\mathbf{e}_{s_1} \dots \mathbf{e}_{s_{N_*}} \mathbf{d}_{s_1} \dots \mathbf{d}_{s_{N_*}}]^T. \quad (4.14)$$

As in Eq. (4.7), the kernel matrix can now be obtained as $\mathbf{K}' = (\mathbf{K} + \mathbf{U}\mathbf{V})$. Notably, the product $\mathbf{U}\mathbf{V}$ has a structure similar to the previous case shown in Eq. (4.8), with kernel differences in each i^{th} row and column, $i \in \{s_1, \dots, s_{N_*}\}$, half the differences in intersections of corresponding rows and columns, and zero everywhere else. Since $\mathbf{U}$ and $\mathbf{V}$ are of rank $2N_*$, the same holds for $\mathbf{U}\mathbf{V}$. Thus, we call $\mathbf{K}'$ a rank- $2N_*$ -update of $\mathbf{K}$.

As for the update of a single example, we can again apply Woodbury's formula to update $\mathbf{K}^{-1}$ leading to the same update rule as in Eq. (4.11). For N_* examples, however, $\mathbf{F}$ is of size $2N_* \times 2N_*$. Thus, computing $\mathbf{F}^{-1}$ requires $\mathcal{O}(N_*^3)$ operations and the entire update scales with $\mathcal{O}(N_*^3 + N^2)$. Hence, in comparison with the iterative version, the direct update should be preferred if $N_* \ll N$.

4.2.1.3 Incremental Learning

Let us now transfer the previous techniques to scenarios with an incrementally increasing dataset. We again start with the special case of adding a single example and then generalize it to arbitrary numbers of new examples.

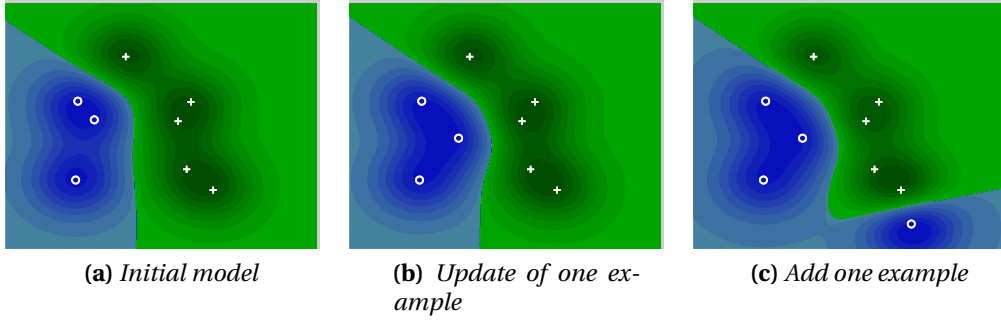


Figure 4.1: Visualizing incremental learning with plain GP models in a $D = 2$ toy example. Training examples are indicated as white markers. Predictive mean values of classes with highest score are color coded with dark colors representing large scores. Results are obtained by following a zero-mean assumption and applying an RBF kernel as covariance function. Left: outputs after initially training from $\mathcal{L}$ consisting of eight data points (Section 2.1.5.8). Middle: results after updating a single example (Section 4.2.1.1 and Section 4.2.1.2). Right: results after adding a new example (Section 4.2.1.3).

Adding One Example: As outlined before, we start with adding a new example $\mathbf{x}_*$ to the existing set of known data $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. From the definition of a Gaussian process given in Definition 2.5, it follows that the covariance matrix can be written as:

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^T & k_{**} \end{bmatrix}. \quad (4.15)$$

Since $\mathbf{K}'$ and $\mathbf{K}$ are now of different size, though, the techniques introduced previously in Section 4.2.1.1 are not directly applicable here. Let us therefore augment the old kernel matrix in each dimension by appending the corresponding unit vector:

$$\mathbf{A} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}. \quad (4.16)$$

Due to the simple block structure of $\mathbf{A}$, its inverse can be computed efficiently using block matrix inversion techniques (Duncan, 1944, Eq.(3.4)):

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{K}^{-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}. \quad (4.17)$$

Since no old kernel values are available, the previous vector of kernel differences from Eq. (4.12) needs to be adapted, too:

$$\mathbf{d}_{N+1}(j) = \begin{cases} \kappa(\mathbf{x}_*, \mathbf{x}_q) & \text{if } j \neq N+1 \\ \frac{1}{2}(\kappa(\mathbf{x}_*, \mathbf{x}_*) - 1) & \text{if } j = N+1 \end{cases} . \quad (4.18)$$

Note that the term -1 accounts for diagonal entries obtained from matrix augmentation. Finally, we again build matrices $\mathbf{U}$ and $\mathbf{V}$ of rank 2:

$$\mathbf{U} = [\mathbf{d}_{N+1} \ \mathbf{e}_{N+1}] \text{ and } \mathbf{V} = \begin{bmatrix} \mathbf{e}_{N+1}^T \\ \mathbf{d}_{N+1}^T \end{bmatrix} . \quad (4.19)$$

Thus, their product $\mathbf{UV}$ has the structure:

$$\mathbf{UV} = \begin{bmatrix} 0 & \cdots & 0 & \mathbf{k}_*(1) \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & \mathbf{k}_*(N) \\ \mathbf{k}_*(1) & \cdots & \mathbf{k}_*(N) & k_{**} - 1 \end{bmatrix} . \quad (4.20)$$

As done for the data update in Eq. (4.11), applying Woodbury's formula yields the final update rule:

$$\mathbf{K}'^{-1} = \mathbf{A}^{-1} - \mathbf{B}\mathbf{F}^{-1}\mathbf{V}\mathbf{A}^{-1} , \quad (4.21)$$

with $\mathbf{B} = \mathbf{A}^{-1}\mathbf{U}$ and $\mathbf{F} = \mathbf{I}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U}$. Since computing $\mathbf{A}^{-1}$ from $\mathbf{K}^{-1}$ requires at most $\mathcal{O}(N^2)$ operations if data needs to be copied, the overall computation time still scales with $\mathcal{O}(N^2)$. Results of model updates after adding a novel example are visualized in Fig. 4.1 for the aforementioned toy example in $D = 2$.

Adding Multiple Examples: The previous result can be generalized to adding a set of N_* new examples $\mathcal{X}_* = \{\mathbf{x}_{N+1}, \dots, \mathbf{x}_{N+N_*}\}$. Due to joint Gaussian distribution of indexing variables (see also Eq. (2.89)), the resulting covariance matrix can be written as:

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{bmatrix} . \quad (4.22)$$

With respect to iterative updates, we can apply the technique for adding a single example N_* times, which increases the resulting inverse in each step. Thus, the resulting computational cost scales with $\mathcal{O}(N_*(N + N_*)^2)$.

If we are instead interested in a direct update, we can follow the previous steps by computing the inverse of the covariance matrix augmented with N_* rows and

columns:

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{K}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N_*} \end{bmatrix}, \quad (4.23)$$

and by defining N_* vectors as:

$$\mathbf{d}_i(j) = \begin{cases} \kappa(\mathbf{x}_{N+i}, \mathbf{x}_j) & \text{if } j \neq (N+i) \\ \frac{1}{2}(\kappa(\mathbf{x}_{N+i}, \mathbf{x}_{N+i}) - 1) & \text{if } j = (N+i) \end{cases}, \quad (4.24)$$

with $i \in \{1, \dots, N_*\}$ and $j \in \{1, \dots, N + N_*\}$. Then again, we construct matrices $\mathbf{U}$ and $\mathbf{V}$ as:

$$\mathbf{U} = [\mathbf{d}_1 \dots \mathbf{d}_{N_*} \mathbf{e}_{N+1} \dots \mathbf{e}_{N+N_*}] \text{ and} \quad (4.25)$$

$$\mathbf{V} = [\mathbf{e}_{N+1} \dots \mathbf{e}_{N+N_*} \mathbf{d}_1 \dots \mathbf{d}_{N_*}]^T, \quad (4.26)$$

which are now of rank $2N_*$. Their product is a generalization of the form given in Eq. (4.20) with N_* rows and columns appended. The final update rule is again obtained by applying Woodbury's formula as in Eq. (4.21). Notably, computing $\mathbf{B}$ and inverting $\mathbf{F}$ requires $\mathcal{O}((N + N_*)^2)$ and $\mathcal{O}(N_*^3)$ operations, respectively. Thus, we obtain the final asymptotic cost of $\mathcal{O}(N_*^3 + (N + N_*)^2)$ operations. For visualization, we continue with the example in $D = 2$ and add multiple examples to $\mathcal{L}$. Results are shown in Fig. 4.2a.

Adding Examples of a New Category: As mentioned earlier, we will likely discover examples of novel categories over time in lifelong learning scenarios. Thus, we do not only require techniques for updating models of known categories, but we also need to include models for new categories efficiently. In Eq. (2.132), we introduced the flexible multi-class technique of one-vs-all for GP models. Under the assumption of shared hyperparameters (see also Eq. (2.134)), the covariance matrix was shared among all categories too and we obtained a joint formulation for the data likelihood. In direct consequence, adding a novel category model for a discovered category m_* requires the same routines as introduced previously for adding novel examples. The only additional step is the computation of a class-specific weight vector $\alpha_{m_*} = \mathbf{K}'^{-1} \mathbf{y}_{m_*}$. How to update weight vectors for known categories in closed-form is presented in Section 4.2.3.

4.2.1.4 Decremental Learning

The remaining scenario consists of removing a set $\mathcal{X}_*$ from the labeled dataset $\mathcal{L}$, *i.e.*, the examples $\mathcal{X}_* = \{\mathbf{x}_{s_1}, \dots, \mathbf{x}_{s_{N_*}}\} \in \mathcal{X}$ need to be forgotten. We can solve this task by applying the idea behind the previously introduced techniques for incremental

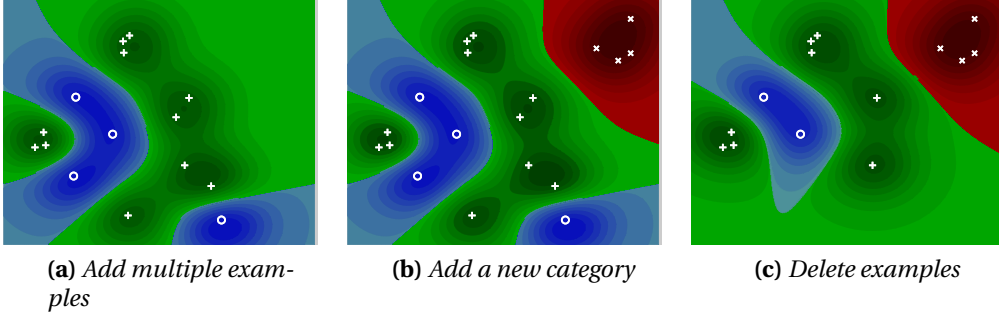


Figure 4.2: Visualizing incremental learning with plain GP models in a $D = 2$ toy example (cont.). Left: results after expanding $\mathcal{L}$ from Fig. 4.1c by adding multiple examples (Section 4.2.1.3). Middle: scores after adding an entire new category (Section 4.2.1.3). Right: scores after deleting known data (Section 4.2.1.4). See Fig. 4.2 for details on color coding and setup.

learning, but in an inverse manner. Therefore, we first define N_* vectors as:

$$\mathbf{d}_i(j) = \begin{cases} -\kappa(\mathbf{x}_i, \mathbf{x}_j) & \text{if } \mathbf{x}_j \notin \mathcal{X}_* \\ \frac{1}{2}(1 - \kappa(\mathbf{x}_i, \mathbf{x}_j)) & \text{if } \mathbf{x}_j \in \mathcal{X}_* \end{cases}, \quad (4.27)$$

with $i \in \{s_1, \dots, s_{N_*}\}$ and $j \in \{1, \dots, N\}$ and compute matrices $\mathbf{U}$ and $\mathbf{V}$ as defined in Eq. (4.25) and Eq. (4.26). W.l.o.g., let us assume that $\{s_1, \dots, s_{N_*}\} = \{N - N_* + 1, \dots, N\}$, *i.e.*, the last N_* examples shall be deleted. Then, when adding the product $\mathbf{UV}$ to $\mathbf{K}$, we obtain:

$$\mathbf{K} + \mathbf{UV} = \begin{bmatrix} \mathbf{K}' & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N_*} \end{bmatrix}. \quad (4.28)$$

Similar to the counterpart for incremental learning, applying Woodbury's formula leads to:

$$(\mathbf{K} + \mathbf{UV})^{-1} = \begin{bmatrix} \mathbf{K}'^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N_*} \end{bmatrix}, \quad (4.29)$$

We can finally obtain $\mathbf{K}'^{-1}$ by removing the N_* unit vectors in each dimension. Thus, computation times scale with $\mathcal{O}(N_*^3 + (N + N_*)^2)$, too.

4.2.2 Efficient Updates of Decompositions of the Kernel Matrix

Inverting matrices is often needed, *e.g.*, for training GP regression models. We have seen in the previous section how to efficiently update the inverse when data is

altered, when new data is added, or when known data is removed. From a numerical point of view, however, computing the inverse of a matrix is time-consuming and a numerically ill-posed problem. Thus, whenever $A^{-1} \cdot b = c$ has to be computed for a given matrix A , a known vector b , and an unknown vector c , solving the equivalent linear equation system $A \cdot c = b$ should be preferred. In the following, we give a brief overview on corresponding techniques. For a specific example, we will then present efficient update rules and thus stay within the focus of approaches for lifelong learning.

On Decompositions of the Kernel Matrix: The most common method for decomposing arbitrary matrices A is the *LU decomposition*. As the name suggests, a matrix A is decomposed into $A = L \cdot U$ such that L and U are lower and upper triangle matrices, respectively (Press et al., 1999, pp. 43). Thus, computing $A^{-1} \cdot b = c$ can be written as the solution of a linear equation system:

$$A^{-1} \cdot b = c \quad (4.30)$$

$$\Leftrightarrow (LU)^{-1} \cdot b = c \quad (4.31)$$

$$\Leftrightarrow b = (LU)c \quad (4.32)$$

Since L and U are diagonal matrices, solving the linear equation system in Eq. (4.32) is easily possible. Therefore, we first do a single forward substitution to obtain $\hat{c}$:

$$b = (LU)c \quad (4.33)$$

$$= L(Uc) \quad (4.34)$$

$$= L\hat{c} \quad (4.35)$$

followed by a single backward substitution (Press et al., 1999, pp. 48):

$$U \cdot c = \hat{c} \quad (4.36)$$

Since both steps Eq. (4.35) and Eq. (4.36) are easy to compute, the matrix vector product $A^{-1} \cdot b = c$ can be efficiently evaluated too given the previously found decomposition of A . For Gaussian process models, however, covariance matrices K are guaranteed to be positive definite and symmetric, which allows for the even more efficient *Cholesky decomposition*:

$$A = L \cdot L^T \quad (4.37)$$

where L is a lower triangular matrix. Due to the aesthetic structure in Eq. (4.37), computing a matrix's Cholesky decomposition is also referred to as “taking the square root of a matrix” (Press et al., 1999, p. 97). With the given decomposition L ,

matrix inversion can be obtained similarly to Eq. (4.32) by solving:

$$\mathbf{L} \cdot \hat{\mathbf{b}} = \mathbf{b} \quad \text{and} \quad \mathbf{L}^T \cdot \mathbf{c} = \hat{\mathbf{b}}. \quad (4.38)$$

Computing any of the previous two decompositions for a covariance matrix has the same asymptotic computation time as computing its inverse and thus scales with $\mathcal{O}(N^3)$. However, involved constants are smaller and thus lead to shorter computation times in practice. A further discussion on that topic is given by (Press et al., 1999, Sect.2.11). Besides, numerical stability of both decomposition techniques is significantly better compared to the direct matrix inversion (Press et al., 1999, p. 49).

Updating Decompositions of the Kernel Matrix: We have already seen in the previous section that the inverse of a matrix can be updated efficiently for all three scenarios introduced earlier. As we will see in the following, efficient updates are also possible for the Cholesky decomposition in incremental learning scenarios.

Let again $\mathbf{K}'$ be the new kernel matrix computed from known data $\mathcal{X}$. Furthermore, let $\mathbf{x}_*$ be the new, additionally added example which resulted in Eq. (4.15). As shown in (Bodesheim et al., 2012; Nguyen-Tuong et al., 2009; Seeger, 2007), the updated Cholesky decomposition $\mathbf{L}'$ can then be obtained from $\mathbf{L}$ by:

$$\mathbf{L}' = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \boldsymbol{\ell}_*^T & \ell_{**} \end{bmatrix}, \quad (4.39)$$

and ℓ_* and ℓ_{**} are computed by solving:

$$\mathbf{L} \cdot \boldsymbol{\ell}_* = \mathbf{k}_* \quad \text{and} \quad (4.40)$$

$$\ell_{**} = (k_{**} - \boldsymbol{\ell}_*^T \boldsymbol{\ell}_*)^{-\frac{1}{2}}. \quad (4.41)$$

Since the computation time for solving $\mathbf{L} \cdot \boldsymbol{\ell}_* = \mathbf{k}_*$ in Eq. (4.40) grows with the number of elements in $\mathbf{L}$, the overall asymptotic time for updating the Cholesky decomposition when adding a new example scales with $\mathcal{O}(N^2)$.

To the best of our knowledge, asymptotically efficient updates for decremental learning or data update are not available in general. An intuitive reason is that we can no longer assume w.l.o.g. that affected examples are located at the end of $\mathcal{X}$. Thus, updates or deletions would require updating all subsequent rows in $\mathbf{L}$. In the worst-case situation, *i.e.*, when changing or deleting the first example, we would need to re-compute the entire matrix $\mathbf{L}$ which removes any asymptotic benefit. However, if an application only requires robust techniques for training and incremental learning, the Cholesky decomposition should be the method of choice for implementing plain Gaussian process regression models. In consequence, it is

supported in the source code for active learning with GP label regression which we released along with the publications (Freytag et al., 2013, 2014c; Kading et al., 2015).

4.2.3 Efficient Updates of the Weight Vector

Updating the inverse of the covariance matrix – or a decomposition thereof – allowed us to efficiently re-compute a GP regression model in the previous sections. In some scenarios, however, updating the complete posterior distribution might be unnecessarily complex. Instead, it can be already sufficient to update only those variables which are required for classification decisions. A common scenario could consist of estimating a valid label for an unseen input, updating only the weight vector α , and checking whether the resulting classification outputs are sensible. Only then, the complete model update should take place which will usually require more operations. In (Freytag et al., 2013), we derived closed-form update formulas for this scenario and applied it in active learning tasks. The following theorem summarizes the main findings.

Theorem 4.2 (Closed Form Update of GP Regression Weights α). *Let y_* be the class label of a new example x_* . Furthermore, let $\mathbf{K} = \kappa(\mathcal{X}, \mathcal{X})$ be the kernel matrix of all training examples seen so far. Similarly, let $\mathbf{k}_* = \kappa(\mathcal{X}, \{x_*\})$ be the vector of kernel values between the new example and all training examples, and let $k_{**} = \kappa(x_*, x_*)$ be the self-similarity value of x_* . In addition, we denote with $\mathbf{y}$ the vector of labels for all training examples. We include the noise parameter σ_n^2 for model regularization within the covariance function as described in Section 2.1.5.7. As in Section 2.1.5, we follow a zero-mean assumption. Finally, let $\alpha = \mathbf{K}^{-1}\mathbf{y}$ be the weight vector for GP regression. Then, we can compute the weight vector α' after adding x_* to the training set as follows:*

$$\begin{aligned} \alpha' &= \mathbf{K}'^{-1} \cdot \begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \\ &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \underbrace{\frac{1}{\sigma_*^2}}_{\langle 1 \rangle} \underbrace{\begin{bmatrix} \mathbf{K}^{-1}\mathbf{k}_* \\ -1 \end{bmatrix}}_{\langle 2 \rangle} \underbrace{(\mu_* - y_*)}_{\langle 3 \rangle}. \end{aligned} \quad (4.42)$$

A proof is given in Appendix B.5 and is based on block-wise matrix inversion. Please note that similar derivations for linear models are also possible and known for several decades, e.g., (Gauss, 1821, Sect. 35) for adding a single example, and (Plackett, 1950, Sect. 7–9) for the general scenario of adding multiple examples.

An Intuitive Interpretation of Theorem 4.2: Interestingly, the three factors in Eq. (4.42) can be nicely interpreted. The first term $\langle 1 \rangle$ states that if a new example

is *unrelated* to the distribution of current training data, *i.e.*, the predictive variance is large, the weight $\alpha' (N + 1)$ for the example as well as the model's overall change $\Delta\alpha = \alpha' - (\alpha^T, 0)^T$ will be small. This observation is in line with the findings of Kemmler et al. (2010) who proposed using a large predictive variance as indication for possible outliers.

In a similar fashion, expression $\langle 2 \rangle$ can be interpreted as a weighted Parzen estimate. Again, if a new example is *dissimilar* with respect to currently known training examples, the overall change $\Delta\alpha$ will be small.

Finally, in $\langle 3 \rangle$ we notice that the overall change $\Delta\alpha$ and the new weight $\alpha' (N + 1)$ will be small if the new label can already be explained given the current model. Thus, the third term accounts for suppressing *redundant* information.

Based on the possibility for efficiently computing the change of the weight vector given a new example and its estimated class label, we will introduce several techniques for active learning in Section 5.4. Thereby, we can ensure to select only those examples for human annotation which contain valuable information. Here, the amount of valuable information is measured in terms of the resulting change of model parameters. In addition, we can thereby compute changes of model *outputs* via $\Delta f(x) = \Delta\alpha^T x$ as noted previously. This property will ultimately lead us to a new active learning strategy as derived in Section 5.5.

4.3 Efficient Incremental Learning with Gaussian Process Models and Histogram Intersection Kernels

In the previous section, we presented several techniques for incremental learning of Gaussian process models without further assumptions on the chosen covariance function. For a specific class of additive covariance functions for histogram features, however, we derived efficient calculations for model training, optimization of hyperparameters, model evaluations, and variance prediction in Chapter 3. The presented techniques reduced computational burden and demanded memory by orders of magnitude. Thus, it is intuitive to also investigate routines for efficient updates and increments which is the focus of this section. In detail, we have seen in Section 3.2.3.4 that the training stage for Gaussian process models with generalized histogram intersection kernels mainly consists of four steps. When a novel examples becomes available, we thus need to re-run these steps and optimally re-use the results of the previous training step.

Step 1: Updating Permutations: The first step consists of sorting features of all training examples in all dimensions. With respect to incremental learning, we

observe that novel examples can be easily integrated into existing permutations. Given the previously sorted features, a binary search can return the positions for including new values to still keep the ordering correct. Thus, for N_* new examples to add, we need to spend $\mathcal{O}(N_* D \log(N + N_*))$ operations.

Step 2: Updating the Weight Vector: For computing the updated weight vector α' in the second step, we could apply the update rules presented in Section 4.2.3. However, this would require solving $K^{-1}k_*$ (see Eq. (4.42)). Instead, we exploit the observation that the influence of novel examples on the weight vector is usually small. This observation especially holds for large numbers of known examples. Thus, using α as initialization for the ILS when computing α' can significantly reduce the number of required iterations until convergence. Note that re-using previous optima as an initialization for an optimization routine is also known as *warm start*. When referring with T'_1 to the number of ILS iterations with applied warm starting, *i.e.*, $T'_1 \ll T_1$, the resulting computation time reduces to $\mathcal{O}(T'_1 D (N + N_*))$.

Step 3: Updating Matrices A and B : As outlined in Section 3.2.3.4, matrices A and B are computed as sub-routine when running the ILS in step 2. Thus, no additional overhead is created. For incremental learning, we originally expected that both matrices can also be updated efficiently, at least if N_* is small. Unfortunately, it turns out that we indeed need to re-compute both matrices from scratch since all entries in α' are affected. So far, we are not aware of any technique for reducing the computational effort in this step which still results in exact solutions.

Step 4: Optional Updates of Look-up Table T : After updating A and B , we consequently need to update the look-up table T as well if we are interested in fast model evaluations using the quantization trick. Again, we are not aware of any efficient algorithm with less required operations than computing it from scratch. In consequence, updating the look-up table T requires $\mathcal{O}(D \max(q, (N + N_*)))$ operations.

Step 5: Optional Updates for Efficient Variance Predictions: So far, we were concerned with updating involved data structures for efficient model evaluations, *i.e.*, for computing regression and classification scores. Nonetheless, one advantage of GP models is the availability of the entire predictive distribution. In Section 3.4, we therefore additionally introduced three different techniques for estimating the predictive variance. Let us therefore shortly inspect which update routines are required when new data becomes available.

For the slow but exact version PUP (see Section 3.4.1), we notice that no updates are required since all computations are carried out at test time. Instead, the

FAPU technique presented in Section 3.4.2 requires eigenvalues and eigenvectors of the kernel matrix $\mathbf{K}$ which are affected when new data is added. While efficient techniques for updating eigenvalues and eigenvectors are likely beneficial, we found that running the Arnoldi iteration from scratch resulted in negligible computation efforts and is therefore justifiable in practice. We thus need to spent $\mathcal{O}(T_3 k D (N + N_*))$ operations, where the number T_3 of iterations until convergence is again found to be small (see also Section 3.4.2).

As an even faster approximation, we additionally introduced CAPU in Section 3.4.3. Similar to necessary updates for model evaluations, we also face a look-up table $\mathbf{A}_{\sigma_*^2}$ which needs to be updated. Interestingly, in contrast to step 2, updating and extending previous entries of $\mathbf{A}_{\sigma_*^2}$ would indeed be possible. In fact, only those matrix indices are affected where original feature values are larger than corresponding entries of new examples (see Eq. (3.44) and Eq. (3.10) with $\alpha = 1$). However, such a shift-and-update scheme would require efficient data structures for supporting this operation and would still not result in worst-case asymptotic nor experimental computation benefits. In consequence, we also compute $\mathbf{A}_{\sigma_*^2}$ from scratch.

We finally introduced q-CAPU as fasted approximation for the predictive variance. As for classification, quantization of unseen examples is the key aspect since resulting terms can mainly be precomputed and stored in a look-up table $\mathbf{T}_{\sigma_*^2}$. In direct consequence, we need to compute the entire matrix $\mathbf{T}_{\sigma_*^2}$ from scratch as soon as new data is available based on the same argumentation as in step 4. Thus, additionally $\mathcal{O}(D \max(q, (N + N_*)))$ operations are required.

Re-optimizing Hyperparameters: In Section 3.3, we have introduced efficient techniques for optimizing involved hyperparameters. On the arrival of new data, the validity of previously found optimal parameters is obviously important and a further optimization is likely required. While we are not aware of techniques for evaluating the involved steps more efficiently, we can indeed re-use the previously found optimum as initialization, *i.e.*, we can warm-start the optimization process. Furthermore, hyperparameters are often found to vary extremely smoothly when new data is added. In consequence, we can delay the time-consuming process of hyperparameter optimization until a sufficiently large number of new data is added.

Summary of Incremental Learning with GP/HIK: We analyzed in detail how to update variables which are involved in any required step of the GP/HIK techniques as introduced in Chapter 3. This analysis complements Section 4.2, where we were concerned with incremental extensions of GP models with arbitrary covariance functions. Depending on the desired operations and chosen approximations, a

variety of different steps is required and the overall computational complexity scales at least with $\mathcal{O}(D(N + N_*))$.

It should be noted that this complexity is also a lower bound for any model update which takes all previously collected data into account. To see this, let us recall that the underlying assumption of a Gaussian process as introduced in Section 2.1.5 is the joint Gaussian distribution of all known examples. Thus, without further assumptions, the joint distribution needs to be estimated upon the availability of new data. This requires the evaluation of similarities between all D -dimensional representations of all $(N + N_*)$ examples. As a final note we adhere that all techniques are also implemented in the source code which was released together with (Freytag et al., 2012b).

5 | Active Learning for Visual Object Recognition

“Which of your unlabeled images is actually worth being labeled?”

In the previous chapters, we have seen how to learn models from thousands of examples, and how to update those models when new training examples become available without expensive learning from scratch. However, with the availability of cheap consumer cameras, we notice a sheer flood of data without any possibility to reliably verify their content. As an illustrative example, consider the amount of video data uploaded to YouTube, which currently is about 100 hours of uploaded video data per minute (YouTube, 2014). Although massive amounts of *unlabeled* data are available, manually annotating all of them is no option at all. Hence, we are interested in automatically selecting a small subset which leads to approximately the same performance as using the entire dataset but with only a fraction of the annotation cost. This task is central to algorithms within the area of *active learning*, which are designed to seek for examples maximizing the estimated gain of a given information measure. If this measure is sensible, we can maximize a desired performance measure (*e.g.*, classification accuracy) for a specified labeling budget.

Contributions of This Chapter: We start in Section 5.1 with an introduction to active learning which covers theoretical concepts, several variants of active learning, as well as a comparison to the concept of active learning in educational research. An overview on established approaches for the active learning scenarios used in this thesis is given in Section 5.2. We then show in Section 5.3 how to speed up existing techniques for the choice of Gaussian process models and histogram intersection kernels, which reduces the idle time of a human annotator in the loop. In the following section 5.4, we present the criterion of estimated model changes (EMC) and extend it to non-parametric techniques. Furthermore, we introduce the new concept of estimated model output changes (EMOC) in Section 5.5 which

approximates the estimated reduction of estimated empirical risk. We will show how active learning scores for EMC and EMOC can be calculated efficiently for the choice of Gaussian process models. For the ease of explanation, we will restrict ourselves to binary classification scenarios. In the final Section 5.6, we start with transferring active learning techniques to multi-class scenarios, which we found to be often neglected in current research. We further introduce the concept of unnameable instances where annotators are likely to refuse the labeling and we extend our approaches to handle these scenarios. The main results presented in this chapter have been published in (Freytag et al., 2012b, 2013, 2014c; Kading et al., 2015).

5.1 An Introduction to Active Learning

Before we start with details of techniques within the area of active learning, let us first of all define what we can actually understand by an “active learning scenario” (Section 5.1.1). We will then give an overview on several sub-fields which developed over time and state on which sub-fields we will focus (Section 5.1.2).

5.1.1 Problem Definition

In contrast to common machine learning scenarios as introduced in Section 2.1, active learning faces the problem that annotations are assumed to be costly. In direct consequence, the labeled set $\mathcal{L}$ as introduced in Section 2.1.1 is usually small. Thus, the model $\hat{h} \in \mathbb{F}$ trained on $\mathcal{L}$ is unlikely to represent the unknown relation between inputs x and outputs y adequately. In this hopeless situation, however, we have access to an additional unlabeled dataset $\mathcal{U} \subset \Omega$ which is potentially huge, and we are allowed to select a subset for labeling.

Cost Functions and Query Functions: However, labeling examples is still a costly process, *e.g.*, by asking experts. Let us therefore specify a *cost function* which reflects the labeling costs for a given set of examples $\mathcal{S}'$:

$$\text{cost} : \mathcal{S}' \rightarrow \mathbb{R}, \quad \mathcal{S}' \subseteq \mathcal{U} . \quad (5.1)$$

In most scenarios, we will measure costs in terms of the number of labels requested, *i.e.*, $\text{cost}(\mathcal{S}') = |\mathcal{S}'|$. See Section 5.1.2.2 and Section 5.6 for different cost functions.

With the cost function at hand, we can start requesting labels until reaching the labeling budget which is limited by a predefined maximum ξ . Consequently, only the most informative examples should be selected to obtain the largest possible information gain. Thus, we face the question of how to measure or estimate *infor-*

mativeness which is at the core of the research area active learning. Let us formalize the estimation of gainable information with respect to a set of unlabeled examples $\mathfrak{S}'$. We therefore specify a second function with the same signature as cost which reflects the estimated amount of gained information when including $\mathfrak{S}'$ to $\mathfrak{L}$ after being labeled:

$$\mathcal{Q} : \mathfrak{S}' \rightarrow \mathbb{R} \quad \mathfrak{S}' \subseteq \mathfrak{U} . \quad (5.2)$$

It should be noted that $\mathcal{Q}$ is often referred to as *query function*. In contrast to the cost function, however, $\mathcal{Q}$ often requires further arguments to estimate the gainable information, *e.g.*, the current labeled dataset $\mathfrak{L}$, the remaining set of unlabeled examples $\mathfrak{U}$, or the model $\hat{h}$ trained from annotated data. For the ease of notation, though, we will skip expanding the set of inputs if not explicitly required.

Given a query function $\mathcal{Q}$ and a cost function cost , we can select a small set $\mathfrak{S}_*$ from $\mathfrak{U}$ for labeling which maximizes the estimated information gain while still staying within the specified labeling budget ξ :

$$\mathfrak{S}_* = \underset{\mathfrak{S}' \subset \mathfrak{U} \text{ with } \text{cost}(\mathfrak{S}') < \xi}{\text{argmax}} \quad \mathcal{Q}(\mathfrak{S}') . \quad (5.3)$$

The selected subset can then be labeled by an expert which results in annotations $\mathbf{y}_*$. Finally, the obtained information is collected $\mathfrak{L} \cup (\mathfrak{S}_*, \mathbf{y}_*)$ ¹ and used for re-training, *i.e.*, for updating our model's parameters.

In summary, an active learning scenario can be formally defined as five-tuple:

$$\mathcal{AL} = (\mathbb{F}, \mathfrak{L}, \mathfrak{U}, \mathcal{Q}(\cdot), \text{cost}(\cdot)) \quad (5.4)$$

consisting of a specified model family $\mathbb{F}$, a small initially labeled dataset $\mathfrak{L}$, an additionally unlabeled set $\mathfrak{U}$, and two functions $\mathcal{Q}$ and cost for estimating the gainable information and resulting annotation costs for chosen subsets of $\mathfrak{U}$.

Useful Changes in Notation: Before we move on, let us introduce and re-define notations. Thereby, we can keep the following formulas in a readable layout while maintaining a notation which is as simple and clean as possible.

First of all, let us introduce an abbreviation for the model which is trained from currently known annotated data $\mathfrak{L}$. According to our notation introduced in Section 2.1.1, this model would be denoted with $\hat{h}_{\mathfrak{L}}$. In the following, we will skip the suffix and use the short-hand $\hat{h}$ if the meaning is clear from the context.

Similarly, let us introduce an abbreviation for the model which is trained on the union of the old dataset $\mathfrak{L}$ and the current subset of investigation $\mathfrak{S}'$. Let $\mathbf{y}'$ denote the current estimation of labels for elements of $\mathfrak{S}'$. According to our previous

¹For the sake of a simple notation, we will denote the combined dataset as $\mathfrak{L} \cup (\mathfrak{S}_*, \mathbf{y}_*)$ although we actually combine sets of ordered pairs as introduced in Section 2.1.1.

notation, the resulting model would be denoted with $\hat{h}_{(\mathcal{L} \cup (\mathfrak{S}', y'))}$. Here, we will use the short-hand $\hat{h}'$ instead.

Finally, let us re-define symbols $\hat{h}$ and f . In Section 2.1.1, we introduced $\hat{h}$ to refer to the actual relation learned from training data, whereas f referred to a continuous intermediate function. For the rest of this chapter, we will not use the symbol $\hat{h}$ any longer. Instead, we will use the symbol f for the learned model and, thus, not for an intermediate function. This will allow for better comparability with our previous publications (Freytag et al., 2012b, 2013, 2014c; Kading et al., 2015).

In summary, we will refer to $\hat{h}_{\mathcal{L}}$ and $\hat{h}_{(\mathcal{L} \cup (\mathfrak{S}', y'))}$ with f and f' , respectively.

Active Learning – Seeking for the Smallest Risk: We have seen in Section 2.1.2.2 how to optimize model parameters given annotated training data and an objective function, *e.g.*, the classification error on hold-out validation sets. Independent of the chosen objective, our general goal is to obtain highly accurate models. Since the focus of this thesis is mainly on classification scenarios, we thus seek for models with high classification accuracy or, similarly, with small induced risk (see Section 2.1.2.1). Given the previous definition of active learning, we can transfer the principle of expected risk minimization (see Section 2.1.2.1), and rely on the decrease in expected risk as introduced in Eq. (2.1) to estimate the amount of information that is gained by $\mathfrak{S}'$:

$$\Delta \mathcal{R} (\mathfrak{S}', y', \mathcal{L}) = \mathcal{R} (f) - \mathcal{R} (f') \quad (5.5)$$

$$= \int_{\mathcal{Y}} \int_{\Omega} p(\mathbb{X} = x, \mathbb{Y} = y) \cdot \dots \quad (5.6)$$

$$(\mathcal{L} (y, f (x)) - \mathcal{L} (y, f' (x))) \, dx \, dy \, .$$

Thereby, specifying the change of risk in Eq. (5.5) as query function leads to selecting a subset $\mathfrak{S}_* \subset \mathcal{U}$ with largest decrease in expected risk:

$$\mathfrak{S}_* = \underset{\mathfrak{S}' \subset \mathcal{U} \text{ with } \text{cost}(\mathfrak{S}') < \xi}{\text{argmax}} \, \Delta \mathcal{R} (\mathfrak{S}', y', \mathcal{L}) \, . \quad (5.7)$$

If we again consider a finite collection of labeled data, we can approximate the estimated risk with its empirical counterpart as introduced in Eq. (2.9). Thus, the decrease in expected risk in Eq. (5.5) translates to the decrease in empirical risk:

$$\Delta \mathcal{R}_{\text{emp}} (\mathfrak{S}', y', \mathcal{L}) = \mathcal{R}_{\text{emp}} (f) - \mathcal{R}_{\text{emp}} (f') \quad (5.8)$$

$$= \frac{1}{|\mathcal{L}|} \sum_{(x,y) \in \mathcal{L}} (\mathcal{L} (y, f (x)) - \mathcal{L} (y, f' (x))) \, .$$

Finally, the resulting optimization translates to selecting a subset with affordable labeling cost which maximizes the decrease in empirical risk:

$$\mathfrak{S}_* = \underset{\mathfrak{S}' \subset \mathcal{U} \text{ with } \text{cost}(\mathfrak{S}') < \xi}{\text{argmax}} \Delta \mathcal{R}_{\text{emp}}(\mathfrak{S}', \mathbf{y}', \mathcal{L}) . \quad (5.9)$$

Although appealing from a theoretical point of view, several limitations prevent us from applying Eq. (5.9) in practice as we will see in the following.

Drawbacks in Optimal Active Learning: Starting from the criterion of maximizing the decrease of empirical risk as introduced in Eq. (5.9), we observe three drawbacks preventing us from implementing an optimal active learner:

- (P1) the labeled set $\mathcal{L}$ is too small for being representative², thus, replacing the true risk $\mathcal{R}$ in Eq. (5.5) by its empirical counterpart $\mathcal{R}_{\text{emp}}$ to obtain Eq. (5.8) is cumbersome,
- (P2) exponentially many subsets $\mathfrak{S}' \subset \mathcal{U}$ in Eq. (5.9) make optimization computationally intractable,
- (P3) the absence of labels $\mathbf{y}'$ for examples in $\mathfrak{S}'$, *i.e.*, a required input for computing $\Delta \mathcal{R}_{\text{emp}}$, makes evaluating Eq. (5.9) impossible.

The first drawback (P1) can be tackled by either leveraging the unlabeled set $\mathcal{U}$ or by relying on information measures not requiring a representative labeled set, and we will see possible solutions for both variants in this chapter. With respect to computational tractability (P2), a common approximation is to perform an iterative optimization by selecting one example at a time, *i.e.*, $|\mathfrak{S}'| = 1$, which is also known as *myopic* learning. The third problem (P3) of missing labels, however, is among the most crucial aspects and prevents from reliably judging the informativeness of a selected subset with respect to Eq. (5.9). In this chapter, we are therefore interested in suitable proxies of Eq. (5.9). Before we move on to our own contributions in Sections 5.3 to 5.6, we will first introduce characteristics of common active learning scenarios (Section 5.1.2) and review established approaches (Section 5.2).

5.1.2 Variants of Active Learning Scenarios

5.1.2.1 On the Availability of $\mathcal{U}$

Depending on the availability of unlabeled examples, we can distinguish between three common scenarios for active learning.

²If a representative labeled dataset was available, we would not need to query new examples.

Pool-based Active Learning: The most common variant is *pool-based* active learning, where the set $\mathcal{U}$ of unlabeled examples is fixed from the beginning, and examples can be selected for annotation until a given budget is reached. Typical examples are annotations of huge data collections from web search results (Vijayanarasimhan et al., 2014), or the annotation of available but unclassified medical data (Liu, 2004).

In this thesis, we will purely focus on pool-based scenarios for two reasons: first, we believe that the most intuitive way for incorporating active learning into lifelong learning systems is to collect data over a certain time period (*e.g.*, a day) and then pick a small set of them to be handed over to a user. Besides, the majority of active learning research done in our community applies to this setting which allows for rich and fair comparisons.

Stream-based Active Learning: In *stream-based* active learning, $\mathcal{U}$ is not fixed. Instead, unlabeled data is available as ongoing data stream where new examples arrive consecutively, as it is the case for monitoring systems. Hence, rather than selecting a subset of $\mathcal{U}$, the query function has to decide at each time step t whether the currently available example has to be labeled, *i.e.*, $|\mathcal{U}_t| = 1$. We refer to (Loy et al., 2010) and (Loy et al., 2012) as examples for stream-based active learning in event recognition and image classification scenarios.

Query Synthesis Active Learning: Finally, in *query synthesis* active learning, new examples can be created on demand, which results in the extreme case of $\mathcal{U} = \Omega$. This setting naturally corresponds to automated experimental design as presented in (King et al., 2009). However, for computer vision applications, a synthesis of query data is often difficult since a one-to-one-mapping between features and images is usually not available. In addition, synthesized images are hardly human interpretable (Lang and Baum, 1992).

5.1.2.2 Further Variants of Active Learning

Depending on the current application, several aspects are worth being considered. In the following, we list some of them and position our contributions with respect to those aspects.

Different Sizes of $\mathcal{S}_*$: In the definition given in Eq. (5.4), the size of $\mathcal{S}_*$ is not constrained *per se*. A standard assumption is the previously mentioned myopic learning, *i.e.*, $|\mathcal{S}_*| = 1$. Thus, only a single instance is queried in every iteration, *e.g.*, by greedily reducing the estimated empirical risk (see Section 5.2.4) or a proxy thereof. However, in some scenarios an oracle is able to label multiple instances simultaneously, known as non-myopic or batch active learning (Brinker, 2003). In

this thesis, we focus on myopic active learning only, since labeling an example by asking an expert is assumed to be an operation too costly to be done set-wise.

Varying Annotations $\mathcal{Y}$ and Costs_{cost}: So far, we only dealt with requesting a single label for each selected example. When dealing with images, though, we can exhibit their spatial structure and request more sophisticated types of annotation. In recent work, users have been asked to provide bounding boxes in order to help localizing the object of interest (Vijayanarasimhan and Grauman, 2011b), to segment the whole image (Vijayanarasimhan and Grauman, 2009), or to give feedback by naming attributes being present in the image (Duan et al., 2012; Parikh and Grauman, 2011). As a result, annotation costs can vary significantly between users and query types, and a cost estimation can be included directly into the query function to optimize which question is asked for which image (Vijayanarasimhan and Grauman, 2011a). In this thesis, however, we mainly assume all costs to be equal and we request a single class label for an entire image. An exception is Section 5.6 where we transfer the introduced techniques to label object proposals provided from an unsupervised segmentation.

Working with Imperfect Oracles: For simple annotation tasks, crowdsourcing was recently found to be an effective alternative to asking experts, and Amazon Mechanical Turk was heavily used (*e.g.*, the ImageNet dataset (Deng et al., 2009) was annotated using Amazon Mechanical Turk workers). However, while being cheap and easily scalable, crowdsourcing comes with its own problems. One of the most important aspects here is the chance of noisy labels, *i.e.*, that the assumed oracle is in fact imperfect. Approaches have been proposed which simply request labeling an instance several times (Vijayanarasimhan and Grauman, 2011b) or which learn the reliability of individual annotators (Long et al., 2013) to reduce or estimate the influence of label noise. Here, we instead assume a reliable oracle, motivated by the observation that parents would (hopefully) pass only reliable information to their lifelong learning children.

Working with Unnameable Instances: Previous variants assumed that an annotator always provides an answer, *e.g.*, the requested label of a given image. From our experience, this assumption does not hold in practice. Instead, likely reject the requested labeling when the example does not belong to a valid category (*e.g.*, lens artifacts, motion blur, or segmentations covering parts of multiple objects) or the example is categorical, but unknown to the annotator or unrelated to the current labeling task. Fang and Zhu (2012) referred to these instances as “blind spots” of an annotator, and the probability of getting the requested example rejected needs to be carefully modeled. While this is unlikely to happen in binary scenarios (*i.e.*,

the decision whether or not the current example belongs to a specific category is usually easy), it becomes highly relevant in multi-class scenarios with special focus on lifelong learning. We therefore present an active learning technique in Section 5.6 which allows annotators to reject examples during labeling and which models possibilities of receiving answers like *“I can not name the object.”*, *“This is not a real object.”*, or *“This object is not of interest.”*

5.1.2.3 A Side Note on Active Learning in Educational Research

It is insightful to note that active learning also refers to a technique in educational research which is well known for several decades. There, the concept of active learning aims at actively integrating students into the learning process:

“Students learn both passively and actively. Passive learning takes place when students [...] do not directly participate in the learning process. Active learning is more likely to take place when students are doing something besides listening [...].”

(Ryan and Martens, 1989, p.29)

Although this concept is similar to our understanding of active learning, we are not interested in active learning from an educational point in the remaining part of this thesis. Instead, we refer the interested reader to (Bonwell and Eison, 1991) for more information about this topic.

Furthermore, active learning as defined previously shows several parallels to latest findings in educational and learning theory. According to prominent fields such as *constructivism* (von Glasersfeld, 1989), observers (learners) construct their own (subjective) reality based on individual prior knowledge and interactions, rather than accepting a “universal truth” transmitted by an educator. For successful education, learning should therefore happen actively:

“[...]the role of the child necessarily changes too. It becomes active instead of passive, the child becomes the questioner and experimenter.”

(Neubert, 2003)

and especially focused on the current problem context:

“Learning begins when learners use and expand their constructive agencies to solve problems and create meanings in the concrete situations they find themselves in.”

(Neubert, 2003)

In consequence, transferring these findings to principles of machine learning seems intuitive. We argue that this directly leads to lifelong learning systems which are able to actively expand their current knowledge (*i.e.*, the ability to discover new clusters and classes), and at the same time focus on the current situation (*i.e.*, guiding question by considering data observed so far and models trained thereof).

5.2 Related Work on Active Learning

The underlying goal shared by all active learning strategies is an improvement in accuracy with as less annotation costs as possible, or similarly, a fast error reduction. However, determining a system's accuracy is usually not possible, and empirical estimates based on validation and test sets are used as proxies thereof (see Section 2.1.2.2). In the following, we review active learning techniques which optimize what is assumed to be important to quickly optimize accuracy, or which directly aim at reducing empirical error estimates. For each group of approaches, we will point to open issues which will lead the way to our techniques introduced in the remaining sections.

5.2.1 Rapid Exploration of Feature Space

A surprisingly large number of approaches for active learning neglects labels obtained so far as well as the currently learned model, and instead solely focus on *unseen* regions in feature space. Thus, techniques in this field are usually called *explorative*, whereas approaches in subsequent fields are termed *exploitative* for their use of current class labels and model estimates.

For selecting unlabeled examples, the technique presumably most intuitive is to pick those examples that are maximally different from all previously seen data, *i.e.*, to determine *novel* examples. The underlying assumption is that covering the entire feature space as fast as possible is likely to reduce errors in all regions simultaneously, and requesting redundant examples is avoided. Baram et al. (2004) propose to select examples being maximally distant in kernel space, whereas Seo et al. (2000) and Kapoor et al. (2010) seek examples with high Gaussian process predictive variance scores.

Note that there is a close relationship between active learning techniques for rapid exploration and approaches for *novelty detection* (see (Bodesheim, 2016) for a comprehensive review on this area). In fact, we successfully applied several approaches in both settings, *e.g.*, (Bodesheim et al., 2013a; Freytag et al., 2012b). For finding novel clusters rather than novel examples, He and Carbonell (2007) present a technique based on probability approximations by nearest neighbor ratios. Pelleg and Moore (2004) model input distributions as a GMM and use

an adapted likelihood criterion to identify useful anomalies. Ebert et al. (2012) propose to build a weighted graph by connecting every unlabeled example with its nearest neighbors in $\mathcal{U}$ and to select highly connected nodes which are additionally unsimilar to previously labeled examples. Demir and Bruzzone (2014) select in a first step uncertain examples, which then are clustered and cluster centroids serve as queries.

Unfortunately, a common drawback of rapid exploration is the tendency of querying outliers unrelated to the currently faced task. We will observe this behavior qualitatively in Section 5.3.1 as well as quantitatively in experimental evaluations presented in Section 7.3.

5.2.2 Highest Model Uncertainty

Seeking examples with highest uncertainty in model decisions is similar in spirit to the previous setting but with the additional advantage of taking known class labels into account. Running in a supervised manner, techniques in this area aim at fast refinements of current decision boundaries.

Probably the first work in this area is given by Lewis and Gale (1994) for text classification with naive Bayes classifiers. For binary SVMs, Schohn and Cohn (2000), Campbell et al. (2000), and Tong and Koller (2002) propose to select examples closest to the SVM margin (see Section 2.1.4.3), and Vijayanarasimhan et al. (2014) present an approximation for working within huge unlabeled datasets resulting in sub-linear query time. Interestingly, strong theoretical results support this approach, *e.g.*, for SVMs it can be shown that querying examples closest to the current decision boundary reduces the set of hypothesis consistent with training data as fast as possible (Tong and Koller, 2002). Similar in spirit is the query-by-committee strategy introduced by Freund et al. (1997) which selects an example for labeling if two models disagree on its label although both are consistent with previously known training data. Kapoor et al. (2010) transferred the minimum margin technique to Gaussian process models and extended it by considering the full posterior distribution available. Jain and Kapoor (2009) similarly select examples in a min-max probability scheme closely related to entropy scores. Joshi et al. (2009) propose an approximation of uncertainty sampling for multi-class scenarios by considering small differences between most likely classes only.

All of these approaches have in common, however, that no implications on future decisions are explicitly taken into account.

5.2.3 Largest Estimated Model Change

Similar in spirit is the preference of examples which are likely to change the current model most drastically after being labeled and added to the training set. The

underlying intuition is to prevent the selection of redundant examples which would not contribute new information to the current model.

For parametric models, this criterion translates to estimating the change of the model's parameters. By computing the gradient of the objective function, changes of model parameters for every unlabeled example under all possible label assumptions can be efficiently estimated. A realization for multiple-instance logistic regression models was shown by Settles et al. (2008). A similar approach for linear regression was presented by Cai et al. (2013). Realizations for linear SVM models are derived in (Cai et al., 2014). Further introductions can be found in (Settles, 2009, Sect. 3.3). We present a new strategy for the choice of Gaussian process label regression models in Section 5.4.

As for the previous approaches, an implicit drawback is the missing connection between large model changes and the estimated reduction of errors.

5.2.4 Largest Estimated Reduction of Estimated Error or Entropy

An ad-hoc solution for tackling the first problem (P1) mentioned in Section 5.1.1 is to additionally rely on collected, unlabeled examples. To this end, let us assume that some unknown magical process provided us with labels for the previously unlabeled data such that we can combine both sets into one large and representative dataset $\mathfrak{D} = \mathfrak{L} \cup \mathfrak{U}$. In consequence, replacing the true risk $\mathcal{R}$ with its empirical counterpart $\mathcal{R}_{\text{emp}}$ in Eq. (5.7) would become well justifiable. The decrease in empirical risk as shown in Eq. (5.8) thus translates to:

$$\Delta \mathcal{R}_{\text{emp}}(\mathfrak{S}', \mathbf{y}', \mathfrak{L}, \mathfrak{U}) = \frac{1}{|\mathfrak{L} \cup \mathfrak{U}|} \sum_{(\mathbf{x}, y) \in \mathfrak{L} \cup \mathfrak{U}} (\mathcal{L}(y, f(\mathbf{x})) - \mathcal{L}(y, f'(\mathbf{x}))) \quad . \quad (5.10)$$

While this looks beneficial, we have no magical process for providing labels of examples in $\mathfrak{U}$ available in practice. Thus, we can not actually compute the empirical risk or the change thereof. However, we can *estimate* missing labels using outputs of the current model, which results in computing the *estimated empirical risk* $\mathcal{R}_{\mathbb{E}}$ (compare with $\mathcal{R}_{\text{emp}}$ in Eq. (2.9)):

$$\mathcal{R}_{\mathbb{E}}(f) = \frac{1}{|\mathfrak{L} \cup \mathfrak{U}|} \sum_{\mathbf{x} \in \mathfrak{L} \cup \mathfrak{U}} \sum_{y \in \mathcal{Y}} \mathcal{L}(y, f(\mathbf{x})) p(y|f(\mathbf{x})) \quad . \quad (5.11)$$

Note that we used the notation of $\mathfrak{L} \cup \mathfrak{U} \subset \Omega$ for referring to the joint set of examples, although $\mathfrak{L}$ actually contains ordered pairs of examples and corresponding labels. Within this chapter, though, we will stick to the short version of $\mathbf{x} \in \mathfrak{L} \cup \mathfrak{U}$ for the ease of readability.

By incorporating the estimation of labels as in Eq. (5.11) into the change of empirical risk $\Delta \mathcal{R}_{\text{emp}}$ shown in Eq. (5.8), we obtain the *estimated change of estimated empirical risk*:

$$\Delta \mathcal{R}_{\mathbb{E}}(\mathbf{x}') = \sum_{y' \in \mathcal{Y}} p(y' | f(\mathbf{x}')) \cdot \frac{1}{|\mathcal{L} \cup \mathcal{U}|} \cdot \dots \quad (5.12)$$

$$\sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} \sum_{y \in \mathcal{Y}} \left(\mathcal{L}(y, f(\mathbf{x})) p(y | f(\mathbf{x})) - \mathcal{L}(y, f'(\mathbf{x})) p(y | f'(\mathbf{x})) \right) .$$

Note that the first occurrence of *estimate* refers to the estimation of labels for the current query $p(y' | f(\mathbf{x}'))$. The second occurrence instead accounts for the estimation of labels of unlabeled examples $p(y | f(\mathbf{x}))$ and $p(y | f'(\mathbf{x}))$ when evaluating the risk.

If we use $\Delta \mathcal{R}_{\mathbb{E}}$ as query function, we seek for the unlabeled example with highest estimated decrease in estimated empirical risk. Depending on the chosen loss function $\mathcal{L}$, this finally results in estimated loss reduction for the 0/1-loss (Freund et al., 1997; Mac Aodha et al., 2014; Roy and McCallum, 2001; Schohn and Cohn, 2000; Settles, 2009), estimated entropy minimization for the log-loss (Kovashka et al., 2011; Li and Guo, 2014; Roy and McCallum, 2001; Settles, 2009; Vijayanarasimhan and Grauman, 2011a), or reduction of the mean-square error for the L_2 -loss (Cohn, 1996; Seo et al., 2000). This general framework was further reviewed by Settles (2009) and used for requesting attributes and class labels by Kovashka et al. (2011). Mac Aodha et al. (2014) show how to reduce computation times by evaluating centers of a hierarchical clustering only. An application for multiple-instance learning is presented by Vijayanarasimhan and Grauman (2011a), and active learning of scene classifiers is shown in (Li and Guo, 2014).

Unfortunately, computing the estimated change of estimated empirical risk requires reliable estimates of class labels, which is an ill-posed problem when labeled data is rare and thus prone to errors.

5.2.4.1 Combination of Different Strategies

Which query strategy to prefer can heavily differ between applications, since data distributions and complexity of problems vary. In addition, different aspects might be needed to different degrees during different phases of learning, *e.g.*, an early exploration of feature space can help to locate unknown clusters, whereas refinement of classification boundaries might be more relevant later on. Thus, an active learning system should be able to combine a set of possibly diverse strategies. Furthermore, it would be beneficial if the system could chose among these complementary strategies based on the current state of learning.

Baram et al. (2004) propose a multi-armed bandit formulation to select among several possible query strategies. Osugi et al. (2005) flip a biased coin to choose among two complementary query techniques and coin probabilities are updated depending on the change of the model after re-training. Cebron and Berthold (2009) introduce a linear combination scheme for explorative and exploitative methods. They experiment with combining the data density criterion (explorative) with the entropy criterion (exploitative) and investigate several options for varying the combination parameter. Hospedales et al. (2013) present how to adaptively switch between queries based on a generative model and queries based on a discriminative model. The actual choice is done depending on the outcome of a carefully designed reward function measuring success in detecting novel categories and success in increasing generalization abilities. Haines and Xiang (2011) introduce the concept of misclassification probability based on Dirichlet process models, which takes both the likelihood of an example stemming from a new class as well as the current classifier uncertainty into account. Ebert et al. (2012) formulate the selection of active learning techniques as a reinforcement learning task.

A comparison of several combination techniques for the task of active learning for image classification scenarios is given in (Sickert, 2012). In this thesis, we instead focus on single strategies for approximating the expected information gain.

5.3 Efficient Active Learning with Gaussian Process Models

After investigation different aspects of active learning techniques in the previous section, we will now familiarize ourselves with active learning approaches for Gaussian process models. As mentioned before, we will entirely focus on myopic active learning (see Section 5.1.2.2) by requesting annotations for only a single example at each iteration. We first review existing techniques which are known from literature in Section 5.3.1. Then, we show in Section 5.3.2 how to exploit techniques for efficient computations as presented in Chapter 3. This section is mainly based on our findings presented in (Freytag et al., 2012b).

5.3.1 Existing Strategies for Active Learning with Gaussian Process Models

For the choice of Gaussian process models, Kapoor et al. (2010) presented three criteria for active learning derived from the predictive distribution: using the predictive mean, the predictive variance, or a combination of both. In the style of Tong and Koller (2002), selecting examples can be done based on smallest absolute mean

(see Eq. (2.99)):

$$\mathcal{Q}_{\text{GP-}\mu}(\mathbf{x}') = |\mu(\mathbf{x}')|, \quad \mathbf{x}_* = \underset{\mathbf{x}' \in \mathcal{U}}{\operatorname{argmin}} \mathcal{Q}_{\text{GP-}\mu}(\mathbf{x}') \quad , \quad (5.13)$$

which aims at selecting examples closest to the current decision boundary, and is thus a member of the maximum-uncertainty framework (see Section 5.2.2). In contrast to $\mathcal{Q}_{\text{GP-}\mu}$, queries based on a high predictive variance (see Eq. (2.100)):

$$\mathcal{Q}_{\text{GP-}\sigma^2}(\mathbf{x}') = \sigma^2(\mathbf{x}') \quad , \quad \mathbf{x}_* = \underset{\mathbf{x}' \in \mathcal{U}}{\operatorname{argmax}} \mathcal{Q}_{\text{GP-}\sigma^2}(\mathbf{x}') \quad (5.14)$$

are purely explorative (see Section 5.2.1) and can be interpreted as selecting examples with low weighted Parzen density estimates (see Section 2.1.5). As a third technique, Kapoor et al. (2010) proposed a combination of the previous two approaches which they call *uncertainty*:

$$\mathcal{Q}_{\text{GP-unc}}(\mathbf{x}') = \frac{|\mu(\mathbf{x}')|}{\sqrt{\sigma^2(\mathbf{x}')}}, \quad \mathbf{x}_* = \underset{\mathbf{x}' \in \mathcal{U}}{\operatorname{argmin}} \mathcal{Q}_{\text{GP-unc}}(\mathbf{x}') \quad . \quad (5.15)$$

The motivation for applying $\mathcal{Q}_{\text{GP-unc}}$ is to obtain a query function similar to the minimum margin approach suitable for SVMs (Tong and Koller, 2002) but with the additional consideration of the classifier uncertainty in terms of the predictive variance. In empirical evaluations, Kapoor et al. (2010) showed that especially $\mathcal{Q}_{\text{GP-}\mu}$ and $\mathcal{Q}_{\text{GP-unc}}$ can lead to impressive performance gains compared to randomly querying new examples. Interestingly, Jain and Kapoor (2009) argued that $\mathcal{Q}_{\text{GP-unc}}$ can also lead to a loss of performance compared to random selection, especially for large number of classes. We will compare all techniques in Section 7.3 for the task of active learning of category detectors on established computer vision datasets.

In Fig. 5.1, we visualized the behavior of all three techniques in a simple $D = 2$ toy example. An RBF kernel with fixed bandwidth serves as similarity measure, and a binary GP model is trained for the two classes, with one known example per class (black cross and black diamond, respectively). First of all, the difference between the explorative technique $\mathcal{Q}_{\text{GP-}\sigma^2}$ and the exploitative methods $\mathcal{Q}_{\text{GP-}\mu}$ and $\mathcal{Q}_{\text{GP-unc}}$ is clearly visible. We can already guess from this simple example that methods too explorative are likely to query mainly outliers (see Fig. 5.1b). On the other hand, methods purely exploitative might fail in discovering new clusters, *e.g.*, the two clusters in the upper left part of space. Thus, a careful trade-off between exploration and exploitation intuitively seems useful. How to tackle this issue by explicit combinations of several techniques was reviewed in Section 5.2.4.1. In the following Sections 5.4 to 5.6 of this chapter, we will additionally introduce two techniques which implicitly combine both aspects.

As a final note, it is interesting to see that also the exploitative methods $\mathcal{Q}_{\text{GP-}\mu}$

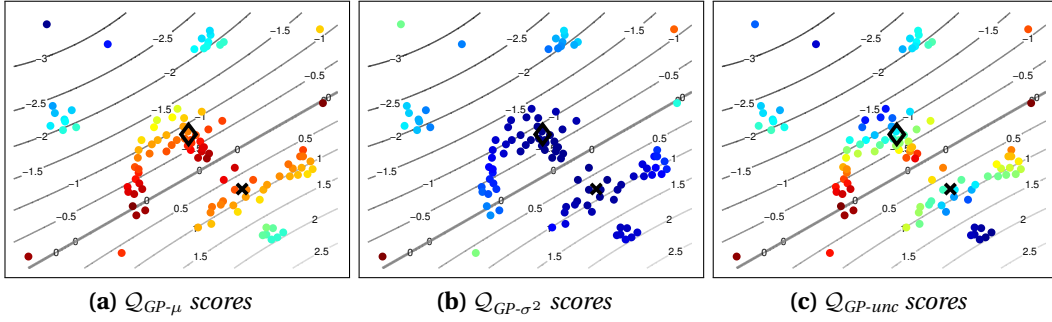


Figure 5.1: Visualization of active learning scores for three query strategies based on Gaussian process models as introduced by Kapoor et al. (2010). Two classes with a single known example are used for model training (black diamond and cross), and remaining dots serve as unlabeled examples to choose from. Levels of constant decision scores are plotted and the current decision boundary is additionally indicated as thick gray line. Active learning scores derived by each of the three techniques are visualized using Matlab's *Jet colormap*, and warm colors indicate examples preferred by the respective technique. Figure is best viewed in color.

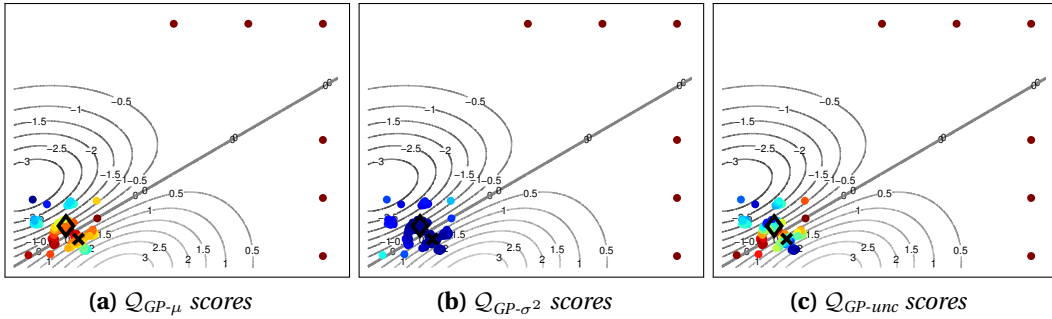


Figure 5.2: Deriving simple query techniques from a Gaussian process' predictive distribution as suggested by Kapoor et al. (2010) is likely prone to outliers. While the setup and visualization is identical to Fig. 5.1, several outliers are added far off the main data distribution (top right). All methods are extremely inclined to query these examples, although they do not contain valuable information with respect to the current task. It is strongly recommended to view the figure in color.

and $\mathcal{Q}_{\text{GP-unc}}$ are tempted to query outliers. In Fig. 5.2, we visualized this observation in a small experiment. To this end, we added several outliers far off the main data distribution while the remaining setup and the majority of data is chosen identical to the scenario in Fig. 5.1. As we can see, all three techniques favor outliers over remaining unlabeled data points as queries.

Clearly, this behavior is counterintuitive at first glance. Similarly, it is in contrast to the motivation of preferring small distances to the classification boundary as stated by Kapoor et al. (2010). Nonetheless, when inspecting Fig. 5.2, we note that all outliers result in classification scores close to zero (see levels of equals classification scores). More precisely, when kernel values between training and test examples are approaching zero for highly distant examples, no information can be transferred. Thus, the output of a GP regression model reduces to the prior terms.

In our case, this results in mean values close to zero due to the zero-mean assumption (see Section 2.1.5.1). Thus, outliers are likely being queried which is a waste of budget in many scenarios. In the following sections, we therefore introduce techniques which are unaffected by outliers and thus allow for querying examples that are truly informative for the current task.

5.3.2 Fast Active Learning with Gaussian Process Models and Histogram Intersection Kernels

Given the previously presented query strategies based on Gaussian process models, it is straightforward to see that efficient computations presented in Chapter 3 can be easily exploited. Note that in the rest of this thesis, we assume the number of queries being the limiting factor and not the time budget *per se*. This assumption is sensible in many scenarios, since annotators usually tend to be annoyed by a tedious labeling process with lots of labels requested. Thus, the number of queries should be kept as small as possible. As an example, note that Amazon Mechanical Turk workers are often paid for annotating a number of requested images as good as possible, rather than annotating images for one hour in a row.

However, if an expert is only available for a predefined time span, we indeed have a limiting time budget. By using efficient computations of Chapter 3, computing scores for unlabeled examples will be significantly faster. Thereby, we can ask for more queries within the same time since computation times for active selection are decreased by orders of magnitudes. As a byproduct, also model updates can be carried out highly efficient while simultaneously breaking common memory limitations of non-parametric vanilla GP.

In detail, computing the scores for $\mathcal{Q}_{\text{GP-}\mu}$ as introduced in Eq. (5.13) can be realized using Eq. (3.10) of Section 3.2.2.1. For the explorative technique $\mathcal{Q}_{\text{GP-}\sigma^2}$ in Eq. (5.14), any of our introduced approximations can be applied (PUP of Sec-

tion 3.4.1, FAPU of Section 3.4.2, Eq. (3.38), or CAPU and q-CAPU of Section 3.4.3, Eq. (3.44)). The combined criterion $\mathcal{Q}_{\text{GP-unc}}$ in Eq. (5.15) benefits from both aspects. Furthermore, model updates can be carried out after receiving annotation feedback using the incremental learning approach as presented in Section 4.3.

5.4 Active Learning Based on Estimated Model Changes

In the previous section, we tackled active learning tasks by relating the estimated informativeness of unlabeled examples with the first and second moments of a Gaussian process model's predictive distribution. While the resulting query functions are intuitively useful and can be efficiently computed with our techniques introduced in Section 5.3.2, they are entirely focused on GP models. In this section, we will therefore introduce a more general active learning strategy which we published in (Freytag et al., 2013) and which is based on estimated model changes as shortly reviewed in Section 5.2.3. For the previous choice of GP models, we present how the resulting technique can again be computed efficiently.

5.4.1 Estimated Model Changes (EMC)

At the chapter's beginning, we motivated active learning from the perspective of selecting examples with strongest reduction of a model's expected risk. Since computing the expected risk was cumbersome due to several aspects (see Section 5.1.1), we were concerned with intuitive proxies thereof in the last section. However, a theoretical connection between query strategies and reduction of expected risk was unclear. Let us therefore stick to the following idea:

If we are not able to reliably compute an example's value for the reduction of expected risk, then at least we should ensure that labeling the example is not for nothing, i.e., that at least something changes.

Deriving Estimated Model Changes: To follow the previous argumentation, let us focus on models which offer a parametric representation θ , e.g., normal distributions (see Section 2.1.4.2) where parameters μ and Σ are of fixed dimension, or representer models (see Eq. (2.110)) such as SVMs and GPs where parameters grow with the number of known examples. Furthermore, let us introduce $\mathcal{L}_\theta(\cdot, \cdot)$ as a loss function suited for comparing these model representations – we will see an example in Section 5.4.2. For a given $\mathcal{L}_\theta(\cdot, \cdot)$, we can evaluate the difference of parameters θ and θ' before and after updating the model. The estimated model

change criterion can then be expressed as follows:

$$\Delta \boldsymbol{\theta}(\mathbf{x}') = \mathbb{E}_{y' \in \mathcal{Y}} (\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\theta}')) \quad , \quad (5.16)$$

where the expectation over $y' \in \mathcal{Y}$ accounts for the yet unknown label of an example $\mathbf{x}' \in \mathcal{U}$. As for the estimated empirical risk in Eq. (5.11), we again estimate target probabilities based on the current model:

$$\Delta \boldsymbol{\theta}(\mathbf{x}') = \int_{\mathcal{Y}} \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\theta}') p(y'|f(\mathbf{x}')) dy' \quad . \quad (5.17)$$

Estimated Model Changes For Binary Classification Scenarios: Since we are concerned with classification tasks in this thesis, we further note that $\mathcal{Y}$ is discrete. Thus, we can finally express the estimated model change as:

$$\Delta \boldsymbol{\theta}(\mathbf{x}') = \sum_{y' \in \mathcal{Y}} \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\theta}') p(y'|f(\mathbf{x}')) \quad . \quad (5.18)$$

As motivated previously, we now approximate the expected gain of information after labeling $\mathbf{x}'$ by its estimated model change and obtain the following general query function:

$$\mathcal{Q}_{\Delta \boldsymbol{\theta}}(\mathbf{x}') = \Delta \boldsymbol{\theta}(\mathbf{x}') \quad , \quad \mathbf{x}_* = \operatorname{argmax}_{\mathbf{x}' \in \mathcal{U}} \Delta \boldsymbol{\theta}(\mathbf{x}') \quad . \quad (5.19)$$

A visualization of the underlying idea is given in Fig. 5.3.

It should be noted that this technique will only result in useful queries if the applied model family is not strongly affected by outliers, *i.e.*, if a single outlier will not result in significantly altered model parameters. While this would not hold for simple models such as least-square regression (see (Murphy, 2012, Sect.7.2 and 7.3)), more complex models often exhibit this property, *e.g.*, kernel-based methods with commonly chosen distance-based kernel functions. In the following, we present how to apply the estimated model change as active learning criterion to representer models within the framework of Gaussian process models.

5.4.2 Efficient Computations of EMC with GP

In Section 4.2.3, we have already seen that weight vectors of Gaussian Process regression models can be updated in closed form. For computing these updates, it turned out that three factors were required (see Eq. (4.42)). Furthermore, we have seen that all three factors allow for nice interpretations. First of all, a previously unseen example resulted in a large entry α_{N+1} in weight vector $\boldsymbol{\alpha}$ if and only if it came with a low predictive variance, *i.e.*, if it was *related* to the previous data distribution

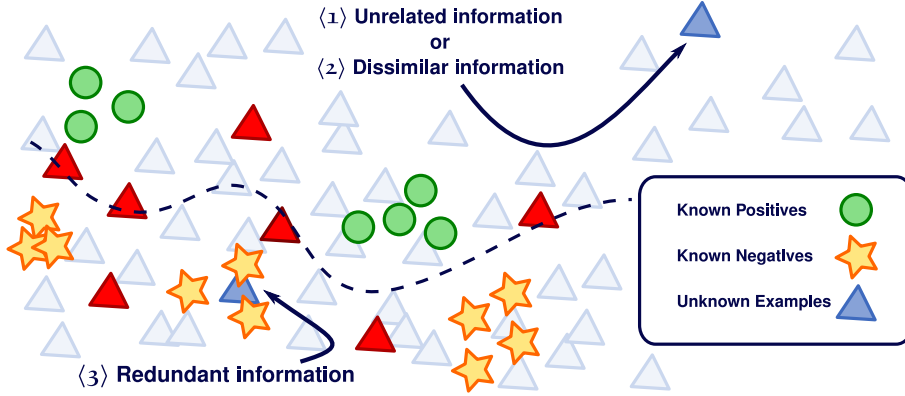


Figure 5.3: Illustration of a typical active learning scenario motivating estimated model changes as criterion to query with. Provided training data with positive and negative examples are shown as green circles and orange stars, respectively. For binary classification, we draw the learned decision boundary as dashed line. In addition, a number of examples without known labels is given as blue triangles. To spend our labeling budget wisely, we should avoid querying outliers or redundant examples. Instead, examples colored in red are likely to change the current model upon re-training and are thus of interest for active learning.

(part $\langle 1 \rangle$ in Eq. (4.42)) and if the previous model could not perfectly explain the new information in advance, *i.e.*, if it was *not redundant* (part $\langle 3 \rangle$ in Eq. (4.42)). Besides, the affect on entries of already known examples was additionally determined by the amount of shared information, *i.e.*, depending on how *similar* new and old examples were (part $\langle 2 \rangle$ in Eq. (4.42)).

EMC for GP Label Regression Models Combines Exploration and Exploitation:

From the viewpoint of active learning, this implicit formulation of “not redundant but still related” precisely meets the balancing of exploration and exploitation as motivated in Fig. 5.3. Let us therefore transfer the previous general definition of estimated model changes in Eq. (5.18) to GP models. As in Section 4.2.3, we will denote the change of α induced by treating x' with label y' as additional training example with:

$$\Delta\alpha = \alpha' - (\alpha^T, 0)^T. \quad (5.20)$$

Furthermore, let us specify the loss function $\mathcal{L}_\theta(\theta, \theta')$ which we need for evaluating the estimated model change in Eq. (5.18). Here, we apply the sum of element-wise absolute differences:

$$\mathcal{L}_\theta(\alpha, \alpha') = \|\Delta\alpha\|_1. \quad (5.21)$$

Given the closed-form solutions for $\Delta\alpha$ as derived in Theorem 4.2, Eq. (4.42), we can now put all parts together to obtain the following query function:

$$\mathcal{Q}_{\text{GP-}\Delta\theta}(\mathbf{x}') = \sum_{y' \in \mathcal{Y}} \left\| \frac{\mu(\mathbf{x}') - y'}{\sigma^2(\mathbf{x}')} \begin{bmatrix} \mathbf{K}^{-1} \mathbf{k}_* \\ -1 \end{bmatrix} \right\|_1 p(y' | f(\mathbf{x}')), \quad (5.22)$$

$$\mathbf{x}_* = \operatorname{argmax}_{\mathbf{x}' \in \mathcal{U}} \mathcal{Q}_{\text{GP-}\Delta\theta}(\mathbf{x}')$$

As we can see, transferring the estimated model change criterion to Gaussian process models implicitly combines aspects of exploration (part $\langle 1 \rangle$ and part $\langle 2 \rangle$ in Eq. (4.42)) and exploitation (part $\langle 3 \rangle$ in Eq. (4.42)). In addition, this combination comes without the necessity of manually defining combination techniques, *e.g.*, as required in (Baram et al., 2004; Cebon and Berthold, 2009; Ebert et al., 2012; Hospedales et al., 2013; Osugi et al., 2005).

Estimating Class Probabilities for Unlabeled Examples: The only question still unanswered is how to compute class probabilities $p(y' | f(\mathbf{x}'))$. Although the underlying GP regression model results in continuous scores $f(\mathbf{x}')$, requested labels y' are in fact discrete. In the application scenario presented in (Freytag et al., 2013), we investigated binary classification scenarios as motivated in Fig. 5.3 with $y' \in \{-1, 1\}$. Due to the probabilistic formulation of GP models, though, we not only obtain the predictive mean estimate $f(\mathbf{x}')$, but rather an entire predictive Gaussian distribution including the variance $\sigma^2(\mathbf{x}')$. We therefore follow Rodner (2011, Sect. 2.6.6) and compute class probabilities in closed form:

$$p(y' = 1 | \mathbf{x}') = \frac{1}{2} - \frac{1}{2} \cdot \operatorname{erf}\left(-f(\mathbf{x}') / \sqrt{2\sigma^2(\mathbf{x}')}\right) \quad (5.23)$$

using the error function $\operatorname{erf}(z)$. Note that Eq. (5.23) is also related to the cumulative Gaussian noise model presented by Rasmussen and Williams (2006, Sect. 3.3). In Section 5.6, we additionally show how to obtain multi-class probabilities if required by the current task. For now, however, we have everything at hand to compute the estimated model change $\mathcal{Q}_{\text{GP-}\Delta\theta}$ according to Eq. (5.22) with GP models.

An Approximation of EMC for GP Label Regression Models: Let us now move to an approximation of the overall model change which will shed new light on known strategies introduced by Kapoor et al. (2010). To this end, let us only focus on the entry $\alpha'(N+1)$ corresponding to the new example $\mathbf{x}'$. If the resulting weight is (close to) zero, the example will have almost no influence on future model predictions (see also Theorem 2.9). Thus, the selected example carries (almost) no additional information. Again, we should therefore intuitively query only those

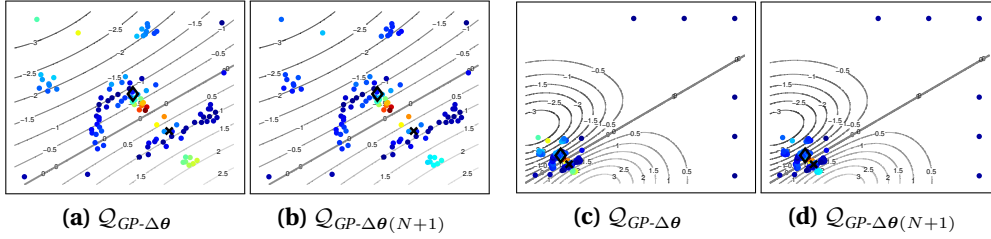


Figure 5.4: Active learning with estimated model changes (EMC) in a $D = 2$ scenario for visualization. The setup is identical to Fig. 5.1 (left) and Fig. 5.2 (right).

examples where estimated weights are large in absolute values:

$$\mathcal{Q}_{\text{GP-}\Delta\theta(N+1)}(\mathbf{x}') = \sum_{y' \in \mathcal{Y}} \frac{|\mu(\mathbf{x}') - y'|}{\sigma^2(\mathbf{x}')} p(y'|f(\mathbf{x}')) \quad , \quad (5.24)$$

$$\mathbf{x}_* = \operatorname{argmax}_{\mathbf{x}' \in \mathcal{U}} \mathcal{Q}_{\text{GP-}\Delta\theta(N+1)}(\mathbf{x}') \quad .$$

Interestingly, we observe a striking similarity to $\mathcal{Q}_{\text{GP-unc}}$ as introduced by Kapoor et al. (2010) (see Eq. (5.15)). Given our previous derivations, we now also have a theoretical justification for the combination proposed by Kapoor et al. (2010) which was only heuristically motivated and empirically validated so far. Thus, we can relate the predictive GP uncertainty to the probability of large resulting entries in the GP weight vector α .

Properties of Both Criteria: We again visualized active learning scores obtained from both techniques in Fig. 5.4. Besides different query techniques, the remaining setup is identical to previous visualizations to allow for direct comparisons.

As we can see, both techniques realize an implicit trade-off between exploration (e.g., top left and bottom right points) and exploitation (examples close to decision boundary between currently known data). In comparison to $\mathcal{Q}_{\text{GP-}\mu}$ and $\mathcal{Q}_{\text{GP-unc}}$, however, the area of large estimated informativeness is surprisingly small and less spread within the region of highest class ambiguity. In contrast to simple parametric models, we further note that the complexity of the GP model is sufficient for being unaffected by outliers. This fact is underlined by small scores for examples which are resident far off the main data distribution in Fig. 5.4c and Fig. 5.4d.

Besides these advantages, both techniques can not yet favor a cluster of new data over an isolated example as can be seen in Fig. 5.4a and Fig. 5.4b. Thus, although we can estimate the informativeness of an unlabeled example with respect to the current model, we can not yet estimate its value with respect to the current task defined by collected data. In the next section, we will therefore go one step

further and present how to solve this drawback.

Differences to the Publication (Freytag et al., 2013): When we originally introduced both techniques $\mathcal{Q}_{\text{GP-}\Delta\theta}$ and $\mathcal{Q}_{\text{GP-}\Delta\theta(N+1)}$, we had not yet placed them in the general context of estimated risk minimization for active learning at that time. As a result, we did not perform a probabilistically justifiable expectation over unknown labels, but instead proposed to rely on the class label which would result in smallest model changes. The underlying assumption was that an example which would strongly affect the current model even under the most pessimistic label is worth being labeled. As can be seen in Eq. (5.22), we thus obtained a lower bound approximation to $\mathcal{Q}_{\text{GP-}\Delta\theta(N+1)}$ and $\mathcal{Q}_{\text{GP-}\Delta\theta}$.

As a final note to avoid confusion, it should be mentioned that we referred to $\mathcal{Q}_{\text{GP-}\Delta\theta(N+1)}$ and $\mathcal{Q}_{\text{GP-}\Delta\theta}$ as GP-weight and GP-impact in our earlier publication (Freytag et al., 2013).

5.5 Active Learning Based on Estimated Model Output Changes

In the previous section, we introduced the criterion of estimated model changes. Thereby, we related an example's estimated informativeness to its probability of changing parameters of the current model after re-training, which was averaged over all possible labels the example might get. Intuitively, examples with large estimated model changes are likely to result in strong changes of future classification decision. Thus, they can possibly reduce classification errors. However, it is unclear whether these changes truly affect the region of feature space where data of the currently faced task is located. Therefore, we extend the previous idea in this section.

We propose to select examples that likely change future outputs rather than parameters of the model evaluated on collected data representative for the current task. As it will turn out, we thereby obtain a clear connection to the goal of reducing the expected risk as introduced in Section 5.1.1 and Section 5.2.4. The main theoretical results of this section are published in (Freytag et al., 2014c). Following the previous structure of sections, we will first introduce the new general approach of estimated model output changes. Afterwards, we will present efficient calculations for the choice of GP models. A summarizing video is available on YouTube³.

³<https://youtu.be/Po4kJCmMeGU>

5.5.1 Estimated Model Output Changes (EMOC)

We have seen in Section 5.1.1 that reliably selecting examples which minimize the expected risk is a desirable goal. However, we also motivated that this goal is almost impossible to realize in many real-world applications. By considering the introduced techniques and results we obtained so far, let us now raise the following final question:

Given a pool of unlabeled examples – some of them will change your model outputs when being labeled and added, others will not change anything – which one would you query?

In absence of further knowledge, we argue that examples that lead to large estimated model output changes should be queried.

Deriving Estimated Model Output Changes: Let us consider *how strongly a new example x' influences the model's decisions, marginalized over all possible inputs $x \in \Omega$ from the current task and over its yet unknown label $y' \in \mathcal{Y}$* :

$$\Delta f(x') = \mathbb{E}_{y' \in \mathcal{Y}} \mathbb{E}_{x \in \Omega} (\mathcal{L}(f(x), f'(x))) \quad . \quad (5.25)$$

Note that in contrast to $\mathcal{L}_\theta$ for the computation of estimated model changes in Eq. (5.16), $\mathcal{L}$ is a common loss function measuring the difference between model *outputs*. If we again rely on the current model for estimating probabilities for unknown labels, we can rewrite the estimated model output change induced by an unlabeled example x' as follows (compare also with the EMC criterion in Eq. (5.17)):

$$\Delta f(x') = \int_{\mathcal{Y}} \left(\int_{\Omega} \mathcal{L}(f(x), f'(x)) p(x) dx \right) p(y'|f(x')) dy' \quad (5.26)$$

$$= \int_{\mathcal{Y}} \int_{\Omega} \mathcal{L}(f(x), f'(x)) p(x) p(y'|f(x')) dx dy' \quad . \quad (5.27)$$

Again, for discrete labels as in classification tasks, this translates to:

$$\Delta f(x') = \sum_{y' \in \mathcal{Y}} \left(\int_{\Omega} \mathcal{L}(f(x), f'(x)) p(x) dx \right) p(y'|f(x')) \quad . \quad (5.28)$$

As for computing the change of empirical risk in Eq. (5.8), it now remains to specify $p(x)$ wisely. A naive solution could consist of regularly sampling the entire feature space Ω . However, this is not only practically infeasible for high-dimensional spaces, it also neglects the data distribution induced by the currently faced task, which usually lives in a low-dimensional manifold. We therefore follow our previous argumentation (see Eq. (2.9) and Eq. (5.8)) and approximate $p(x)$ with the *empirical*

density induced by collected examples, *i.e.*, all examples $\mathbf{x} \in \mathcal{L} \cup \mathcal{U}$:

$$p_{\delta}(\mathbf{x}) = \frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{\mathbf{x}_j \in \mathcal{L} \cup \mathcal{U}} \delta_{\{\mathbf{x}_j\}}(\mathbf{x}) . \quad (5.29)$$

Again, $\delta_{\{\mathbf{x}_j\}}(\mathbf{x})$ is the Dirac measure (see Eq. (2.8)) induced by the set which contains only $\mathbf{x}_j$ and which is evaluated on $\mathbf{x}$. By replacing $p(\mathbf{x})$ with $p_{\delta}(\mathbf{x})$ in Eq. (5.28), we can thus express the estimated change of model outputs as follows:

$$\Delta f(\mathbf{x}') = \sum_{y' \in \mathcal{Y}} \left(\frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} \mathcal{L}(f(\mathbf{x}), f'(\mathbf{x})) \right) p(y' | f(\mathbf{x}')) . \quad (5.30)$$

In consequence, our estimates for $\Delta f(\mathbf{x}')$ are directly influenced by the collected datasets $\mathcal{L}$ and $\mathcal{U}$. We could further stress this fact by explicitly denoting the dependency as $\Delta f(\mathbf{x}' | \mathcal{L}, \mathcal{U})$ which would be even more precise from a notational point of view (see also the short discussion in Section 5.1.1). Nonetheless, we will again stick to the short version of $\Delta f(\mathbf{x}')$ in the remaining sections to keep formulas as compact as possible.

EMOC versus Estimated Reduction of Estimated Empirical Risk: To finally see the close connection to the estimated reduction of estimated empirical risk, let us slightly re-arrange involved terms in Eq. (5.30) (only black solid terms):

$$\begin{aligned} \Delta f(\mathbf{x}') &= \sum_{y' \in \mathcal{Y}} p(y' | f(\mathbf{x}')) \cdot \frac{1}{|\mathcal{L} \cup \mathcal{U}|} \cdot \dots \\ &\quad \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} \sum_{y \in \mathcal{Y}} \left(\mathcal{L}(y, f(\mathbf{x})) p(y | f(\mathbf{x})) - \mathcal{L}(y, f'(\mathbf{x})) p(y | f'(\mathbf{x})) \right) . \end{aligned} \quad (5.31)$$

We additionally displayed the corresponding terms of $\Delta \mathcal{R}_{\mathbb{E}}$ in gray to allow for an easy comparison with the estimated reduction of estimated empirical risk in Eq. (5.12). As we see, both approaches are strikingly similar. However, $\Delta \mathcal{R}_{\mathbb{E}}$ matches each model's prediction against an additionally required estimate of the ground truth function $g(\cdot)$ (this causes the double use of “estimate” in the method's name). In contrast, our introduced criterion Δf directly compares model predictions before and after re-training.

Thus, we can interpret $\Delta \mathcal{R}_{\mathbb{E}}$ as selecting examples which likely change the model's predictions towards the “right” direction, *i.e.*, towards the ground truth function. With EMOC, we relax this task since reliably estimating $g(\cdot)$ is usually tricky, if not impossible at all. As a result, we approximate the estimated decrease

How much does the output change?

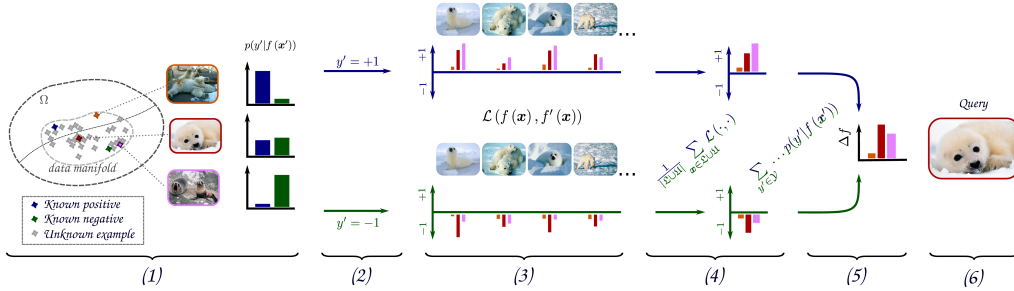


Figure 5.5: Illustration of active learning with estimated model output changes (EMOC) Δf in a binary classification scenario: (1) an initial model is trained from labeled examples (blue and green). (2) under each possible label, unlabeled examples (gray) are evaluated with respect to their probability of changing the model's outputs on collected data representative for the current task. (3) for three unlabeled examples (red, orange, and pink), the output change on four different examples is visualized. Final estimates are obtained by marginalizing over all collected data (4) and possible labels (5). The unlabeled example with highest score is finally selected for human annotation (6). See also text in Section 5.5.1 for details. Figure is best viewed in color and by zooming in.

in estimated empirical risk using the following general query function:

$$\mathcal{Q}_{\Delta f}(\mathbf{x}') = \Delta f(\mathbf{x}'), \quad \mathbf{x}_* = \operatorname{argmax}_{\mathbf{x}' \in \mathcal{U}} \Delta f(\mathbf{x}') . \quad (5.32)$$

Properties of the EMOC Criterion: An illustration of the underlying idea is given in Fig. 5.5. On the left side, we visualized the entire input space as well as the manifold induced by collected unlabeled data representative for the current task. In addition, the decision boundary of a binary model learned from two labeled examples is shown as dotted line.

We then visualize how to compute Δf for three examples $\mathbf{x}' \in \mathcal{U}$. While the orange example is an outlier with respect to collected data and thus does not significantly influence outputs on any other example under any possible label, the red and pink example are located in regions of high density. Although strongest changes would be obtained for the pink example under the assumption of $y' = +1$, it is not strongly supported by the current model. In consequence, after marginalizing over all collected data and all possible labels, the red example is estimated to result in largest estimated changes of model outputs. An additional visualization on $D = 2$ data is given in Section 7.3.1.1.

EMOC is an Upper Bound on Estimated Risk Reduction: Our introduced EMOC criterion can be additionally motivated from a theoretical perspective by further inspecting the tight connection to estimated reduction of the expected risk $\mathcal{R}$. To this end, let us start with a short motivation, and then formalize the investigation.

Ideally, we want select examples which reduce the expected error as much as possible. However, we have already seen in previous discussions that estimation of expected errors is likely error prone. A common solution is to limit the estimated reduction of expected errors and to optimize these bounds instead. Intuitively, maximizing a lower bound is desirable, since this would guarantee that the selected example will decrease expected errors *at least* for the calculated amount. Unfortunately, finding sufficiently tight lower bounds is hard as well. On the contrary, an upper bound does not guarantee *per se* that expected errors decrease, however, a large estimate of an upper bound function does also not limit the maximal possible decrease in advance. In other words, if an example only receives a small values from the upper bound function, it is guaranteed to not result in large benefits. A large upper bound output instead preserves the *possibility* for large error reductions.

Based on the previous argumentation, it now remains to investigate the connection between $\Delta\mathcal{R}$ and Δf in detail. As it turns out, selecting an example with largest estimated change of model outputs is equivalent to the aforementioned maximization of an upper bound on estimated loss reduction. To see this, let us follow an argumentation by Vezhnevets et al. (2012) and let us assume that the loss function $\mathcal{L}$ obeys the following triangle inequality for $a, b, c \in \mathcal{Y}$:

$$\mathcal{L}(a, b) \leq \mathcal{L}(a, c) + \mathcal{L}(c, b) \quad . \quad (5.33)$$

Among other choices, the triangle inequality also holds for the aforementioned 0/1-loss or the L_1 -loss. Let us further denote with $f_* \in \mathbb{F}$ the model learned from all possible data representing the gold standard with minimal risk, *e.g.*, for regression, $f_*(x)$ could be the ground truth function $g(x)$. Finally, let us measure the (dis-)agreement of two models with $\mathcal{L}_f(f_1, f_2)$:

$$\mathcal{L}_f(f_1, f_2) = \mathbb{E}_{x \in \Omega} \left(\mathcal{L}(f_1(x), f_2(x)) \right) \quad . \quad (5.34)$$

Note that if $f_2(x)$ is indeed the ground truth function $g(x)$, Eq. (5.34) is the expected risk $\mathcal{R}(f_1)$ as defined in Eq. (2.1). A notational difference between Eq. (5.34) and Eq. (2.1) is the assumed uniqueness of class labels which removes the previously necessary expectation over $y \in \mathcal{Y}$ (see also Eq. (2.9) and the following discussion). The estimated decrease in risk can now be defined as:

$$\Delta\mathcal{R}(x') = \mathbb{E}_{y' \in \mathcal{Y}} \left(\mathcal{L}_f(f, f_*) - \mathcal{L}_f(f', f_*) \right) \quad . \quad (5.35)$$

Given the assumed triangle inequality of Eq. (5.33), we directly obtain the desired connection between estimated risk reduction and estimated model output changes:

$$\Delta \mathcal{R}(x') = \mathbb{E}_{y' \in \mathcal{Y}} (\mathcal{L}_f(f, f_*) - \mathcal{L}_f(f', f_*)) \quad (5.36)$$

$$= \mathbb{E}_{y' \in \mathcal{Y}} \mathbb{E}_{x \in \Omega} (\mathcal{L}(f(x), f_*(x)) - \mathcal{L}(f'(x), f_*(x))) \quad (5.37)$$

$$\leq \mathbb{E}_{y' \in \mathcal{Y}} \mathbb{E}_{x \in \Omega} (\mathcal{L}(f(x), f'(x))) \quad (5.38)$$

$$= \Delta f(x'). \quad (5.39)$$

Let us again stress that it is impossible to directly maximize the estimated reduction of expected risk on the left-hand side since f_* is unknown. As derived previously, we therefore suggest to search for the unlabeled example x_* with highest upper bound in loss reduction given by EMOC. Even if this of course does not guarantee a proper decrease in risk, it at least does not limit it in advance by selecting examples that do not change model outputs at all.

Differences to the Publications (Freytag et al., 2014c) and (Käding et al., 2015):

Before we move on, we would like to stress a minor difference to the nomenclature in our earlier publications (Freytag et al., 2014c) and (Käding et al., 2015). In detail, we referred to $\mathcal{Q}_{\Delta f}$ as *expected* model output change rather than *estimated* model output change in both publications. For the reduction of risk as introduced in Eq. (5.5), however, we used the term *expected* risk when referring to the theoretical, abstract quantity with known distributions over class labels. Instead, we used *estimated* empirical risk for the scenario where class labels are unknown. In this thesis, we thus follow this common nomenclature and denote $\mathcal{Q}_{\Delta f}$ as *estimated* model output change due to the necessity of estimating class labels during the evaluation as shown in Eq. (5.30).

Relationship to State-of-the-art: The only work we are aware of which follows the EMOC principle was introduced by Vezhnevets et al. (2012) for the task of semantic segmentation and further adapted by Roig et al. (2013). The authors propose to request labeling an image segment which is estimated to result in strongest changes of label outputs for the underlying conditional random field, and special care is taken on efficient implementations. However, both works did not put the technique into a general perspective. To the best of our knowledge, we are the first ones posing EMOC into a general perspective of approximating the reduction of estimated empirical risk, applying EMOC to active learning for pure image classification, and presenting efficient computations for the choice of Gaussian process models.

It should be noted that from the viewpoint of EMOC, our early paper on novelty detection (Bodesheim et al., 2012) approximates Δf by computing the output

change only for x' , *i.e.*, on the example to evaluate. At the time of publication, we argued that large changes of the model's outputs on the new example itself can only occur under all possible label assumptions if the new example is an outlier with respect to known data and, thus, no further information on the output can be transferred. See (Bodesheim et al., 2012, Figure 2) for a visualization. While this motivation still remains valid for novelty detection, we are interested in an orthogonal goal in this thesis: discovering new clusters and classes rather than outliers. In other words, we still need to be unaffected by outliers, but we are additionally interested in discovering compact clusters in space (which could correspond to new sub-species or previously unrecorded viewpoints of known categories). Thus, we argue that including the task-induced data density is essential to finally allow for active learning and discovery for lifelong learning. Further details on applying the EMOC principle to scenarios with initially unknown classes are given in Section 5.6.

5.5.2 Efficient EMOC with GP Label Regression

The previously introduced technique of estimated model output changes was neither restricted to any model family nor to a specific task defined by the label space $\mathcal{Y}$. We now follow our publication history and start by transforming the general concept of Δf into an actual active learning scenario (see definition in Eq. (5.4)) for the choice of GP regression models. As in Section 5.4.2, we follow our previous publication (Freytag et al., 2014c) and restrict us to binary classification scenarios. An extension to multi-class scenarios as published in (Käding et al., 2015) will be discussed in Section 5.6.

When inspecting the general formulation of $\Delta f(x')$ in Eq. (5.30), we again note that two main ingredients need to be specified besides the actual choice of the model family $\mathbb{F}$: (i) the loss function $\mathcal{L}(\cdot, \cdot)$ and (ii) a transfer function to compute class probabilities $p(y'|f(x'))$ from model outputs. For the latter one, we again rely on the predictive distribution obtained from the GP model and apply Eq. (5.23) to obtain estimates for class probabilities. The choice of $\mathcal{L}(\cdot, \cdot)$, however, naturally complies with the faced problem setting.

Specifying a Sensible Loss Function: We already gave a short introduction to loss functions in Section 2.1.2.1 noticing that an L_1 -loss as defined in Eq. (2.3) would be well suited for regression tasks with continuous output values. Although our final label space is binary, decisions are obtained by thresholding underlying continuous model outputs or probability estimates derived thereof. Thus, instead of applying a 0/1-loss on classification decisions, we argue to already compare the continuous outputs to better reflect subtle changes in the latent function and the non-ordinal nature of GP regression models. Furthermore, this decision will allow for developing a fast approximation in the following Section 5.5.3. To this end, let

us inspect the inner part of Δf in Eq. (5.30) in detail for the choice of GP models and the aforementioned L_1 -loss:

$$\sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} |f'(\mathbf{x}) - f(\mathbf{x})| \quad (5.40)$$

$$\stackrel{\boxed{\text{Def. } f}}{=} \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} |\kappa(\mathbf{x}, \{\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{x}'\}) \boldsymbol{\alpha}' - \kappa(\mathbf{x}, \{\mathbf{x}_1, \dots, \mathbf{x}_N\}) \boldsymbol{\alpha}| \quad (5.41)$$

$$= \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} \left| \kappa(\mathbf{x}, \{\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{x}'\}) \left(\boldsymbol{\alpha}' - \begin{bmatrix} \boldsymbol{\alpha} \\ 0 \end{bmatrix} \right) \right| \quad (5.42)$$

$$\stackrel{\boxed{\text{Eq. (5.20)}}}{=} \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} |\kappa(\mathbf{x}, \{\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{x}'\}) \Delta \boldsymbol{\alpha}| \quad (5.43)$$

Thus, changes of continuous outputs evaluated on all collected examples $\mathbf{x} \in \mathcal{L} \cup \mathcal{U}$ can again be computed in closed form using the update rules for $\boldsymbol{\alpha}$ introduced in Section 4.2.3.

The Resulting Query Criterion: In consequence, we obtain the following query strategy when transferring the EMOC criterion to GP models for label regression:

$$\mathcal{Q}_{\text{GP-}\Delta f}(\mathbf{x}') = \sum_{y' \in \mathcal{Y}} \left(\frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} |\Delta \boldsymbol{\alpha}^T \mathbf{k}'_*| \right) p(y' | f(\mathbf{x}')), \quad (5.44)$$

$$\mathbf{x}_* = \operatorname{argmax}_{\mathbf{x}' \in \mathcal{U}} \mathcal{Q}_{\text{GP-}\Delta f}(\mathbf{x}')$$

We again applied the resulting criterion $\mathcal{Q}_{\text{GP-}\Delta f}$ to the previously introduced $D = 2$ toy example and visualized results in Fig. 5.6. In comparison with Fig. 5.1 and Fig. 5.4, we note that $\mathcal{Q}_{\text{GP-}\Delta f}$ combines several desirable properties: while being unaffected by outliers, we additionally observe a strong focus on dense clusters due to the expectation over collected examples. Furthermore, examples being likely redundant are neglected as well. This intuition will be further confirmed in a more detailed analysis in Section 7.3.

5.5.3 An Efficiently Computable Approximation of EMOC

Approximating EMOC Scores of Representer Models: Efficient model updates as presented previously can greatly reduce the computational effort for evaluating $\mathcal{Q}_{\text{GP-}\Delta f}$. However, the computational burden is still huge for large sets of unlabeled examples. We therefore propose the following upper bound approximation to

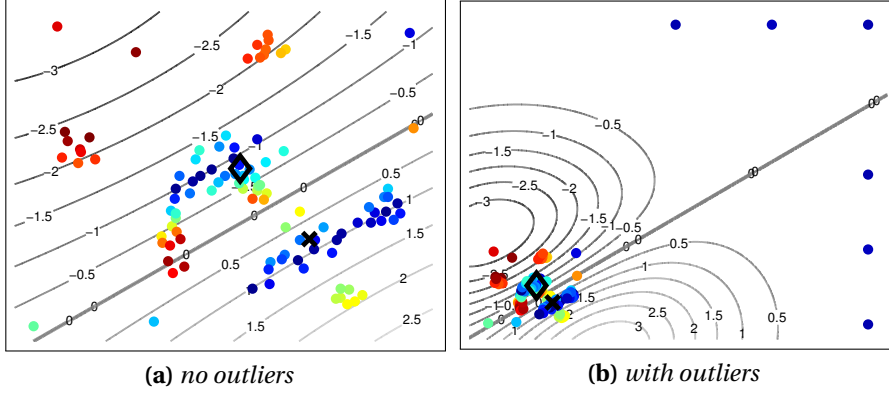


Figure 5.6: Active learning with estimated model output changes (EMOC) transferred to GP for label regression resulting in $\mathcal{Q}_{\text{GP-}\Delta f}$. The $D = 2$ scenario for visualization and the setup is identical to previous visualizations in Fig. 5.1, Fig. 5.2, and Fig. 5.4.

$\mathcal{Q}_{\text{GP-}\Delta f}$ which significantly reduces the number of required operations:

$$\mathcal{Q}_{\text{GP-}\Delta f}(x') = \sum_{y' \in \mathcal{Y}} \left(\frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{x \in \mathcal{L} \cup \mathcal{U}} |\Delta \alpha^T k'_*| \right) p(y'|f(x')) \quad (5.45)$$

$$\leq \sum_{y' \in \mathcal{Y}} \left(\frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{x \in \mathcal{L} \cup \mathcal{U}} \|\Delta \alpha^T\|_1 \|k'_*\|_1 \right) p(y'|f(x')) \quad (5.46)$$

$$= \sum_{y' \in \mathcal{Y}} \left(\sum_{i=1}^{N+1} \frac{|\Delta \alpha(i)|}{|\mathcal{L} \cup \mathcal{U}|} \sum_{x \in \mathcal{L} \cup \mathcal{U}} |\kappa(x_i, x)| \right) p(y'|f(x')) \quad (5.47)$$

$$\doteq \mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}(x') \quad , \quad (5.48)$$

with $x_i \in \mathcal{L} \cup \{x'\}$. The last step follows from the fact that the change of α itself is independent of examples used to evaluate the change of model outputs. The computational benefit of the approximation introduced above is two-fold: (i) the asymptotic computation time reduces, and (ii) instead of demanding the kernel matrix consisting of kernel values between every two examples of $\mathcal{L} \cup \mathcal{U}$, computing the approximated version only needs the resulting row sums.

Density Re-Weighting of EMC Approximates EMOC: Interestingly, we can interpret the inner part in Eq. (5.47) as a Parzen density estimate centered at each training example x_i and evaluated with all collected data $\mathcal{L} \cup \mathcal{U}$. Following this point of view, we can even further approximate $\mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}$. Let us therefore denote the

vector containing density values of all labeled examples with:

$$\mathbf{p}_{\mathcal{L}}(i) = \frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} \kappa(\mathbf{x}_i, \mathbf{x}), \quad i = 1, \dots, N. \quad (5.49)$$

Furthermore, let $p_{\mathbf{x}'}$ be the density estimate for the unlabeled example $\mathbf{x}'$ induced by collected data:

$$p_{\mathbf{x}'} = \frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} \kappa(\mathbf{x}', \mathbf{x}) \quad (5.50)$$

Then, we can rewrite the approximated EMOC criterion for GP in Eq. (5.47) as follows:

$$\mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}(\mathbf{x}') = \sum_{y' \in \mathcal{Y}} \left(\left\| \Delta \boldsymbol{\alpha}^T \begin{bmatrix} \mathbf{p}_{\mathcal{L}} \\ p_{\mathbf{x}'} \end{bmatrix} \right\| \right) p(y' | f(\mathbf{x}')). \quad (5.51)$$

Applying the triangle inequality and further removing all terms not depending on $\mathbf{x}'$ finally gives us the following coarse approximation:

$$\mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}(\mathbf{x}') \leq \sum_{y' \in \mathcal{Y}} \left(\|\Delta \boldsymbol{\alpha}\|_1 \cdot \left\| \begin{bmatrix} \mathbf{p}_{\mathcal{L}} \\ p_{\mathbf{x}'} \end{bmatrix} \right\|_1 \right) p(y' | f(\mathbf{x}')) \quad (5.52)$$

$$\propto \sum_{y' \in \mathcal{Y}} \left(\|\Delta \boldsymbol{\alpha}\|_1 \cdot |p_{\mathbf{x}'}| \right) p(y' | f(\mathbf{x}')) \quad (5.53)$$

$$= |p_{\mathbf{x}'}| \sum_{y' \in \mathcal{Y}} \|\Delta \boldsymbol{\alpha}\|_1 p(y' | f(\mathbf{x}')) \quad (5.54)$$

$$\doteq \mathcal{Q}_{\text{GP-}\Delta \theta\text{-pde}}(\mathbf{x}'). \quad (5.55)$$

As we can see, the approximation above is equivalent to taking queries based on the estimated model change criterion $\mathcal{Q}_{\text{GP-}\Delta \theta}$ (see Eq. (5.22) in Section 5.4), but with an additional weighting factor determined by the Parzen density estimate of the corresponding unlabeled example. In contrast to using $\mathcal{Q}_{\text{GP-}\Delta \theta}$ alone, we thereby ensure that the selected example not only affects the current model, but that it is also likely to occur in dense regions of the feature space. And, perhaps most importantly, denseness is no universally fixed measure, but it is instead tailored to the current current task defined by its data distribution.

Thoughts on Density Re-Weighting: In fact, we can even generalize the idea behind Eq. (5.54) and transfer the principle of density re-weighting to arbitrary strategies:

$$\mathcal{Q}^*(\mathbf{x}') = p_{\mathbf{x}'} \cdot \mathcal{Q}(\mathbf{x}') . \quad (5.56)$$

As we can see, this strategy is extremely easy to implement and adjusts arbitrary query criteria such that examples of high density areas with respect to the current problem and its induced data distribution are preferred over outliers from non-important, sparse regions. It should be noted that this heuristic is by far not novel, *e.g.*, it has been introduced in Settles and Craven (2008) as density-weighting. Nonetheless, the authors proposed it as a heuristic without a clear theoretical motivation and simply noted that they would “recommend it in practice”. With our previous derivation at hand, we can now place this technique in a theoretical background as a special approximation of the estimated model output change criterion.

5.6 Active Learning for Scenarios with Multiple Classes and Unnameable Instances

For all previously derived techniques, we focused on binary classification scenarios and derived efficient algorithms for GP label regression models. However, lifelong learning scenarios will naturally face more than two categories over time. To account for this aspect, we now present possible extensions for multi-class scenarios with labels $y \in \mathcal{Y} = \{1, \dots, M\}$ in Section 5.6.1 and Section 5.6.2.

Furthermore, we observed that going from binary questions like “Is this image of class X – yes or no?” to categorical questions such as “Which of this M categories does the image belong to?” comes with two new types of problems. First of all, selected images can contain non-categorical content, *e.g.*, due to camera errors. Besides, even if the contained objects are categorical, they are not necessarily known to the annotator or not of interest to him (*i.e.*, not part of his problem domain). In both cases, the annotator would refuse the labeling which results in no information gained at all. We reflect this fact by referring to the corresponding instances as *unnameable*. In Section 5.6.3, we show first attempts which expand the current EMOC criterion to circumvent both issues.

The results of this section are based on our findings published in (Käding et al., 2015). A video introduction is available on YouTube⁴.

5.6.1 Active Learning for One-vs-all GP Models

Let us start by transferring techniques of Kapoor et al. (2010) as reviewed in Section 5.3.1 to multi-class settings. For the predictive mean, several options arise which all have different characteristics. The first intuitive option is to focus on the

⁴<https://youtu.be/AEIrYqMHH74>

most probable category and apply the binary criterion of Eq. (5.13):

$$\begin{aligned} \mathcal{Q}_{\text{GP-}\mu\text{-MC-1}}(\mathbf{x}') &= \left| \max_{m=1\dots M} \mu_m(\mathbf{x}') \right|, \\ \mathbf{x}_* &= \underset{\mathbf{x}' \in \mathcal{U}}{\operatorname{argmin}} \mathcal{Q}_{\text{GP-}\mu\text{-MC-1}}(\mathbf{x}') \end{aligned} \quad (5.57)$$

This would prefer examples where the most probable category can not be decided reliably. To take the distribution of all category scores into account, we could alternatively prefer examples where all binary models are similarly undecided. This goal is reflected by selecting examples where all class scores deviate rarely from the binary decision margin:

$$\begin{aligned} \mathcal{Q}_{\text{GP-}\mu\text{-MC-dev.}}(\mathbf{x}') &= \max_{m=1\dots M} |\mu_m(\mathbf{x}')|, \\ \mathbf{x}_* &= \underset{\mathbf{x}' \in \mathcal{U}}{\operatorname{argmin}} \mathcal{Q}_{\text{GP-}\mu\text{-MC-dev.}}(\mathbf{x}') \end{aligned} \quad (5.58)$$

A third option arises by preferring examples where even the most plausible category is extremely implausible (as usually the case for outliers):

$$\begin{aligned} \mathcal{Q}_{\text{GP-}\mu\text{-MC-out.}}(\mathbf{x}') &= \max_{m=1\dots M} \mu_m(\mathbf{x}'), \\ \mathbf{x}_* &= \underset{\mathbf{x}' \in \mathcal{U}}{\operatorname{argmin}} \mathcal{Q}_{\text{GP-}\mu\text{-MC-out.}}(\mathbf{x}') \end{aligned} \quad (5.59)$$

With respect to the predictive variance, we already noticed that active learning scores are independent of class labels. In consequence, the explorative criterion $\mathcal{Q}_{\text{GP-}\sigma^2}$ as defined in Eq. (5.14) can be readily used for multi-class scenarios.

For the predictive uncertainty as introduced by Kapoor et al. (2010) and reviewed in Eq. (5.15), we can apply the same ideas as for the predictive mean criterion. Thus, we obtain the three variants $\mathcal{Q}_{\text{GP-unc-MC-1}}$, $\mathcal{Q}_{\text{GP-unc-MC-dev.}}$, and $\mathcal{Q}_{\text{GP-unc-MC-out.}}$. In the next section, we show how to apply our EMOC criterion to multi-class scenarios. It is noteworthy that the resulting techniques can be also used for our EMC criterion introduced in Section 5.4.

5.6.2 EMOC for Multi-class Scenarios

In Section 5.5, we introduced the EMOC criterion Δf as an approximation to the estimated risk reduction. For the specific choice of GP label regression models, we then showed how to implement EMOC in binary classification scenarios as proposed in our publication (Freytag et al., 2014c). To also apply EMOC in multi-class scenarios with one-vs-all models, we need to answer two questions for the computation of EMOC scores shown in Eq. (5.30):

- (i) Which loss function $\mathcal{L}(\cdot, \cdot)$ is suitable for multi-class settings? and
- (ii) How can we reliably estimate class probabilities?

In the following, we provide solutions for both aspects.

Loss Functions for Model Outputs of One-vs-all Classifiers: As reviewed in Section 2.1.5.8, applying Gaussian process models to classification scenarios with $M > 2$ categories can be done most easily by following the one-vs-all idea. After training M binary models, continuous classification scores of all models can finally be transformed into a discrete decision, *e.g.*, by returning the class label with largest score (see Eq. (2.132)). Besides the intuitive choice of a 0/1-loss as loss function to compare discrete classification results, we can thus also apply loss functions which directly evaluate changes of the intermediate, continuous scores of all one-vs-all models. We therefore suggest to apply an L_1 -loss directly on the class-specific scores:

$$\mathcal{L}_{|\cdot|}(f(\mathbf{x}), f'(\mathbf{x})) = \sum_{m=1}^M |f_m(\mathbf{x}) - f'_m(\mathbf{x})| \quad (5.60)$$

and fuse changes of individual classifier outputs by adding all scores. Besides, it would also be interesting to investigate the application of a *log-loss* which would favor examples that maximally change the classification entropy.

Multi-class Classification Probabilities: In contrast to the multi-class loss function in Eq. (5.60), where loss values can be computed for each binary classifier separately and results are easily fused, computing class probability estimates $p(y = m | \mathbf{x})$ from one-vs-all models is more challenging. As in Section 2.1.5.8, let $y_m \in \{-1, 1\}$ denote the binary label of $\mathbf{x}$ with respect to category m . Naively computing probability estimates could then be done by computing class-specific probabilities $p(y_m = 1 | \mathbf{x})$ for each binary classification problem as shown in Eq. (5.23). While being suitable for each binary task on its own, the obtained probability estimates are not sufficiently normalized with respect to the multi-class scenario. Furthermore, simply normalizing them in advance is difficult due to imbalanced scores as shown by Milgram et al. (2005, Figure 1). Instead, a common alternative is to transform class scores using a soft-max function as introduced by Duan et al. (2003) and Milgram et al. (2005):

$$p(y = m | \mathbf{x}) \propto \exp(\alpha_m \cdot f_m(\mathbf{x}) + \beta_m) \quad (5.61)$$

Class-specific parameters α_m and β_m can be estimated from left-out training data via cross-validation. An obvious drawback of Eq. (5.61), however, is the fact that probability estimates are computed using only the binary class scores, although

a GP model would offer the entire predictive distribution instead. Following this observation, Fröhlich et al. (2013) propose an alternative technique by applying a Monte-Carlo sampling strategy. To this end, the authors argue that the multi-class probability for class m can be interpreted as the probability that the random variable associated with the m^{th} binary model resulted in largest score:

$$p(\bar{y}(\mathbf{x}) = m | \mathbf{x}) = p\left(m = \underset{m'=1 \dots M}{\operatorname{argmax}} f_{m'}(\mathbf{x})\right). \quad (5.62)$$

As argued by Fröhlich et al. (2013), it is unknown whether there exists an analytically closed-form solution for Eq. (5.62) even if each probability distribution $f_{m'}(\mathbf{x})$ is only a $D = 1$ Gaussian. Therefore, the authors propose to sample Z times from every class-specific predictive distribution $\mathcal{N}(\mu_m(\mathbf{x}), \sigma^2(\mathbf{x}))$ defined by predictive mean and predictive variance. For every category, a variable Z_m counts how often a random draw of category m resulted in the largest value:

$$p(y = m | \mathbf{x}) = p(\bar{y}(\mathbf{x}) = m | \mathbf{x}) \approx \frac{Z_m}{Z}. \quad (5.63)$$

Thereby, uncertain class estimates indicated by a large variance σ^2 result in nearly uniform distributions $p(y = m)$. Similarly, small variances result in highly peaked probability distributions which ultimately converge to unity vectors associated with the category of highest posterior mean estimate.

When applying both presented techniques, we obtain a multi-class variant of EMOC which we denote with $\mathcal{Q}_{\text{GP-}\Delta f\text{-MC}}$.

5.6.3 Active Learning With Unnameable Instances

So far, we assumed the human-in-the-loop to be a perfect and reliable oracle who is always able to provide the requested label information for an unlabeled example. However, especially in lifelong learning scenarios where the number of potentially occurring categories can be vast, this assumption might not be feasible. From our experience, annotators are in fact likely to refuse the labeling of an requested example in two scenarios, which we visualized in Fig. 5.7:

- (i) **Rejection of non-categorical examples:** The unlabeled example does not show a valid object. Possible reasons are *noise* during image acquisition (e.g., sensor noise, motion blur, or JPEG artifacts), segments covering *multiple objects*, or *background* regions. Examples are bounding boxes (b) and (c) in Fig. 5.7.
- (ii) **Rejection of categorial examples:** The unlabeled example is a valid object, but the annotator is *not able* to name it or he decides that it is *not part* of the

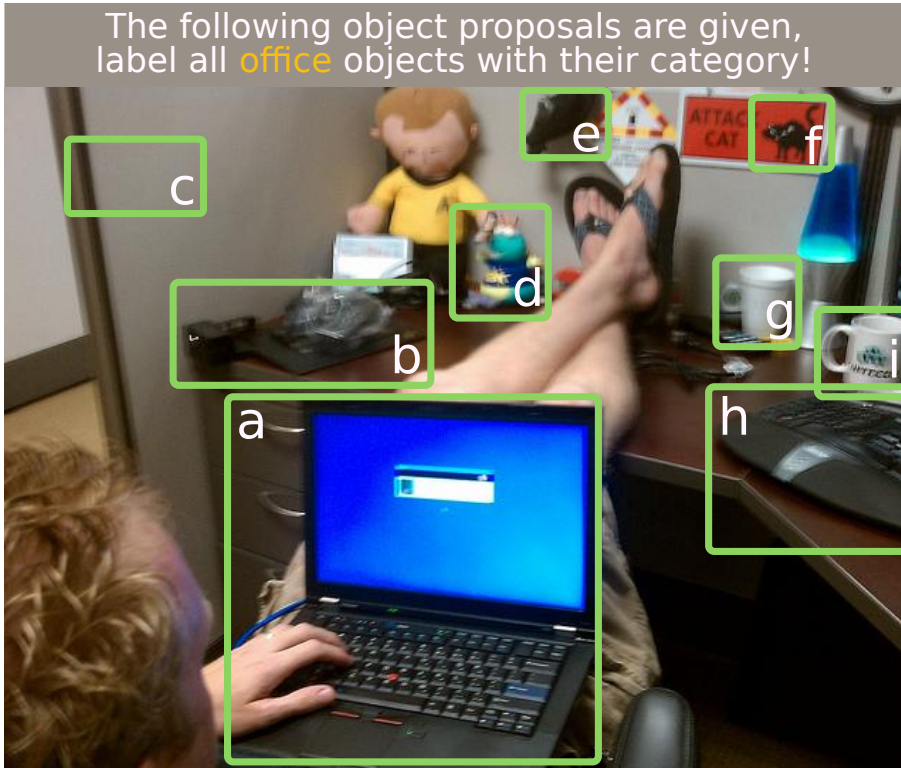


Figure 5.7: Visualizing problems in active learning with unnameable instances. The answers of an annotator for the given task are likely to be: (a) laptop (b) *what the heck is that?*, (c) *no object!*, (d) *some toy figure, but not really an office category*, (e) *no idea*, (f) *cat – but not an office category*, (g) cup, (h) keyboard, and (i) cup. In an active learning scenario, an annotator might reject examples that *do not show valid objects*, that *cannot be identified*, or that are *not part of the problem domain*, which should be considered when requesting annotations.

problem domain defined by $\mathcal{Y}$. Examples are bounding boxes (d), (e), and (f) in Fig. 5.7.

To avoid selecting examples for annotation which would be rejected by an annotator, we need to integrate both aspects into query strategies. After a short review of related approaches, we present two simple solutions by integrating additional terms into the previously introduced EMOC criterion for multiple categories.

Relationship to State-of-the-art: For dealing with noisy oracles in crowd-sourcing, Long et al. (2013) combine the maximum entropy framework with estimates about expected labeling quality. In contrast, we assume experts to be correct, but with the *choice of rejecting examples* during labeling. Similar in spirit is the work by Fang and Zhu (2012), which models the possibility of oracle rejections in binary

classification scenarios. Their assumption of zero classification entropy for rejected examples, however, is likely to fail in practice. Therefore, we differ both in tackling the more general multi-class setting and a sound theoretical model for selective oracles. Closest in terms of targeted application is the inspiring work of Hospedales et al. (2012) which introduces a framework for joint active learning and class discovery. Unfortunately, their approach does not scale to applications with several thousands of dimensions and unlabeled examples as we will see in Section 7.3.2. In addition, we empirically found that no explicit modeling of unknown classes is required when incorporating data density instead.

Rejections of Non-categorical Examples: Although we can imagine arbitrarily many images showing no object at all, the probability that users actually take these images is low. In fact, non-categorical examples are typically outliers in data collections, *e.g.*, due to sensor noise or uncorrected motion blur, especially if data collections are pre-filtered for aesthetic purpose such as Flickr. We argue that it is therefore a reasonable assumption to expect non-categorical examples only in regions of low density. In contrast, examples of object categories should cluster since multiple examples from the same category are likely to be recorded over time. In consequence, requested examples should ideally exhibit high density values, which can be explicitly integrated into the chosen query strategy. We have already seen in Section 5.5.3 that approximating the EMOC criterion can be viewed as a density-weighted version of the estimated model change criterion introduced in Section 5.4. As a simple heuristic, we therefore suggest to transfer the explicit weighting of EMOC scores with the data density of each unlabeled example x' , computed as a Parzen density estimate. We thus obtain the following general query strategy:

$$\begin{aligned} \mathcal{Q}_{\text{GP-}\Delta f\text{-MC-PDE}}(x') &= \sum_{y' \in \mathcal{Y}} \left(\frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{x \in \mathcal{L} \cup \mathcal{U}} \mathcal{L}(f(x), f'(x)) \right) p(y'|f(x')) \cdot p_{x'} , \\ x_* &= \operatorname{argmax}_{x' \in \mathcal{U}} \mathcal{Q}_{\text{GP-}\Delta f\text{-MC-PDE}}(x') . \end{aligned} \quad (5.64)$$

where $p_{x'}$ denotes the Parzen density estimate for x' with respect to $\mathcal{L} \cup \mathcal{U}$ as defined in Eq. (5.50).

Rejections of Categorical Examples: Unknown or Unrelated: The previously introduced non-categorical examples could easily be filtered by inspecting their density induced by collected data. A second type of rejections, however, is less easy to detect and stems from the fact that either the oracle is uncertain about the actual class label, or the class label is not part of the predefined task. We refer to both

types of rejected examples as unknown and unrelated, respectively. Fang and Zhu (2012) referred to unknown examples as “blind spots” in a related investigation. With respect to our previously introduced EMOC criterion, we note that a rejected example does not result in any change of model outputs, irrespective of its estimated class label. Therefore, we can expand the original EMOC formulation given in Eq. (5.30) as follows:

$$\begin{aligned} \mathcal{Q}_{\text{GP-}\Delta f\text{-MC-R}}(\mathbf{x}') &= p(y' \neq r | \mathbf{x}') \cdot \sum_{y' \in \mathcal{Y}} \left(\frac{1}{|\mathcal{L} \cup \mathcal{U}|} \sum_{\mathbf{x} \in \mathcal{L} \cup \mathcal{U}} \mathcal{L}(f(\mathbf{x}), f'(\mathbf{x})) \right) p(y' | f(\mathbf{x}')) \\ &\quad + p(y' = r | \mathbf{x}') \cdot 0 \\ &= (1 - p(y' = r | \mathbf{x}')) \cdot \mathcal{Q}_{\text{GP-}\Delta f\text{-MC}}(\mathbf{x}') . \end{aligned} \quad (5.65)$$

It now only remains to estimate $p(y' = r | \mathbf{x}')$, which both depends on the task defined by $\mathcal{Y}$ (unrelated examples) and the current oracle itself (unknown examples). In absence of prior knowledge, we suggest to learn probability estimates over time by treating $p(y' = r | \mathbf{x}')$ as a binary classification problem. Following our previous derivations, we train a GP label regression model learned with all previously rejected examples as positives and all remaining labeled examples as negatives. Probability estimates can again be obtained from regression scores by applying Eq. (5.23). Note further that the learned model is also trained to be sensitive for non-categorical examples.

If we combine the techniques for rejecting non-categorical and categorical examples, we finally obtain our most complex EMOC variant denoted by $\mathcal{Q}_{\text{GP-}\Delta f\text{-MC-PDE-R}}$.

6 | Learning of Mid-level Features

“What makes this object class different from pretty similar ones?”

In previous chapters, we were concerned with learning models from large datasets, extending learned models incrementally, and actively selecting informative examples for manual annotation. For keeping a clear focus, we assumed that representations of images were given and fixed. However, image representations which are fixed in advance might not be appropriate in all scenarios. In this chapter, we will therefore investigate techniques to automatically identify features for discriminating between different object categories and to focus feature extraction to meaningful object parts.

Contributions of This Chapter: We will start with an overview about existing approaches for feature learning in Section 6.1, covering the areas of codebook learning, deep learning, and the discovery of mid-level features. Furthermore, we will give a short introduction to fine-grained recognition which will serve as application scenario for the presented techniques. In Section 6.2, we then introduce our approach for (un)supervised patch discovery. Our technique allows for discovering mid-level patterns which re-occur in collected data, *e.g.*, a striped wing or a dotted belly. An exemplar-specific extension of this approach is presented in Section 6.3. Thereby, we obtain compact and informative representations specifically learned for every new test image on-the-fly. In Section 6.4, we finally present how part annotations obtained from human experts can be transferred from training images to unseen test images with a simple yet effective non-parametric method.

6.1 Related Work on Feature Learning

In order to reliably differentiate between object categories, the identification of relevant features is among the most crucial aspects, especially if categories are visually similar. Difficulties mainly arise from two aspects. On the one hand, objects can show strong variations of pose or viewpoint, which especially holds for

highly deformable objects such as cats and dogs (Parkhi et al., 2011) or birds (Wah et al., 2011b). On the other hand, objects can exhibit strong variations of appearance within an object category, *e.g.*, colors may significantly differ between male and female birds of the same species. Both observations impose the necessity of discovering and localizing features which are truly distinctive.

Although pose-alignment techniques can reduce effects based on significant pose variations (Farrell et al., 2011; Zhang et al., 2013), they still have to struggle with the identification of discriminative features to encode the yet pose-aligned objects. Early papers used off-the-shelf features such as bag-of-visual-word statistics (see Section 2.2.2) and more recent approaches further exploit human annotations to focus feature extraction to informative regions (Deng et al., 2013; Duan et al., 2012). Latest trends, however, suggest that *learning* of relevant features is possible, even with little or no human interaction. Thereby, classification performance can be significantly boosted compared to hand-designed representations. In the following, we will review three prominent ways for learning image representations: (i) codebook learning to replace the ad-hoc choice of unsupervised K-means clustering in bag-of-words approaches (Section 6.1.1), (ii) deep-learning architectures which learn classifiers and representations jointly (Section 6.1.2), and (iii) mid-level features such as parts or attributes which often allow for rich semantic interpretations (Section 6.1.3).

6.1.1 Codebook Learning

The most popular choice for image encryption within the last decade was probably the extraction of local features (see Section 2.2.1) and the computation of statistics thereof, *e.g.*, by following the bag-of-words principle (Csurka et al., 2004) as reviewed in Section 2.2.2. Although the actual choice of local features heavily varied over time, statistics have usually been computed by unsupervised clustering of local features in training using the easy-to-use K-means algorithm (Bishop, 2006, pp. 424), and hard assignment of local features to closest prototypes (see Eq. (2.137)).

We have already seen in Section 2.2.2 that K-means clustering minimizes reconstruction error and thus results in prototypes close to frequently occurring features. However, frequently occurring features are not necessarily discriminative. In fact, Jurie and Triggs (2005) already showed that unsupervised K-means clustering often leads to results worse than random sampling of cluster centers. As a direct consequence, several techniques have been developed incorporating additional aspects such as optimal assignments or discrimination abilities into the process of codebook learning rather than solely focusing on small reconstruction error for hard assignments of training data to prototypes.

6.1.1.1 Unsupervised Codebook Learning

In contrast to minimizing reconstruction error with the constraint of hard assignments, sparse coding techniques (Lee et al., 2006; Mairal et al., 2009; Olshausen and Field, 1997) aim at assignments which are sparse but can indeed obtain several non-zero entries. Following our notation from Section 2.2.2, this translates to:

$$(\hat{C}, \{\hat{v}_i\}) = \underset{C \in \mathbb{R}^{D \times \mathbb{R}}, \{v_i\}}{\operatorname{argmin}} \sum_{i=1}^{\mathcal{N}} (\|d_i - C v_i\|_2 + \lambda \|v_i\|_1) \quad , \quad (6.1)$$

where λ is a regularization parameter (compare with Eq. (2.140) which includes a fixed constraint on v_i). From this point of view, K-means is an extreme version of sparse coding and it is, in direct comparison, faster but less powerful for discrimination of categories (Coates and Ng, 2011).

Similar in spirit is the utilization of Gaussian mixture models (see Section 2.1.4.2) as done in (Coates et al., 2011; Csurka and Perronnin, 2010; Farquhar et al., 2005; Perronnin et al., 2006), which results in using a cluster-specific Mahalanobis distance instead of the Euclidean distance frequently used in K-means. Thus, GMMs can be interpreted as a generalization of K-means clustering with data-driven distance metrics and soft assignments during estimation. Nonetheless, both versions are usually estimated using the expectation-maximization principle (Bishop, 2006, pp. 435). Jia et al. (2013) propose a codebook refinement strategy which first computes an overcomplete K-means codebook from which a smaller set of centers is selected by pooling correlations between clusters.

An alternative choice for codebook creation is the utilization of topic models, and Larlus and Jurie (2009) showed that replacing K-means with a topic model improves performance especially when the number of training examples is small. In addition, further techniques such as sparse auto-encoders or restricted Boltzmann machines are known too (see (Coates et al., 2011) for an example), although rarely used in bag-of-visual-words applications.

Interestingly, Coates et al. (2011) provided an in-depth comparison of unsupervised techniques introduced before, with the outcome that the choice of clustering parameters is often more critical than the choice which clustering method to apply.

6.1.1.2 Supervised Codebook Learning

Moosmann et al. (2006) proposed an easy-to-use alternative to unsupervised clustering by learning random decision trees or forests from local features. BoW statistics can then obtained by counting how many features reach every leaf node. Since training of random forests is done by optimizing information gain in every node, the clustering is discriminative *per se*.

An implicit way for incorporating class statistics into codebook learning was presented by Perronnin (2008). The author proposed to concatenate a universal codebook with class-specific counterparts. Mairal et al. (2008) show how to extend the common sparse coding scheme towards discriminative sparse coding. Learned class-specific codebooks minimize overall reconstruction error and simultaneously achieve especially smaller errors for patches from the corresponding class. In (Yang et al., 2008), the authors combine codebook learning and classifier training by iteratively alternating optimization of each aspect. Mairal et al. (2009) show how to combine reconstruction error of sparse coding and discrimination error of linear or bilinear models into a joint optimization framework and extend their analysis in (Mairal et al., 2012).

Similar in spirit to the unsupervised version of Jia et al. (2013), Winn et al. (2005) propose to iteratively select a subset of a large codebook by fusing codewords that hardly contribute to current discrimination results. Later, Yang et al. (2007) introduce an extended SVM formulation to automatically select co-occurring clusters and select those ones relevant for classification. Lazebnik and Raginsky (2009) propose to learn prototype assignments which minimize the loss of mutual information between resulting bag-of-words histogram and class labels.

While previously reviewed techniques can offer significant performance gains, they still depend on specified low-level features. Additionally, learned representations are usually not human-understandable, making trained models often hard to interpret.

6.1.2 Joint Learning of Features and Classifiers: Architectures for Deep Learning

We already motivated in Section 2.2.3 that the previously common pipeline of local feature extraction and bag-of-words assignments has been replaced recently by deep-learning architectures. A deep introduction to this area would go far beyond the scope of this thesis. Thus, we only give a short overview of recently popular concepts.

The Idea of Deep Learning: The underlying idea of deep learning for arbitrary applications could be summarized as follows:

- (i) collect huge amounts of data representative for the application domain,
- (ii) specify a cost function (*e.g.*, classification error),
- (iii) specify a model with a large number of tuneable parameters that operates directly on original inputs (*e.g.*, images), contains multiple processing stages, and outputs desired target values (*e.g.*, class labels), and

- (iv) train all involved parameters jointly using a suitable optimization technique.

Note that the combination of the last two aspects gives rise to the methods name as deep learning. The processing stages are commonly understood as implicitly learned representations of increasingly complexity. Current techniques by Zeiler and Fergus (2014), Mahendran and Vedaldi (2015), and Yosinski et al. (2015) for visualizing the learned connections in deep neural networks provide evidence for this interpretation.

Backpropagation for Training Deep Models: We already introduced deep neural networks briefly in Section 2.2.3. Inspired by hypotheses about signal processing in the human brain, individual units in neural networks process inputs as weighted summation of individual activations. According to the previously presented idea of deep learning, deep neural networks consist of cascades of such linear and non-linear operations which are parameterized with millions of parameters. Similar to the classification techniques introduced in Section 2.1.4, deep neural networks can be trained by specifying a parameterized architecture, collecting labeled data, and selecting a loss function. Based on a given initialization for all parameters, iterative optimization techniques can be applied to adjust the networks parameters such that the resulting loss is minimized. Note the similarity to Section 3.2.3.1 where we needed to solve linear equation systems iteratively.

As iterative technique, the *back-propagation* algorithm is probably today's most commonly used method. Backpropagation is based on gradient descent, which adjusts the currently estimated parameters towards the direction where the gradient of the loss function with respect to parameters is largest. However, the output of a deep neural network results from the concatenation of several parameterized functions. To still be able for applying gradient descent successfully, Rumelhart et al. (1986) presented how to apply the chain-rule for propagating errors from final layers to earlier layers (an earlier version of this approach was presented in (Rumelhart et al., 1985)). This approach gives rise to the methods name back-propagation. Note that also precomputed networks can be adjusted to new training data by running the same training procedure and use the previously found parameters as initialization. This process is also known as *fine-tuning*.

We skip a more detailed revision of involved concepts here. Instead, we refer the interested reader to the groundbreaking paper by LeCun et al. (1998) where individual aspects of gradient-based learning using backpropagation for (convolutional) neural networks are presented in detail. Furthermore, a detailed introduction on backpropagation and learning of deep neural networks can be found in the inspiring tutorial of Vedaldi and Zisserman (2015)

6.1.3 Discovery of Mid-level Features

In contrast to previous directions of feature learning, mid-level features are usually human-interpretable and topics mainly cover learning of attributes, parts, and patches. We already gave a brief introduction to parts and attributes for computer vision in Section 2.2.4. Here, however, we will provide a more detailed review since our techniques presented later in this chapter have been influenced by approaches of this field.

Learning of Attributes: For binary-valued attributes such as “is striped”, the probably most intuitive way for learning an attribute predictor is by viewing the task as a classification problem. Thus, we require labeled data which specifies the presence or absence of binary-valued attributes in each image, and binary classifiers are trained for each attribute independently. Lampert et al. (2009b) propose to learn human-specified binary attributes to detect unseen object classes where attribute-descriptions are available, but training images are not (also called zero-shot learning). Attribute predictors are either learned directly from image data (direct prediction), or from class probabilities computed for known classes (indirect prediction). Wah and Belongie (2013) extend this approach to detect also unfamiliar classes, *i.e.*, unseen classes where also class-attribute relations are not available, and further show how to incorporate users to assist in attribute prediction (see Chapter 5). Su et al. (2010) show that hand-defined attributes such as color, object shape, or scene properties can boost performance when added to low-level features for image encoding. Donahue and Grauman (2011) additionally request rationales of human annotators to explain choices of (binary) attribute labels. Obtained rationales are then used to train attribute predictors more robustly.

Another line of research first discovers discriminative features for differentiating between classes, and prunes in a second step those features which are not human-understandable, mostly by having a human in the loop (see Chapter 5). Parikh and Grauman (2011) show how to iteratively learn discriminative splits of data which are consistent with Flickr tags, and accept a split as attribute if the decisions are human-understandable. Duan et al. (2012) propose a latent CRF model similar to a multi-instance learning framework for learning localized, discriminative attributes and keep only human-understandable discovery results.

A third line of research covers learning of optimal separations of classes without enforcing semantic consistency. Learned splits are consequently referred to as *discriminative attributes* (Choi et al., 2013; Rastegari et al., 2012; Torresani et al., 2010; Yu et al., 2013). Due to the lack of semantics, however, the resulting pipeline is rather similar to boosting approaches (Freund and Schapire, 1997; Schapire, 1990) than to *semantic attribute* encodings as reviewed previously. Farhadi et al. (2009) show how to combine both semantic and discriminative attributes to finally obtain

rich descriptions of objects.

Learning of Parts: Humans often use quantities and relations of small objects to describe composed objects. We can explicitly address this observation by learning to predict part occurrences and constellations, and to compare objects by focusing on quantities and appearances of visible parts. Note that naturally, representations using part information are closely connected to attributes introduced before.

Among the earliest work on detecting and representing objects with parts is the paper by Fischler and Elschlager (1973) which introduces a hand-crafted connection model. Later on, Felzenszwalb and Huttenlocher (2005) presented efficient matching techniques for pictorial structures, which ultimately lead to the DPM models as reviewed in Section 2.2.4.

For data where objects are additionally annotated with part positions, Berg and Belhumeur (2013) proposed to learn part-specific one-vs-one features (POOF) to discriminate between two similar classes based on informative part information. Bourdev and Malik (2009) introduced poselets as discriminative human parts in $D = 2$ which allow for estimating a person's pose in $D = 3$. Zhang et al. (2013) present deformable part descriptors which are especially suited to capture the pose of birds. Later on, Zhang et al. (2014) showed how to exploit CNN models to extract powerful representations from localized head and body regions of birds. Recently, Simon and Rodner (2015) introduced a technique to discover part constellations in an unsupervised manner where parts correspond to previously learned filters of a given CNN model.

Discovery of Patch Descriptors: In contrast to parts, patches refer to consistently re-occurring patterns of objects which are not necessarily nameable by humans. Two examples are a characteristic constellation of freckles of a person or specific fur shadings. We will use the notation of *patches* for blocks or sub-windows in images showing those characteristic patterns, and models trained for detecting a group of consistent patches will be called *patch detectors*. Since nameability of patches is not always given, patches are usually discovered rather than specified by human.

Patch discovery has been an important research field since the early works of Agarwal and Roth (2002) and Vidal-Naquet and Ullman (2003). Whereas Agarwal and Roth (2002) cluster similar patches of training images to obtain a vocabulary in an unsupervised manner, Vidal-Naquet and Ullman (2003) find class-specific patches by maximizing mutual information between appearance and class indicator. Both approaches (and related ones during the same time) use simple detectors based on gray values, which are hardly able to tackle the variability in natural images.

The paper of Singh et al. (2012) was the first one to present a patch discovery

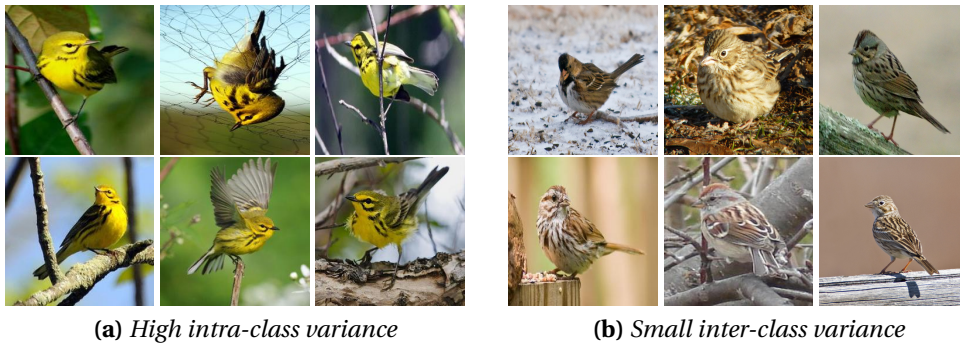


Figure 6.1: Visualization of high intra-class variance and small inter-class variance in fine-grained recognition scenarios. Left: Six images showing a prairie warbler exhibiting strong variations in viewpoint and pose – can you guess the main characteristics for identifying this bird? Right: Six images of five different sparrow species living in North America – can you tell which two birds belong to the same species? Answers for both questions are given in Fig. 6.2. All bird images used in this chapter are originally part of the Caltech-UCSD Birds-200-2011 (CUB-2011/200) dataset by Wah et al. (2011b).

method that made use of recent advances in object localization, such as HOG features and Exemplar-SVM models. The usefulness of patch discovery schemes for differentiating between visually similar classes has been demonstrated by Lee et al. (2013). The work of Doersch et al. (2013) show how to cast patch discovery as a non-convex optimization problem related to mode seeking.

A common drawback is the time-consuming step of model learning with hard negative mining (Singh et al., 2012), and Juneja et al. (2013) present a time-efficient version using whitened HOG features (Hariharan et al., 2012). Their patch discovery scheme is based on bootstrapping a set of initial seed patches, which are previously derived from an unsupervised segmentation result, similar to our technique presented in Section 6.2. Finally, note that deep-learning approaches (see also Section 6.1.2) such as (Zhang et al., 2014) and (Branson et al., 2014b) can be interpreted as fully supervised approaches for simultaneous patch discovery in multiple layers where resulting patch detectors are called filters.

6.1.4 A Side-note on Fine-grained Recognition

Within the last decade, research in visual object recognition mostly focused on category-level classification (*e.g.*, Everingham et al. (2010) organized one of the most important challenges of those days), and the techniques presented in Chapter 3 directly tackle this aspect. Although useful for coarse distinctions between object

classes, common approaches often fail in scenarios where the differentiation has to be done on a finer level. In these scenarios, only small inter-class differences exist and specific details matter. This area of research, which is referred to as subordinate or *fine-grained recognition*, has become recently popular in our community (e.g., (Branson et al., 2014b; Farrell et al., 2011; Wah et al., 2011a; Zhang et al., 2014)).

Fine-grained recognition offers a variety of challenging applications, such as recognition of dogs (Liu et al., 2012), plants (Kumar et al., 2012a), shoes (Yu and Grauman, 2014), airplanes (Maji et al., 2013), or birds (Wah et al., 2011b) – tasks even difficult for human experts.¹ Resulting problems are often two-fold: on the one hand, objects within a single class usually offer high intra-class variations, e.g., due to varying poses or different appearances over time. On the other hand, different tasks are often visually similar, i.e., the resulting inter-class variance is low, and it requires expert knowledge to reliably tell two species apart. A visualization of both phenomena is given in Fig. 6.1². We additionally show in Fig. 6.2 how similar species can be distinguished based on unique appearances of specific object parts. Thereby, we also answer the questions raised in Fig. 6.1.

We can conclude that fine-grained recognition tasks – in contrast to recognition at coarse class levels, such as elephant versus giraffe – can only be tackled by identifying small differences at certain positions, such as color, pattern, shape, or length of a bird's wing. Thus, we specifically motivate the discovery of mid-level features within this scenario. In detail, we will focus on the discrimination of birds as our running example throughout this chapter. Nonetheless, the techniques developed are generic and could be applied to other fine-grained classification task equally well.

6.2 Learning of Mid-level Representations

As mentioned before, a lifelong learning system should ideally be able to not only update classification models over time, but to update also image representations if needed. Thus, an (un)supervised discovery of relevant features seems an ideal building block within the lifelong learning cycle (see Fig. 1.3). In this section, we present our patch discovery scheme published in (Freytag et al., 2014a) which offers three advantages: (i) the resulting representations are human-interpretable since detectors are trained from interpretable patches rather than from low-level statistics (see Fig. 2.13 in Section 2.2.2), (ii) the set of discovered patch detectors can easily be extended when new training data is available as in lifelong learning scenarios,

¹See the taxonomic recommendations for birds of the Association of European Records and Rarities Committees (2003) as an impressive example for difficulties in categorization of species.

²Note that all images of birds in this chapter are originally taken from the CUB-2011/200 dataset (Wah et al., 2011b).



Figure 6.2: Answers for questions in Fig. 6.1. Left: *Prairie warbler* exhibit olive-green upperparts, yellow throats and bellies, and have black streaks on face, sides and flanks. Right: *Vesper sparrows* have a unique white eyering besides the common streaked chest and back. Top row middle image and bottom row right image in Fig. 6.1b both show a *vesper sparrow*. Details of bird species are according to (Cornell Lab of Ornithology, 2015).

and (iii) no human annotations are required during the discovery process.

As we will see in the following, three main parts need to be tackled: (1) finding initial seed patches as proposals, (2) learning exemplar-like patch detectors from these proposals, and (3) convolution-based bootstrapping of the learned patch detectors. The discovery can be done supervised or unsupervised and is followed by a feature extraction step where we generate features by spatially pooling patch detector outputs. Within this chapter, we use the terms *patch* and *block* interchangeable and denoted with x when referring to a small region or window in an image. Furthermore, a template or linear classifier w learned with a given set $\mathcal{M}(w)$ of patches will be called *patch detector*. In the following, we will go through each step in detail and motivate the chosen techniques.

6.2.1 Identifying Region Candidates for Mid-level Features

Finding initial seed patches is important to guide the following bootstrapping steps in the right direction. The purpose is similar to an interest point detector, which was often used in the earlier works on bag-of-visual-words (see (Jurie and Triggs, 2005) and references cited therein). The main pipeline of our proposed seeding technique is visualized in Fig. 6.3. Furthermore, we show qualitative results in Fig. 6.5.

Starting with a given image, we already assume a rough focus on the actual object of interest, *i.e.*, the image is cropped to a slightly enlarged bounding box which is usually available or can be estimated with robust object detection techniques (Felzenszwalb et al., 2010). To further focus seed patches towards foreground locations only, we mask background regions in step (1) by using either a pixel-wise annotation if available or running GrabCut (Rother et al., 2004) to robustly estimate

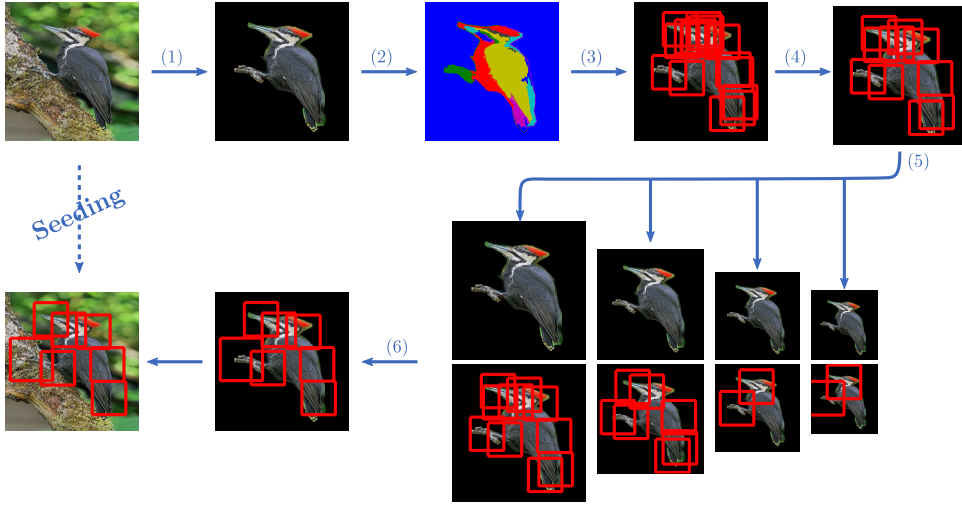


Figure 6.3: Overview of our seeding step: as input serves an image with focus on the object, i.e., cropped to its bounding box. In the following steps, we focus stronger on the object by masking background pixels (1) and run an unsupervised segmentation (2) resulting in proposed blocks (3). After proposal cleaning (4), the procedure is repeated on multiple scales (5), and all results are finally fused (6).

foreground pixels (see Fig. 6.4 for a visualization of both techniques). Although this is only an optional step, it helps in guiding patch discovery to parts located on the object of interest (see results in Fig. 6.5). Thus, the number of discovered patch detectors corresponding to background structures is decreased, which is likely beneficial since these patches are at most contextual related (e.g., surrounding leaves, rocks, or branches). Simultaneously, the ratio of meaningful detectors firing at the object of interest is increased.

Let us now follow the pipeline as shown in Fig. 6.3. In step (2), we split the object into sub-regions using the unsupervised region segmentation method by Felzenszwalb and Huttenlocher (2004). Following the idea of Juneja et al. (2013), we place quadratic patches on the centers of all previously computed regions and clean up non-intuitive results, i.e., we discard regions that are too small, too large, or too skewed. Additionally, a non-maximum suppression-like filtering further discards regions overlapping too strongly with each other (step (4)). In order to add size invariance, we perform the outlined scheme on several resolutions of the input image (step (5)). Finally, results from multiple scales are combined by again running a non-maximum suppression-like filtering on all retrieved regions in step (6). Exemplary results after applying all steps are displayed in Fig. 6.5. Note that in direct comparison with human annotations of informative keypoints, our proposals are less redundant in head regions, while at the same time covering most of the visually informative aspects.

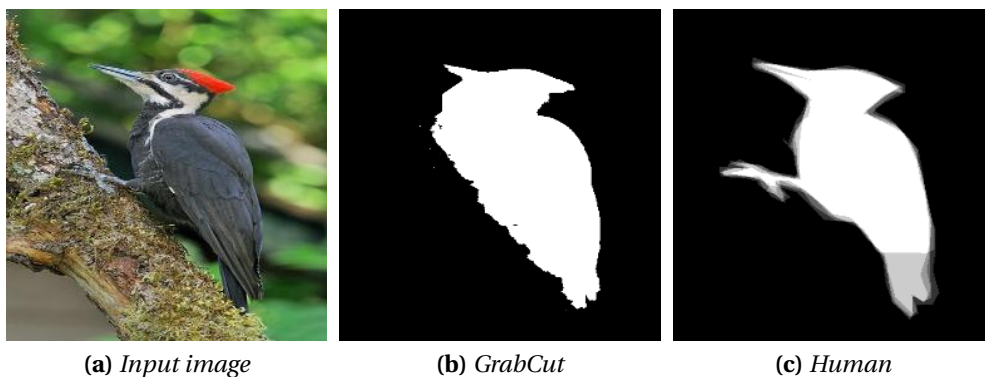


Figure 6.4: Background masking of an input image using GrabCut (Rother et al., 2004) or human annotations (averaged over multiple annotators) as provided by the CUB-2011/200 dataset.

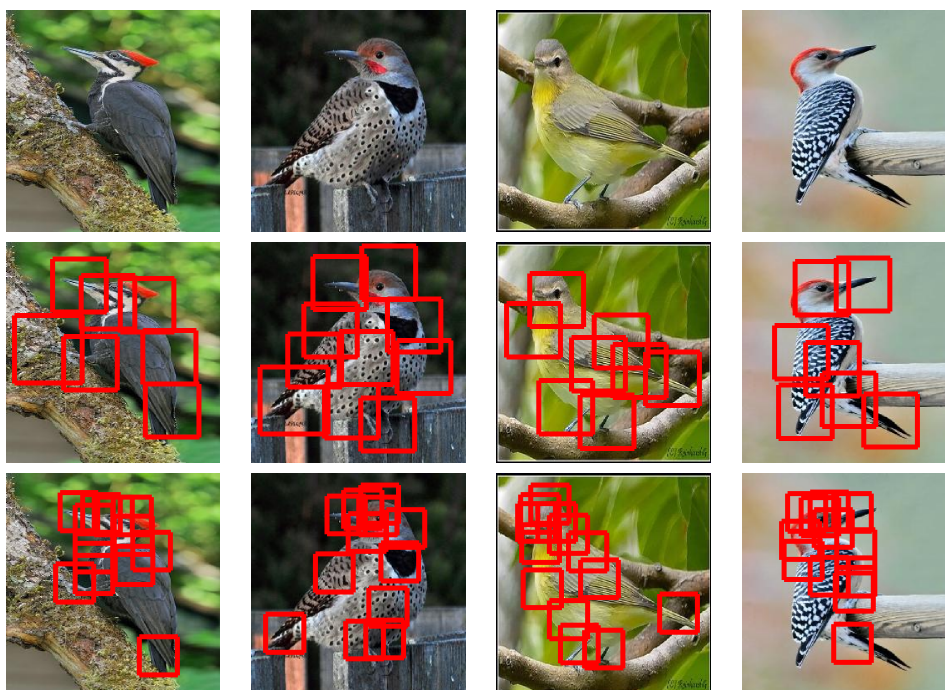


Figure 6.5: Results of the proposed seeding for unsupervised patch discovery. Top: Three input images. Middle: Resulting regions serving as proposal seeds for training patch detectors later on. Bottom: Human annotations of 15 informative key points, e.g., beak, crown, or tail.

6.2.2 From Region Candidates to Patch Detectors

Given the seeding blocks from the previous section, we now aim at learning patch detectors from these regions. Ideally, a patch detector should fire on an image if and only if that image contains the specific region the detector was trained with, *i.e.*, the detector should discriminate between its patch and *everything else*. Viewed as a machine learning problem, this would intuitively lead to training a classification model from a single positive example and millions or even billions of negatives. Note that with “negative examples”, we refer to all possible sub-windows *not* showing the specific patch its corresponding detector was trained with. Although seeming ill-conditioned on first glance, the Exemplar-SVM idea presented by Malisiewicz et al. (2011) shows that this might actually be possible, and we adapt their idea here.

Representing Individual Patches: Before turning on training patch detectors, however, we first of all need to know how to best represent a single patch. Since our running example is recognition of birds, we are inspired by what humans would do, who frequently use a mixture of typical color and texture occurrences in order to describe a bird's appearance, *e.g.*, a dotted red belly or feathers with blue stripes. We meet this observation by representing patches by histogram of oriented gradients (HOG) features and color name (CN) feature histograms computed for small cells of pixels (see Section 2.2.1 for an introduction to local features). Note, however, that the method is not restricted to this particular choice of features, and incorporating latest trends for image representations using outputs of CNNs likely increases the reliability of learned patch detectors further.

Models for Patch Detection: As detector model, Singh et al. (2012) suggested a linear SVM (see Section 2.1.4) due to its powerful discrimination abilities in high dimensional data and off-the-shelf solvers publicly available. However, SVM training for object detection usually requires computationally expensive hard negative mining (Felzenszwalb et al., 2010). Therefore, Juneja et al. (2013) proposed to use LDA as linear model instead (see Section 2.1.4 for an introduction to LDA). As shown by Hariharan et al. (2012), training object detectors on HOG features can be done efficiently using standard Gaussian assumptions of LDA. Although only presented for HOG features, their technique can be applied to arbitrary features such as the combination of HOG and ColorName features.

Let us consider a single filter w that represents a classifier which differentiates between the selected positive patch and all negative examples. According to Hariharan et al. (2012), applying an LDA model assumes positive and negative examples to be Gaussian-distributed with the shared covariance matrix Σ_0 and mean vector μ_1 and μ_{-1} , respectively. As reviewed in Section 2.1.4, the resulting hyperplane

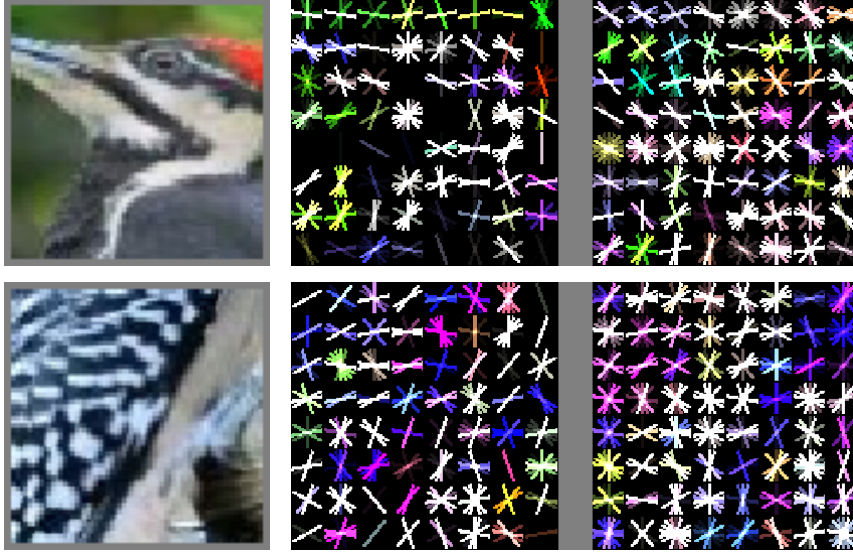


Figure 6.6: Learned patch detectors from single positive examples. Left: Two patches as output of the previous seeding step. Right: Visualization of learned weight vectors from HOG and ColorName features for patch detectors using an LDA model as described in Section 6.2.2. Left and right part show positive and negative entries of w , respectively.

which separates positives and negatives is given by:

$$w = \Sigma_0^{-1} (\mu_{-1} - \mu_1) . \quad (6.2)$$

Properties of LDA Models for Patch Detection: Arguable, the underlying assumptions when applying LDA in Eq. (6.2) might seem unrealistic. This especially holds for the assumption of a covariance matrix which is shared among a single positive and millions of negative examples. However, we argue that the above formulation can also be interpreted as feature whitening, since it implicitly decorrelates all features using statistics of a large set of (negative) examples. Hariharan et al. (2012) suggested that this aspect is likely to be highly relevant to deal with high correlations naturally arising in feature extraction, *e.g.*, between neighboring HOG cells.

Since every detector discriminates a positive patch against everything else, the notion of “everything else” can be shared by all detectors. In terms of LDA models, the covariance matrix Σ_0 as well as the mean μ_{-1} of negative examples can be used for all patch detectors once estimated from an arbitrary set of background images and features calculated therein. As a consequence, we can easily pre-compute it and re-use it for every patch model.

In summary, the only remaining steps for learning a new patch detector from a

given example is to compute the feature of the positive patch and to solve the linear equation system. For detectors with multiple positive examples – as needed in the next section – the only additional step is the averaging of features computed from all positive examples to obtain μ_1 .

6.2.3 Convolutional Unsupervised Bootstrapping of Patch Detectors

After passing all steps of the previous section, we obtained a trained patch detector for each initial seed patch. However, detectors trained from a single positive example are unlikely to generalize well. To slightly increase their generalization abilities, we proceed with expanding the set of positive examples. While this could be done manually by searching for similar image patches in images of the same category, resulting annotation costs would be unreasonably high. We therefore aim at an automated search for sub-images similar to the initial seed patches to include the most best-ranked responses into the training set. This process of (unsupervised) expansion is also referred to as bootstrapping³ (Felzenszwalb et al., 2010; Huang and Riloff, 2012; Riloff et al., 1999; Rowley et al., 1995; Sung and Poggio, 1994).

Convolutions of Detectors and Feature Planes: To discover sub-images which likely correspond to the initially trained patch detector, a complete search over all possible locations and scales in all available images would be required. Let x be the feature vector extracted from a training image for one possible combination of position and scale. Since our patch detectors are linear models as shown in Eq. (6.2), the response on x is an additive combination of dimension-specific scores:

$$w(x) = \langle w, x \rangle = \sum_{d=1}^D w(d) \cdot x(d) \quad . \quad (6.3)$$

Evaluation of w on all possible locations can then be done in a sliding window manner by computing Eq. (6.3) for densely extracted features. In this case, we can also change the order of computations and can equivalently compute Eq. (6.3) by adding D convolutions of 1×1 filters with corresponding feature planes. For the aforementioned choice of HOG and ColorName as feature types, however, we note that features are computed independently for small cells and, thus, x follows a spatial tiling composed of $T \times T$ cells with D_C feature dimensions per cell. Since

³The term *bootstrapping* is often used for only mining hard negatives, where negative training data is expanded with informative examples. This mainly comes from the fact that for detection, the number of negative examples is usually much higher than the number of positives. Thus, only a fraction of negative examples is likely to be valuable, e.g., as done by (Rowley et al., 1995; Sung and Poggio, 1994) for bootstrapping of face detectors.

the final feature vector is obtained by concatenating all cells, *i.e.*, $D = T \cdot T \cdot D_C$ in Eq. (6.3), the evaluation of a detector in Eq. (6.3) translates to:

$$f(\mathbf{x}) = \sum_{d_i, d_j=1}^T \sum_{c=1}^{D_C} \mathbf{w}((d_j T + d_i) D_C + c) \cdot \mathbf{x}((d_j T + d_i) D_C + c) . \quad (6.4)$$

As we can see, we can again exchange the order of summations which leads to adding results of D_C convolutions of $T \times T$ filters with corresponding feature planes. Thus, we can perform an exact search by computing convolutions of $\mathbf{w}$ and corresponding feature planes computed from all images. For further computational speed-ups on discrete convolutions, we refer the interested reader to (Dubout and Fleuret, 2012). In direct consequence, our method allows for discovering important patches from detection responses, also if they have not been found by the unsupervised segmentation in the beginning.

Bootstrapping with Data-driven Thresholds: Given the detection responses, we can select corresponding patches to increase the training set size of every detector. In standard expansion techniques (Juneja et al., 2013; Riloff et al., 1999; Sharma and Nevatia, 2013), we would add a fixed number m of the highest scored examples in each iteration if their score is larger than a pre-defined specificity parameter θ :

$$\mathcal{M}^{t+1}(\mathbf{w}^k) = \text{top}_m \left(\left\{ \mathbf{x} \mid (\mathbf{w}^k)^T \mathbf{x} > \theta \right\} \right) \cup \mathcal{M}^t(\mathbf{w}^k) , \quad (6.5)$$

with $\mathcal{M}^{t+1}(\mathbf{w}^k)$ denoting the set of possible examples in iteration $t + 1$ for detector k . However, outputs of independently trained classifiers are usually not directly comparable (Platt, 2000). Thus, the existence of an applicable global threshold is an assumption too strong in practice. In consequence, bootstrapping would result either in too few or too many returned examples, which leads to weak generalization or weak discrimination power, respectively.

To meet this observation, we propose detector-specific thresholds by adding *at most* m examples in every iteration. In particular, we do not add examples that received a detection score worse than a percentage γ of the minimum score that has been achieved by examples which are already used as positive training examples:

$$\mathcal{A}^{t+1}(\mathbf{w}^k) = \{ \mathbf{x} \mid (\mathbf{w}^k)^T \mathbf{x} > \gamma \cdot \min_{\tilde{\mathbf{x}} \in \mathcal{M}^t(\mathbf{w}^k)} (\mathbf{w}^k)^T \tilde{\mathbf{x}} \} . \quad (6.6)$$

Here, $\mathcal{A}^{t+1}(\mathbf{w}^k)$ is the set of acceptable patches for detector $\mathbf{w}^k$ in iteration $t + 1$ and γ controls the exploration/exploitation trade-off during discovery. Intuitively, larger values for γ prevent patch detectors from being “blurred” during bootstrapping by outliers.

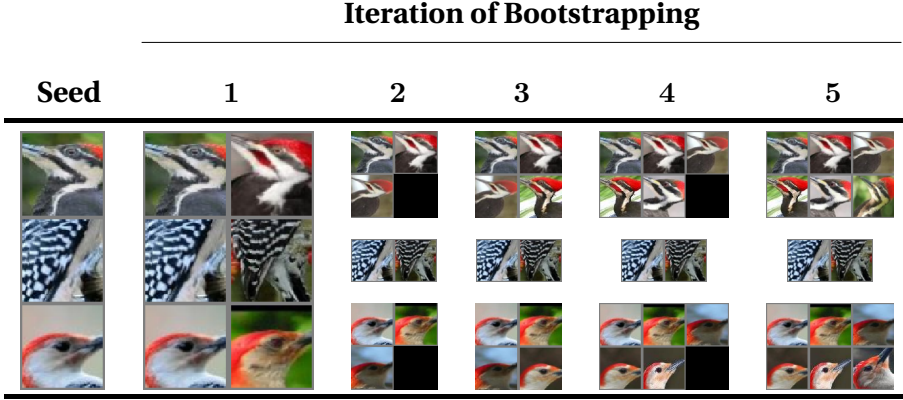


Figure 6.7: Results of bootstrapping patch detectors for at most 5 iterations. In every iteration, only a single new example is added if available ($m = 1$). Whereas the second detector converged already after one iteration, i.e., $\mathcal{A}^{t=2}(\mathbf{w}^2) = \text{empty}$, bootstrapping of $\mathbf{w}^1$ and $\mathbf{w}^3$ individually converged significantly later while obtained positive examples are still highly consistent.

If $\mathcal{A}^{t+1}$ is empty, we consider $\mathbf{w}^k$ as converged. Otherwise, we select m examples of $\mathcal{A}^{t+1}(\mathbf{w}^k)$ with highest score and add them to the set of positive examples:

$$\mathcal{M}^{t+1}(\mathbf{w}^k) = \text{top}_m(\mathcal{A}^{t+1}(\mathbf{w}^k)) \cup \mathcal{M}^t(\mathbf{w}^k) . \quad (6.7)$$

The process of bootstrapping a detector for at most five iterations is visualized in Fig. 6.7, where we set $m = 1$ for illustration purpose. Note that adaptive thresholds are also used in the natural language processing domain (Huang and Riloff, 2012).

The previously described process from seeding (Section 6.2.1) over detector training (Section 6.2.2) to bootstrapping (Section 6.2.3) runs fully unsupervised by default, i.e., no class labels or keypoint annotations are required in any step. However, if class information is available, we can indeed incorporate it into the bootstrapping process. For a supervised version, all we need to do is to restrict the search for similar sub-images to examples of the same category the initial seed patch was extracted from (Doersch et al., 2013; Juneja et al., 2013; Singh et al., 2012). Visualizations for the results of supervised and unsupervised bootstrapping are given in Fig. 6.8. Initial seeding blocks for every detector are indicated with colored frames.

6.2.4 From Detector Responses to Image Representations

In the beginning of the chapter, we have been originally interested in describing images based on re-occurring, informative patterns such as striped wings or dotted

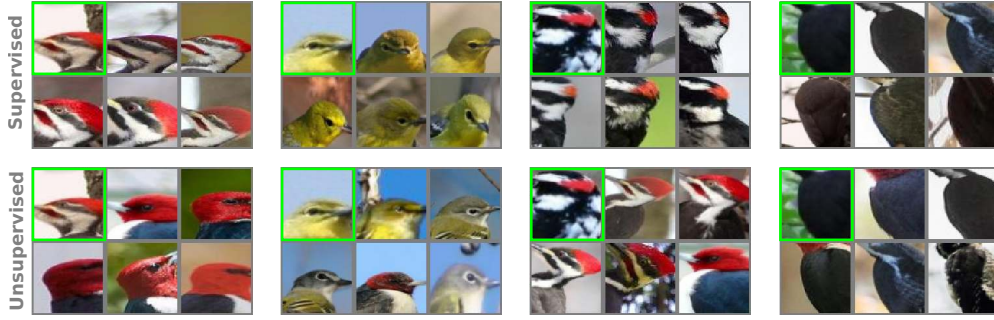


Figure 6.8: *Bootstrapping: initial seed patches are indicated with colored frames and the set of positive blocks for resulting detectors is displayed accordingly. While the technique is applicable to supervised and unsupervised scenarios, adding supervision prevents grouping of visually similar blocks from different categories.*

bellies. To this end, we have already seen how patch detectors can be discovered and learned from training images. Thus, it only remains to build a final image representation d from those detectors.

Note that each patch detector was trained on just a couple of training patches. In consequence, we can expect that it also fires only on an extremely small number of windows. We follow this observation and evaluate each patch detector w_i on every position p in all desired scales of an image. For every detector, we then take the largest response as indicator whether or not this specific patch was present in the current image. Similar to the notation introduced in Section 2.2.4, this operation can be expressed as:

$$d(i) = \max_p f_{w_i}(p) . \quad (6.8)$$

Due to our linear detection models (see Section 6.2.2), detection responses can again be obtained efficiently by convolving the weight vectors w_i of all learned models with computed feature planes of the image. Finally, max-pooling detection responses over the entire image leads to a feature vector with as many dimensions as previously discovered detectors. The resulting pipeline is visualized in Fig. 6.9. Note that more sophisticated pooling schemes such as the well-known pyramid matching of Lazebnik et al. (2006) can be easily applied, too.

Since detection results are computed via convolutions between feature planes extracted from the image and learned weight vectors of detector models, the overall pipeline shows an interesting parallel to deep learning techniques (see Section 6.1.2). In direct comparison, we fix the lower layers and – instead of learning planes associated with mid-level features by backpropagation – we instead bootstrap patch detectors which could be also used in unsupervised settings for deep architectures.

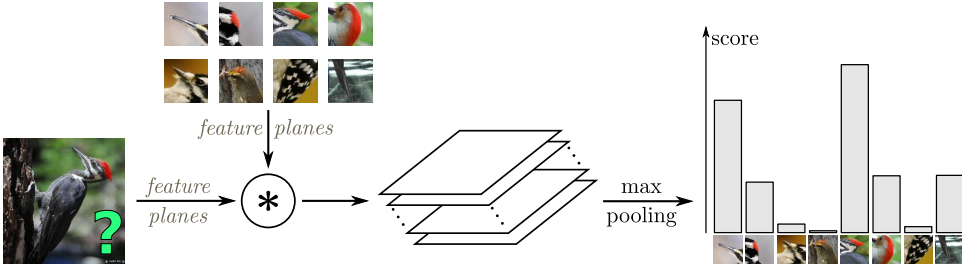


Figure 6.9: Image encoding with learned patch detectors: for a given image, feature planes are extracted and convolutions between feature planes and learned patch detectors are computed. Maximum pooling every convolution map over the entire image results in a score per detector where each score indicates the absence or presence of the corresponding detector within the image. Stacking responses of all patch detectors together gives the final representation.

6.2.5 Asymptotic Computation Times for Patch Discovery

Previous image representations were designed by smart engineers, *e.g.*, (Bay et al., 2006; Calonder et al., 2010; Dalal and Triggs, 2005; Lowe, 2004; Rublee et al., 2011). In consequence, the main time for the development was spent by the developers themselves in their labs. However, when representations are learned instead of designed, significant amount of time is shifted from engineers to machine computations. Thus, it is inevitable to at least mention the asymptotic computation times which are required by the discovery process. Therefore, we briefly investigate asymptotic complexity for seeding, detector learning, and bootstrapping as outlined previously.

Computation Time for Seeding: As presented in Section 6.2.1, the seeding process is performed independently for every training image, and computation times are dominated by unsupervised segmentation. According to Felzenszwalb and Huttenlocher (2004), this requires $\mathcal{O}(n_{px} \log n_{px})$ steps with n_{px} being the number of pixels per image. Since seeding steps are run on focused objects, *i.e.*, sub-windows scaled to fixed size, n_{px} is constant for all inputs. Thus, the whole seeding requires $\mathcal{O}(N)$ operations when N images are given.

Computation Times for Training Detector Models: When training patch detectors, we first notice that we have as many detectors as region candidates obtained from the seeding step. Since the unsupervised segmentation can not produce more regions than pixels in a given image, the number of initial seed patches per image $p_{\mathcal{I}}$ is guaranteed to be smaller than the number of pixels n_{px} , *i.e.*, $p_{\mathcal{I}} < n_{px}$. Furthermore, we assumed that training images are scaled to fixed image sizes. Thus, we obtain $N \cdot n_{px}$ as an upper bound for the total number of patch detectors p . For

training each patch detector as LDA model, we have two major operations to be performed: averaging of positive examples and solving the linear equation system of Eq. (6.2). Note that computations of negative mean and covariance matrix can be shared by all detectors and can thus be pre-cached (Hariharan et al., 2012). We further note that the dimensionality of patch representations is fixed and the number m of added examples per iteration is bounded, too. Thus, both steps produce constant overhead in terms of asymptotic computation times, and training all patch detectors per iteration requires $\mathcal{O}(N)$ operations in total.

Computation Times for Patch Detection: During bootstrapping, we evaluate each patch detector in every training image by computing discrete convolutions of filter masks and feature planes. Let each filter mask be of size $T \times T$ and consist of D_C feature dimensions. Furthermore, let each image consist of not more than n_{px} pixels. According to Dubout and Fleuret (2012), computing the convolution of a linear model and an image's representation can then be done in $\mathcal{O}(T^2 n_{px} D_C)$ operations using standard implementations. Note that this could be speeded up further using efficient computations in Fourier space as well as cache-usage as shown by Dubout and Fleuret (2012). We further note that each cell covers a constant number of pixels in each dimension, thus, $T = \text{const}$. If we again use the upper bound for the number of patches, $p < n_{px} \cdot N$, we see that convolutions for a single image can be conducted in $\mathcal{O}(N \cdot n_{px}^2)$ time. For fixed image sizes n_{px} , this finally results in a total computation time of $\mathcal{O}(N^2)$ for bootstrapping.

Computation Times for Supervised Bootstrapping: Note that for the above consideration, we assumed fully unsupervised bootstrapping of patch detectors, *i.e.*, all detectors are convolved with all training images without considering from which class the seeding patch was originally extracted. In Section 6.2.3, however, we have already discussed that bootstrapping can also be guided when performed in a supervised manner. Doing so reduces the number of images every detector needs to be evaluated on to approximately $\frac{N}{M}$ given equally sized classes. Consequently, we arrive at $\mathcal{O}(N^2 \cdot M^{-1})$ operations for supervised bootstrapping. However, note that the asymptotic computation times stay within $\mathcal{O}(N^2)$ for arbitrarily imbalanced class distributions.

In summary, our proposed technique for (un)supervised patch discovery scales quadratically in the number of training images, *i.e.*, $\mathcal{O}(N^2)$ operations are required. From a practical point of view, computation times could further be reduced using efficient techniques during bootstrapping for finding correspondences, *e.g.*, approaches as introduced by Dean et al. (2013); Lampert et al. (2009a).



Figure 6.10: *Do we need to know how to differentiate between 20 sparrows, when we currently spot a woodpecker? Instead of learning fixed representations and models for all known examples, local learning techniques aim at exemplar-specific solutions focusing on species most similar to the current image of investigation.*

6.3 Exemplar-specific Representations via Local Learning

In the previous section, we were interested in describing objects based on distinctive patterns, such as striped wings or dotted bellies. With our presented technique, we are now able to learn these patterns from data in an (un)supervised manner, and to encode images based on the likeliness that these patterns are present in the current image of investigation. For lifelong learning scenarios, we can expect that the number of discovered patterns will grow over time to adequately represent new categories. Since only a fraction of them will be actually present in a single image, resulting representations will be sparse. In general, sparsity of representations is regarded as valuable for machine learning systems in general (Niemann, 1989, p. 9). However, confirming the presence or absence of an ever increasing set of discovered patches will sooner or later affect the computation time of our system negatively. We acknowledge this fact by asking the rather provocative question: *“Do we need to know how to differentiate between 20 sparrows, when we currently spot a woodpecker?”*

In this section, we argue that a valid answer might indeed be “No”. A visualization of the underlying motivation is given in Fig. 6.10. In addition, we draw inspiration from the famous words by Vladimir Vapnik:

*“When solving a given problem,
try to avoid solving a more general problem as an intermediate step.”*

(Vapnik (2000, Sect. 1.9))

Whereas previous methods aim at calculating a unique general representation which is able to differentiate all known categories as good as possible, we propose here to follow an orthogonal path and find for every unseen image a compact, informative, and image specific feature representation. The techniques presented are based on our findings published in (Freytag et al., 2014a,b).

6.3.1 Related Work on Local Learning

Structuring data into easier sub-problems is known for decades, and usually referred to as divide-and-conquer (Cormen et al., 2009, p. 30). For our running example of bird recognition, this relates to first roughly comparing the current bird (*e.g.*, a woodpecker) with birds already seen, and then take a closer look on features which are relevant to differentiate between most similar classes only (*e.g.*, all known woodpeckers). Note that such a hierarchical decision process intuitively makes sense, since features which are relevant for differentiating woodpeckers might differ significantly from informative features for gulls or warblers.

Nonetheless, the majority of our recognition systems are still build on flat classification models which treat all classes alike. This especially holds for the classification models which we presented in this thesis so far. Although pre-defined hierarchical models introduce flexibility in this aspect, *e.g.*, (Deng et al., 2009, 2012), the underlying hierarchies are usually based on linguistic relations, *e.g.*, WordNet (Fellbaum, 1998), which are not necessarily related to visual consistency. In contrast, a hierarchy which is based on visual features can yield a plausible solution. Early works addressed learning of hierarchies from visual features, *e.g.*, Sivic et al. (2008), but even these schemes might still be too rigid for real-world scenarios. Intuitively, an optimal dividing strategy would adapt to every test example and splits data according to query-specific criteria. Thereby, for every test example, only “related” training examples are considered for model training which is also known as *local learning*.

In the early nineties, local learning was introduced by Bottou and Vapnik (1992). According to the authors, local learning could lead to suitable trade-offs between model capacity (see Section 2.1.2.2) and number of training examples, especially if training examples are non-uniformly distributed in space. Some rare exceptions followed this idea through the years, *e.g.*, (Zhang et al., 2006) for image categorization, or (Ionescu et al., 2013) for recognizing facial expressions. Urtasun and Darrell (2008) introduced a local learning extension of Gaussian process regression to overcome computational limitations during training. Yu and Grauman (2014) showed that local learning is beneficial for relative attribute prediction when attributes are not necessarily transitive and global models are hard or even impossible to learn. Similar in spirit to the local learning principle is the approach of Frome et al. (2006) which learns distance functions specifically tailored for every training example to

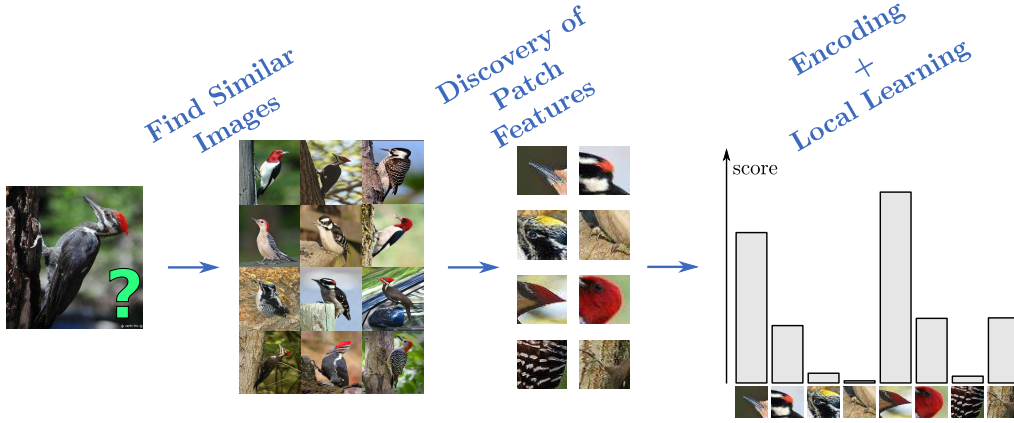


Figure 6.11: Pipeline of our local learning approach which results in exemplar-specific representations and models: Whereas common approaches train a single model with a universal, fixed image representation, we propose to query for a given test image (left) its most similar training images (middle left). Learning of mid-level representations is done on the retrieved set only and thus results in descriptions relevant to differentiate highly similar objects (middle right). Query set and test example are then encoded with the discovered patches (right) and a local model is trained, which finally serves for classifying the query image.

result in small distances to examples of similar classes. An extension for comparability of learned distance functions is presented in (Frome et al., 2007). In our recent paper (Bodesheim et al., 2015), we showed that local learning is also valuable for novelty detection, especially for increasing number of known categories.

6.3.2 On Finding Similar Objects

As mentioned previously, we are interested in finding the $K \ll N$ most similar training examples for a previously unseen test image. The search results can then serve as input set for exemplar-specific patch discovery and model training, which is visualized in Fig. 6.11. We have already seen in Section 6.2 how patch detectors can be learned from given images. Thus, it remains to find objects which are similar to the current object. This task is closely related to the research area of *object retrieval* (Arandjelovic and Zisserman, 2012; Babenko et al., 2014; Chum et al., 2007; Philbin et al., 2008).

In object retrieval, we are usually interested in finding images within a huge dataset showing the exact same object as given in the query image, *e.g.*, all previously taken holiday snapshots which contain the Hertford bridge. For our running scenario of bird recognition, however, our data collection consists of individual birds from different species rather than multiple images of the exact same bird. Thus, we argue that a rough search for objects with similar pose and similar overall

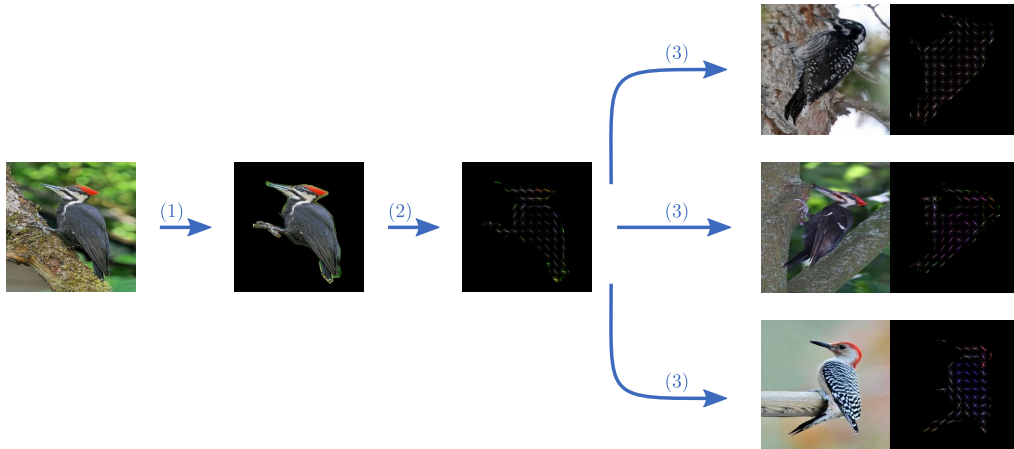


Figure 6.12: *Selecting objects with similar shape and overall appearance to be used for learning exemplar-specific representations: (1) background pixels are masked out, (2) global representations are computed, and (3) matching with the entire set of known images results in objects most similar to the query.*

visual appearance is already sufficient. The final distinction between retrieved classes is then done based on discovered patches and learned classification models.

A visualization of the resulting query technique is given in Fig. 6.12. As for seeding during patch discovery (see Section 6.2.1), we again mask background regions in step (1) which might negatively influence the patch discovery process otherwise. For the sake of simplicity, we propose to represent the masked image with a global cell array consisting of HOG features (capturing pose and texture) and color name features (capturing color) in step (2). For matching representations of all training examples with the query, we can apply any metric of our choice (Jia and Darrell, 2011). Furthermore, hashing techniques such as (Sadeghi and Forsyth, 2014; Yagnik et al., 2011) for fast approximate correspondence search are likely beneficial for huge dataset sizes, too. However, we will see in the experimental evaluation in Section 7.4.3 that already a simple Euclidean distance based search leads to decent results. Thereby, we can reliably find objects in the same pose and with roughly same color and texture patterns. Following established techniques for data augmentation (Krizhevsky et al., 2012; Rowley et al., 1995; Sung and Poggio, 1994), we further add the flipped version of each training image and thereby allow for matches between a query image and known objects facing opposite directions.

As outlined before in Fig. 6.11, all query results are then used as input for the patch discovery technique described in Section 6.2. This results in a set of patch detectors that are specific for the current test image and especially for the global shape and pose of the object in it. With the discovered patch detectors at hand, we can finally compute feature representations for the test image as well as for the

query results, train a model of choice on the encrypted queried images, and use the model to ultimately predict the category label of the test image.

6.3.3 Asymptotic Computation Times for Exemplar-specific Patch Discovery

In Section 6.2.5, we analyzed asymptotic computation time behavior of our proposed patch discovery scheme, which resulted in $\mathcal{O}(N^2)$ operations required. Considering the local learning idea presented previously, we directly notice that no separate training step is required. Instead, all computations are performed for every test image on-the-fly. For finding K similar images, we asymptotically need to compare each training image with the given query, resulting in $\mathcal{O}(N)$ operations. With the set of similar images at hand, we can then perform the patch discovery process in $\mathcal{O}(K^2)$. Thus, we shifted $\mathcal{O}(N^2)$ computations from training to $\mathcal{O}(N + K^2)$ computations in testing.

6.4 Transferring Semantic Part Annotations

In the previous sections, we have been interested in automatically discovering patterns from data to describe objects based on the presence or absence of these patterns. These representations, among others, offer the advantage that no human annotations are required during the discovery process. In some applications, however, a user might be willing to annotate the exact positions of object parts in training images, *e.g.*, locations of beak, wings, or tail of a bird. In this case, the provided information is likely beneficial, especially when we consider that experts usually define species based on distinct characteristics of certain parts. In the following, we show how provided part locations can be transferred from training images to new unseen objects. The technique presented in this section is based on our findings published in (Göring et al., 2014).

6.4.1 Parametric Models for Part Annotation Transfer

From a machine learning point of view, transferring part annotations from training images to test images can be viewed as learning part detectors from annotated data and evaluating them on test data. For the common scenario of predicting a bounding box as approximation of part positions, this results in learning a model $M : \mathcal{I} \mapsto (x, y, w, h)^{n_p}$ which estimates position (x, y) as well as extent (w, h) of n_p object parts in previously unseen images. Note that for $n_p = 1$, the task boils down to pure object detection. In Section 2.2.4, we have introduced the deformable parts

models (DPMs) by Felzenszwalb et al. (2010) which have been the most commonly used detection models within the last decade. Therefore, let us take a closer look on how parts are actually located within that framework.

Can DPMs Express Common Variations in Fine-grained Recognition?: Following the notation of (Felzenszwalb et al., 2010), the deformation model $\mathbf{d}$ of a DPM is defined by the parameters $\mathbf{d} = (\mathbf{d}_1, \dots, \mathbf{d}_{n_p}, \mathbf{v}_1, \dots, \mathbf{v}_{n_p})$ where $\mathbf{d}_i = (x_i, y_i)$ is the anchor position of part i relative to the root filter, and $\mathbf{v}_i = (a_i^x, a_i^y, b_i^x, b_i^y)$ are coefficients of a quadratic function judging the displacement of part location $\mathbf{z}_i$ relative to anchor positions $\mathbf{v}_i$. Then, we can compute the deformation cost as follows⁴:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{z}; \mathbf{d}) = \sum_{i=1}^{n_p} \left(z_i^x, z_i^y, (z_i^x)^2, (z_i^y)^2 \right) \cdot \mathbf{d}_i . \quad (6.9)$$

On a closer look, the cost function $\mathcal{L}_{\mathcal{D}}$ corresponds to a Gaussian model for the part locations, *i.e.*, covariance matrices $\mathbf{S}_i \in \mathbb{R}^2$ and mean vectors $\mu_i \in \mathbb{R}^2$ can be derived, such that the following holds:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{z}; \mathbf{d}) \propto -\log \prod_{i=1}^{n_p} \mathcal{N}((z_i^x, z_i^y) \mid \mu_i, \mathbf{S}_i) . \quad (6.10)$$

The question remains whether this model is complex enough to cope with real-world data. Notably, visual deformations of common objects due to changes in viewpoint can often be captured (see (Felzenszwalb et al., 2010, p. 18) for visual results). However, recall that we are interested in lifelong learning scenarios. In consequence, our model will likely be confronted with heavily articulated objects such as birds over time. Hence, we are interested in examining which deformations actually occur for these object classes, and whether or not DPMs as a powerful representative for parametric models are able to capture them. Therefore, let us analyze part offsets present in the CUB-2011/200 birds dataset by Wah et al. (2011a) by visualizing the relative position of every part with respect to bird's bounding box. Results for four selected parts computed from all training images of all 200 bird species are shown Fig. 6.13.

As we can see, parts often occur in similar relative locations, however, a significant number is also far off the location centroids. Due to this heavy-tailed behavior, the assumption of Gaussian-distributed locations is not perfectly adequate. Thus, a single DPM model would not be able to model the present variation. Note that in

⁴The deformation model given here ignores the fact that in the original DPM formulation, a finer resolution is used for all parts. However, our analysis still holds and we only ignore this fact for the DPM review to simplify the presentation.

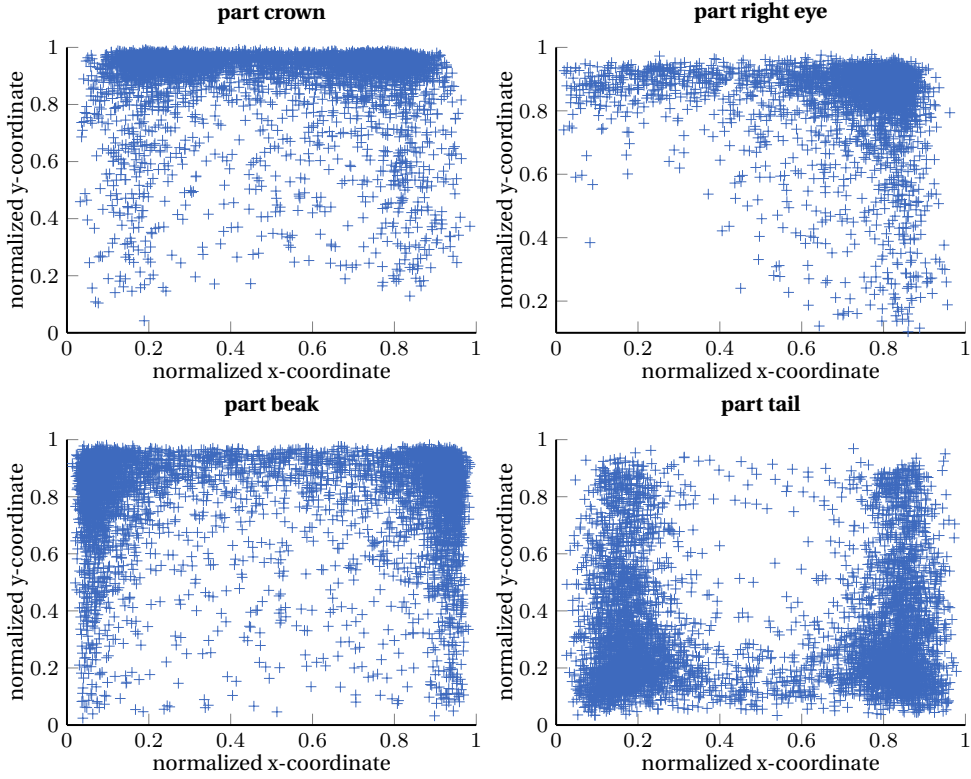


Figure 6.13: Analysis of relative part offsets with respect to the object bounding box computed from all 200 bird species of the CUB-2011/200 dataset (Wah et al., 2011b). Shown are results for beak, tail, forehead, and crown of birds. Since distributions are clearly non-Gaussian, a parametric model with Gaussian deformation costs is unlikely to generalize well.

the original DPM version by Felzenszwalb et al. (2010), parts and configurations are learned in a fully unsupervised fashion. However, the analysis of the ground truth parts provides a certain intuition for the complexity of the problem tackled, and especially holds for the supervised counterpart of the original DPM version by Azizpour and Laptev (2012).

Drawbacks of DPMs: One reason for the non-Gaussian distribution of part locations is the existence of different viewpoints under which objects are recorded. For coping with these issues, mixtures of components $\mathcal{M} = \{M_{DPM}^0, M_{DPM}^1, \dots\}$ can be trained, and z is augmented to add the latent component label the example belongs to. However, as can be seen in Fig. 6.13, the distribution has a large number of different components and the data can not be easily clustered. Thus, learning an appropriate mixture of deformable part models is an ill-posed problem.

Furthermore, a single linear appearance model is not flexible enough to capture the large variability in fine-grained recognition scenarios. This complements the problem that the deformation model is likely too restrictive. In the next section, we will therefore investigate a non-parametric technique for “detecting” object parts in test images. Our proposed technique circumvents the estimation of linear detectors for objects or parts and deformation models by *transferring* annotations from training images to visually similar test images.

6.4.2 Non-parametric Models for Part Annotation Transfer

In the previous section, we have seen that moderately simple models struggle with tackling strong variations of object poses present in images. Our provided analysis naturally motivates the development of models with higher complexity, both for deformation of part positions as well as part detection.

For deformation modeling, using non-Gaussian deformation models has been proposed by Ramanan (2006). To overcome limitations of linear detectors, Vedaldi et al. (2009) proposed to use non-linear models to increase model complexity. However even with speed-ups of explicit feature transformations (Sreekanth et al., 2010; Vedaldi and Zisserman, 2012) or efficient model evaluations (see Chapter 3), these methods are still costly during learning and prediction. Therefore, we follow an orthogonal path and again draw inspiration from one of the most successful companies today:

“Simplicity is the ultimate sophistication.”

(Apple computer inc., 1977)⁵

Following this philosophy, we do not learn an increasing number of models with increasing complexity from given training data. Instead, we make use of a simple but effective technique for part annotation *transfer*.

Objects in Similar Pose Should Have Similar Part Positions: Our approach is based on the assumption that two objects which appear in the same overall pose should exhibit similar part configurations and part locations. We acknowledge this intuition by the following three steps: (1) capturing the current pose, (2) matching with training images, and (3) transferring annotations “as is” to the test image. For capturing the pose of an object, we again apply a pipeline similar to the one in Section 6.2.1 and Section 6.3.2. As explained in the overview for local learning in Fig. 6.12, we again start by masking background pixels using either a pixel-wise segmentation if available or unsupervised techniques such as GrabCut otherwise.

⁵Note that the quotation is also attributed to Leonardo da Vinci. However, to the best of our knowledge, there is no original source proofing this claim.

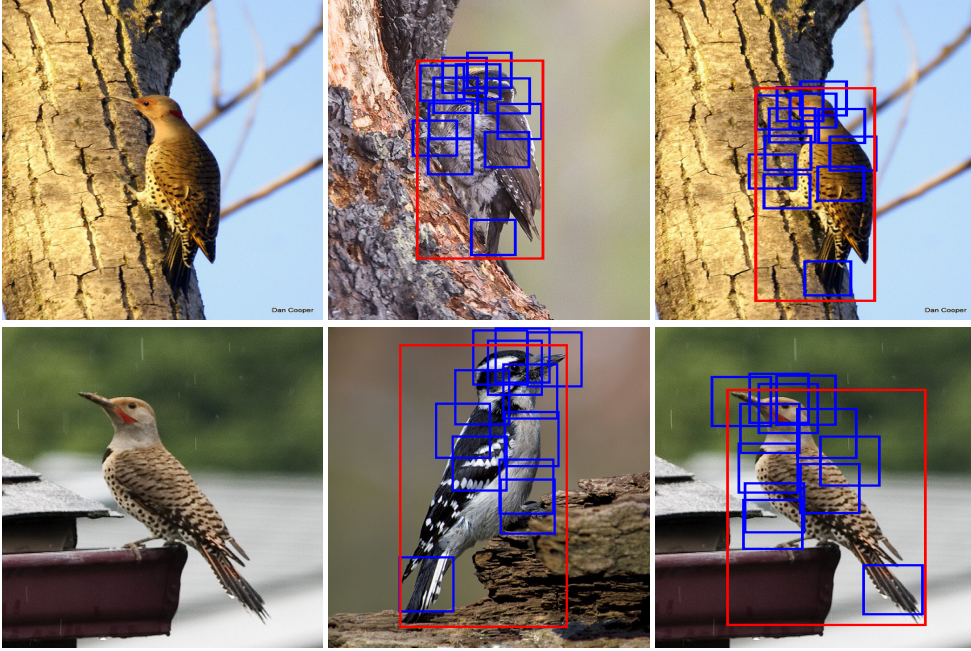


Figure 6.14: *Transferring part annotations from known to unknown objects based on similar pose. Left: unknown test image. Middle: known image most similar with provided part annotations. Right: result after transferring part annotations.*

Information of resulting foreground pixels is then encoded in a global representation. Here, however, we argue that a description which solely relies on HOG features is more than justifiable, since an object's pose is defined by its boundary (corresponding to shape) as well as strong texture (corresponding to part transitions) rather than by its visual appearance in terms of color. Given this representation, we again apply a nearest neighbor search based on Euclidean distances, and thereby retrieve a small set of objects in similar pose. Two examples from the application area of bird recognition on the CUB-2011/200 dataset are given in Fig. 6.14.

As for local learning, we augment training data using the flipped version of each image to again allow for matches between objects of different orientation. For the case of part annotation transfer, however, a careful handling of part correspondences is necessary when a test image matches with a flipped training image, *e.g.*, the flipped left wing position will be the right wing position of the flipped image.

Interestingly, our part detection technique is related to the idea of Exemplar-SVMs by Malisiewicz et al. (2011) as mentioned previously. A main motivation for the development of their technique was to allow for transferring segmentation masks to previously unseen images. Thus, information transfer should only take place if presence *and* position of the learned object are known, which requires expensive learning including hard negative mining to tackle the large number of

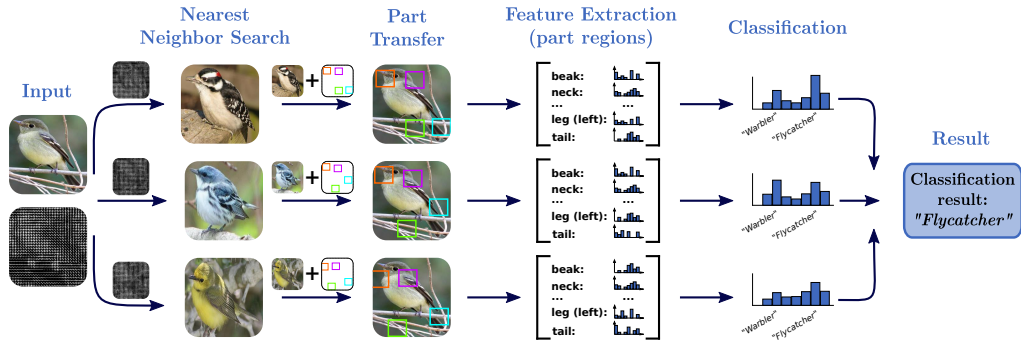


Figure 6.15: Visualization of our approach for transferring part annotations to a test image with unknown content. Training images showing objects in highly similar pose as present in the test image are selected, and annotations are directly transferred “as is”. Feature extraction is then performed on part locations only, and classification results are obtained by averaging individual classification scores for every transferred part configuration.

possible negative examples. However, in our scenario, restrictions are less strong since objects are already in focus, either by human interactions or generic detectors. Thus, the necessity of detection during information transfer is avoided.

Exploiting Transferred Part Annotations: Given the transferred annotations, we can extract features on relevant positions only. Thereby, a mapping of corresponding parts is directly possible. To fine-tune the final encoding with respect to characteristics of every part, codebooks are learned for each part individually. This allows for learning prototypical local elements specific for each part, *e.g.*, characteristic elements of a beak. If not all parts are visible in an image due to the relative position of bird and camera, we use zero imputation (Saar-Tsechansky and Provost, 2007) in absence of further knowledge.

To finally fuse results from transferring annotations of k objects in most similar pose, two complementary options exist: position information for every part can be directly averaged as suggested by Gavves et al. (2013), or every transferred part configuration is independently used for feature extraction and classification, and classification scores are finally combined, *e.g.*, via average pooling.

For a summarizing overview, the whole approach is visualized in Fig. 6.15. Note that due to its simplicity, it perfectly scales with increasing number of known images, since no computationally demanding model updates are required.

7 | Experiments

“Grey [...] is all theory [...]”

(von Goethe, 2015)

In this chapter, we analyze the previously presented techniques in several application scenarios. We start with introducing relevant datasets and evaluation metrics in Section 7.1. Our techniques for large-scale Gaussian process models with histogram intersection kernels are evaluated in Section 7.2. We further investigate active learning for binary and multi-class classification scenarios in Section 7.3. Finally, we evaluate our techniques for feature learning and part annotation transfer in Section 7.4.

We report each individual experiment in a standard layout. The *experimental setup* section specifies all relevant settings regarding data, models, parameters, and baselines. Experimental results are presented and discussed in the *experimental results* section. Finally, a *summary* of one sentence length concludes the findings. In addition, note that the summary of all experimental findings is collected in Appendix C.

7.1 Experimental Setup

Before we present evaluations of our techniques in the subsequent sections, let us introduce the datasets which will be used within the evaluations (Section 7.1.1). In addition, we briefly review evaluation metrics in Section 7.1.2.

7.1.1 Overview of Datasets Used

7.1.1.1 ImageNet

ImageNet is a still growing dataset, which already contained more than three millions of images when Deng et al. (2009) released the first version in 2009. Categories are organized hierarchically following WordNet – a semantic ontology by Fellbaum



Figure 7.1: Exemplary images of eight different species obtained from the ImageNet dataset by Deng et al. (2009).

(1998) – such that each image belongs to a series of semantic concepts of increasing granularity.

Within our experiments, we will use the part of the dataset on which the ImageNet Large Scale Visual Recognition Challenge in 2010 (ILSVRC’10) was run. The resulting dataset contains in total 150,000 images from 1,000 different categories, where each category offers 100 images for training and additionally 50 disjoint images for evaluation from the training and validation set, respectively. In Fig. 7.1, we displayed images of eight different categories which are contained in ImageNet.

Besides the provided images, the challenge organizers also offered a variety of software tools for establishing computer vision baselines, *e.g.*, routines for feature extraction¹. In detail, we will often rely on the provided BoW features² to allow for easy reproducibility and comparison.

ImageNet will be used in a multitude of experiments, including evaluations regarding large-scale learning with GP/HIK in Section 7.2 as well as comparisons of active learning techniques in Section 7.3.

7.1.1.2 Caltech-256

The Caltech-256 dataset introduced by Griffin et al. (2007) can be seen as the predecessor of ImageNet. The dataset was designed to evaluate image classification algorithms and covers 256 different categories, mainly consisting of man-made objects (*e.g.*, bathtub, chess-board, mailbox, or radio telescope) and animals (*e.g.*, elk, grasshopper, leopard, or zebra). As for ImageNet, all images show only one centered object as can be seen in Fig. 7.2.

The number of images per category ranges from 80 up to 827 and sums to 29,780 images in total. No official split in training and test images is provided. Note that the dataset also contains an additional “clutter” category which, however, plays no role in any experiments shown in this chapter. It should be noted that Caltech-256 was

¹http://www.image-net.org/challenges/LSVRC/2010/download/ILSVRC2010_devkit-1.0.tar.gz

²<http://www.image-net.org/download-features>



Figure 7.2: Exemplary images of eight different species obtained from the Caltech-256 dataset by Griffin et al. (2007).



Figure 7.3: Exemplary images of eight different scene categories obtained from the 15Scenes dataset by Lazebnik et al. (2006). Top: four indoor categories (kitchen, living room, office, and store). Bottom: four outdoor categories (open country, coast, street, and highway).

the de-facto standard evaluation dataset during the last decade, *i.e.*, until ImageNet was released.

Similar to ImageNet, we will use Caltech-256 in Section 7.2 to evaluate GP/HIK as well as in active learning experiments in Section 7.3.

7.1.1.3 15Scenes

15Scenes was introduced by Lazebnik et al. (2006) and serves as third dataset for image classification in our experiments. As the dataset's name suggests, it consists of 15 scene categories, covering both indoor and outdoor scenes (*e.g.*, kitchen or bedroom and coast or highway). The number of provided images range from 200 to 400 for each category and sums to 4,485 images in total. Note that all images are grayscale. Images of eight categories are shown in Fig. 7.3. As suggested by Quattoni and Torralba (2009), all images are scaled to have fixed size of $256\text{px} \times 256\text{px}$ to overcome dataset bias induced by class-specific image sizes and aspect ratios.

The 15Scenes dataset will serve as small-scale dataset for detailed analyses of learning, classification, and optimization of hyper parameters in Section 7.2.

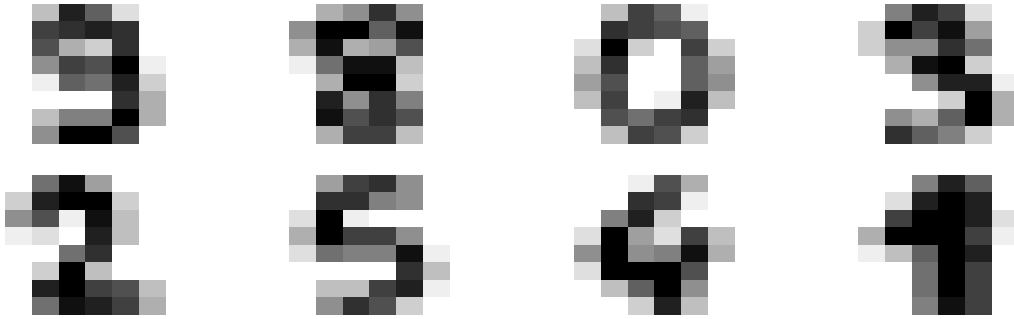


Figure 7.4: Exemplary images of eight handwritten digits from the USPS dataset by Bache and Lichman (2013) after preprocessing to $16\text{px} \times 16\text{px}$ resolution. From top left to bottom right, ground truth categories are 9, 8, 0, 3, 2, 5, 4, 1.

7.1.1.4 USPS Handwritten Digits

The USPS handwritten digits dataset by Bache and Lichman (2013) is a common benchmark dataset in the machine learning community. It contains 5,629 binary images showing handwritten digits of resolution $32\text{px} \times \text{px}$. Images were recorded from 43 people. The dataset provides an official split into 3,823 examples for training and 1,797 examples for testing. Notably, images of an individual writer are either provided for training or for testing, but not both. Thereby, the training set consists of images from 30 persons, whereas the test set covers 13 different writers. Features are commonly extracted by cropping $4\text{px} \times 4\text{px}$ non-overlapping blocks and quantizing gray values to the interval $[0, 16]$. Finally, resulting values of all blocks are concatenated into a single vector. Images of eight categories after preprocessing are shown in Fig. 7.4. We will use the USPS dataset to analyze multi-class active learning and discovery techniques in the presence of unnameable instances in Section 7.3.2.

7.1.1.5 Labeled Faces in the Wild (LFW)

The “Labeled faces in the Wild” dataset was released by Huang et al. (2007). It contains faces images of 5,749 individual persons which are publicly known, *e.g.*, from politics, sports, or movies. There is no official split available. While the majority of categories contains only few examples, 9 categories provide at least 55 individuals views. Thus, the distribution of examples per category is heavily scattered. We will use this dataset to analyze multi-class active learning and discovery techniques in the presence of unnameable instances in Section 7.3.2.



Figure 7.5: Exemplary images of eight different species obtained from the Caltech-UCSD Birds-200-2011 dataset by Wah et al. (2011b).

7.1.1.6 Common Objects in Context (COCO)

Microsoft COCO by Lin et al. (2014) is a recently released dataset with carefully annotated data in several levels of detail. The dataset was released in two parts. The training set consists of 164,000 images which are distributed over 91 categories. It is complemented by a validation set with 82,000 of the same categories. Instances of individual objects from the 91 categories are annotated pixelwise.

As for the datasets USPS and LFW, we will use MS-COCO to analyze multi-class active learning and discovery techniques in the presence of unnameable instances in Section 7.3.2.

7.1.1.7 Caltech-UCSD Birds-200-2011

As dataset for fine-grained recognition, the Caltech-UCSD Birds-200-2011 dataset by Wah et al. (2011b) is currently the de-facto standard dataset within the computer vision community. The dataset covers 200 bird species which are native in North America. Eight bird species which are present in the full dataset are shown in Fig. 7.5.

In total, the dataset provides 11,788 labeled images which are additionally annotated with 15 keypoints, *e.g.*, position of a bird's beak, neck, crown, or left and right wing. Exemplary landmark annotations are shown in Fig. 7.6. Furthermore, the dataset offers pixelwise foreground masks collected via Amazon Mechanical Turk. For evaluation, the dataset provides an official split into images for train and testing such that both sets are of approximately same size. In addition, Farrell et al. (2011) introduced a commonly used subset of the entire dataset which consists of 14 categories of warblers and woodpeckers and contains 817 images. It should be further stressed that the provided category labels do not distinguish between male and female, nor between young and adult. We refer to the entire set and its 14-class subset as CUB-2011/200 and CUB-2011/14, respectively. The CUB-2011 will be used to analyze fine-grained recognition techniques in Section 7.4.



Figure 7.6: Two individuals of the species “Prairie Warbler” contained in the Caltech-UCSD Birds-200-2011 dataset Wah et al. (2011b) with annotated keypoints. Parts of head regions are shown in *bright blue*, body parts are given in *dark blue*, and wings and legs are marked in *red*.

7.1.1.8 Datasets for Semantic Segmentation

In Section 7.2.5, we will apply Gaussian process models to semantic segmentation scenarios where each pixel is assigned to the category it probably belongs to. Consequently, we also need adequate data to train and evaluate models that compute classification decisions for pixels rather than for entire images. Therefore, we will use two datasets which provide labels for each pixel of the contained images. The first dataset is eTRIMS, which has been introduced by Korc and Förstner (2009). The dataset comes with 60 images which are labeled according to eight categories (*building*, *car*, *door*, *pavement*, *road*, *sky*, *vegetation*, and *window*) as well as one category for pixels which do not belong to any of the previous categories. An image contained in eTRIMS along with its provided ground truth segmentation is shown in the left part of Fig. 7.7.

Since eTRIMS by itself is quite small, we use LabelMeFacade introduced by Fröhlich et al. (2010) as second dataset in evaluations which can be obtained from <http://www.inf-cv.uni-jena.de/labelmefacade>. The dataset is a subset of the larger LabelMe dataset by Russell et al. (2008) and consists of 945 images in total. Ground truth segmentations cover the same categories as eTRIMS. An image of LabelMeFacade along with its ground truth is shown in the right part of Fig. 7.7.

7.1.2 Evaluation Metrics

Before we move on to the experimental evaluations, let us shortly introduce different measures of accuracy to estimate the performance of different approaches. To this end, we assume a model $\hat{h}$ trained on some labeled data $\mathcal{L}$. Since we want to estimate its real-world performance as accurate as possible, we additionally collect a representative, labeled test dataset $\mathcal{T} = \{(x_i, y_i)\}_1^{N_{\mathcal{T}}}$.

Similar to the empirical risk framework as introduced in Section 2.1.2, we can

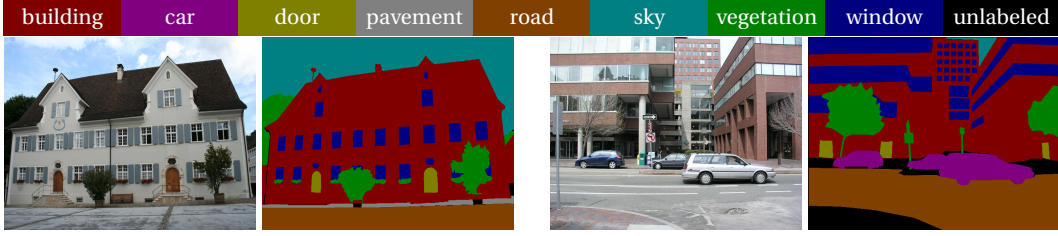


Figure 7.7: Two datasets for semantic segmentation: *eTRIMS* (left) by Korc and Förstner (2009) and *LabelMeFacade* (right) by Fröhlich et al. (2010). For each dataset, we show a provided image along with its ground truth annotation obtained from human labeling. Class labels are color coded as shown on the top.

now evaluate $\hat{h}$ on all labeled test examples. After applying a suitable loss function to compare outputs with ground truth data, we can fuse individual results to compute a final accuracy estimate. Depending on the provided data and the tackled scenario, we have several choices for possible measures to apply. In the following, we review some of them which will be of major importance for the remaining evaluations.

7.1.2.1 Evaluation Metrics for Multi-class Classification

For multi-class scenarios, ground truth labels are discrete and given as one of M possible values, *i.e.*, $y_i \in \mathcal{Y} = \{1 \dots M\}$. It is therefore common to evaluate the trained model $\hat{h}$ by applying the 0/1-loss to compare estimated with desired class labels. For fusing all decisions into a final accuracy estimate, the simplest option is to average all results which is known as *overall recognition rate (ORR)*:

$$\text{ORR}(\hat{h}, \mathfrak{T}) = \frac{1}{N} \sum_{i=1}^{N_{\mathfrak{T}}} \mathcal{L}_{0/1}(\hat{h}(x_i), y_i) \quad . \quad (7.1)$$

In direct consequence of the simple averaging, the ORR can be affected by strongly imbalanced class sizes, *i.e.*, if a single category is significantly over-represented in $\mathfrak{T}$, ORR will mainly indicate the model's performance for this category. An alternative evaluation metric which is invariant to skewed class distributions is the *average recognition rate (ARR)*. Here, class accuracies are normalized by their relative numbers of examples and then averaged:

$$\text{ARR}(\hat{h}, \mathfrak{T}) = \frac{1}{M} \sum_{m=1}^M \text{ORR}(\hat{h}, \{(x_i, y_i) \in \mathfrak{T} : y_i = m\}) \quad . \quad (7.2)$$

Both metrics can also be motivated from the perspective of class confusion. A common visualization tool which follows this idea is the so-called confusion matrix

$M \in \mathbb{N}_*^{M \times M}$ which contains in entry $M(k, l)$ the number of examples $x_i \in \mathcal{T}$ with ground truth label $y_i = k$ and predicted label $\hat{h}(x_i) = l$. Note that we assumed the set of available categories to be the same during training and testing. Nonetheless, a generalization to non-equal class sets would be directly possible. From a given confusion matrix, the ORR can be computed as mean of all diagonal entries divided by the total amount of entries in M :

$$\text{ORR} = \frac{\text{tr}(M)}{|M|_F} . \quad (7.3)$$

Similarly, the ARR is the mean of all main diagonal entries after L_1 -normalizing each row to length one:

$$\text{ARR} = \frac{1}{M} \sum_{k=1}^M \frac{M(k, k)}{\sum_{l=1}^M M(k, l)} . \quad (7.4)$$

An important generalization is often used when datasets exhibit an extremely large number of categories. For these datasets, it might be extremely difficult to predict the correct category if only a single prediction for each example is allowed. A less stringent version is to allow the system the prediction of k most likely categories, and count success if the correct category is among these top- k . Since classification errors among categories are still treated equally, this measure is known as flat- k error rate. It is commonly applied to assess classification systems on ImageNet using the *flat-5 error* (see (Russakovsky et al., 2015)). Note that the previously introduced ORR can be seen as flat-1 error.

It should be noted that all measures can be easily generalized by applying different loss functions, *e.g.*, to account for different costs occurring from different classification errors (see Section 2.1.2.1 for a short discussion). As an example, note that ImageNet is organized in a tree-format according to WordNet. Distances in this tree between ground truth category and model prediction can then be used as hierarchical cost. For the ImageNet challenges, the top-5 hierarchical costs are commonly reported in addition to the flat-5 error.

7.1.2.2 Evaluation Metrics for Binary Classification

Since binary classification scenarios with $\mathcal{Y} = \{-1, 1\}$ can be viewed as special case of multi-class scenarios, the previously introduced metrics of ORR and ARR can be directly applied, too. To reflect the specific nature of both categories, it is common to refer to correctly classified examples as true positives and true negatives. Similarly, confused examples are denoted as false positives or false negatives, depending on their ground truth category.

Nonetheless, we have seen in the previous chapters that decisions in binary

scenarios are often obtained by thresholding an intermediate, continuous function f . Depending on the chosen threshold, discrete classification results can be significantly affected, and it is by far not trivial to decide upon a “correct” threshold. Even worse, a suitable threshold can vary between application scenarios, *e.g.*, in automotive applications, we could be interested in designing a person detector with a false negative rate as small as possible to not miss a single pedestrian. Alternatively, a different setup might require low false positive rates to prevent the car from breaking unnecessarily. To reflect this ambiguity in choosing a specific threshold (which is also referred to as working point), we can evaluate a system with respect to all possible thresholds. For each working point, we can then plot false positive rate against true positive rate, which will go from (0, 0) when classifying every example as negative to (1, 1) when classifying every example as positive. The obtained graph is known as *receiver operating characteristic (ROC)* and we refer to (Fawcett, 2004) for an insightful introduction. Note further that the similarly popular Recall-precision curve is a closely related measure, and we refer to (Davis and Goadrich, 2006) for a comparison of both metrics. We can finally compress the curve into a single scalar by integrating over all possible thresholds which leads to the *area under an ROC curve (AUC)*. Note that for balanced datasets, random guessing results in approximately 0.5 AUC score, whereas larger scores indicate superior classifiers.

7.1.2.3 Statistical Measures

Given the previously introduced techniques to assess a system’s accuracy, we can collect labeled data, train our model on a (randomly) selected subset, and evaluate it on all remaining data. As mentioned in Chapter 2, this procedure is usually repeated on multiple (random) splits of the dataset to increase statistical reliability of the resulting accuracy estimates, which is also known as cross-validation. For k different splits, we thus obtain k accuracy estimates $s^1, \dots, s^k$. It is common practice to average all estimates and report the resulting score as final accuracy estimate of the evaluated model:

$$s^* (s^1, \dots, s^k) = \frac{1}{k} \sum_{i=1}^k s^i . \quad (7.5)$$

For comparing two different models f_1 and f_2 , this evaluation concept can be directly generalized: on each split, both models are evaluated, which results in two sets of comparable accuracy estimates $s_1^1, \dots, s_1^k$ and $s_2^1, \dots, s_2^k$. A final decision can be obtained by comparing averaged estimates s_1^* against s_2^* . In consequence, this approach can lead to valuable statements of the type “model X works better than model Y for this type of data and under these specific assumptions and conditions”.

While these statements are important to quantitatively compare different approaches, the variation of accuracy estimates is not taken into account when comparing averaged estimates only. A more sophisticated evaluation therefore compares the sequences of estimates and asks whether or not the measured differences are *statistically significant*. In the following, we shortly introduce two techniques which will be frequently used in the following experiments. For further details, we refer to (Bronstein and Semendjajew, 1979, Sect. 5.2.3).

The Paired-sample t-test: Let two series of accuracy estimates be given, $s_1 = s_1^1, \dots, s_1^k$ and $s_2 = s_2^1, \dots, s_2^k$, which have been computed on identical splits but with two different models f_1 and f_2 . If we assume that f_2 was consistently superior to f_1 , we should observe this trend in the majority of comparable estimates, *i.e.*, $s_2^i > s_1^i$ for most of the $i \in \{1, \dots, k\}$. On the contrary, if both models are expected to perform equally well, we can also expect that the majority of accuracy estimates is similar. Thus, their element-wise difference are likely close to zero. The *paired-sample t-test* follows this assumption by computing the statistical probability that the sequence $s_1 - s_2$ was obtained by sampling from a normal distribution with zero mean and unknown variance. Note that the test's name stems from the Student's t-distribution which can be seen as heavy-tailed counterpart of a Gaussian distribution. The Student's t-distribution is used to assess how likely a mean value computed from finite, sampled data coincides with the true unknown mean of the Gaussian distribution the samples were drawn from. As an example, the call

$$[b_hyp, p_conf] = ttest(s_1, s_2);$$

to Matlab's implementation returns the probability estimate p_conf stating how likely the difference of both sequences was sampled from a zero-mean Gaussian.

If both sequences were obtained independently, *e.g.*, by evaluating both models on different random splits, we no longer have matching pairs of accuracy estimates. For these cases, the *two-sample t-test* can be applied which relaxes the previous assumption and instead asks whether both *sequences* have been sampled from Gaussian distributions with identical means.

The Wilcoxon signed rank test: The paired-sample t-test assumed that $s_1 - s_2$ are samples from a Gaussian distribution and estimated the probability that its mean is equal to zero. However, the assumption of a Gaussian distribution is often likely to fail, especially if large variances in accuracy estimates occur. In these scenarios, the *Wilcoxon signed rank test* can be applied, which relaxes the previous assumption and instead computes the probability that $s_1 - s_2$ was sampled from *any* symmetric distribution with zero median. Again, a possible call to Matlab's implementation would be:

$$p_conf = signrank(s_1, s_2);$$

7.2 Experiments on Large-scale Learning with Gaussian Process Models

The first major part of our evaluations will cover our techniques for efficient GP inference using histogram intersection kernels as presented in Chapter 3. After a short comparison of different GP classification techniques (Section 7.2.1), we evaluate various aspects GP/HIK for training and classification in Section 7.2.2. Furthermore, we present experiments regarding the optimization of hyperparameters (Section 7.2.3), efficient variance prediction (Section 7.2.4), and the application of GP/HIK for semantic segmentation (Section 7.2.5).

7.2.1 Prelude – GP Label Regression versus GP Classification

Since the remaining experiments will be concerned with various evaluations of GP models in classification settings, we start with a short comparison of possible classification strategies derived from the GP framework.

For theoretically sound classification models, we introduced transfer functions in Section 2.1.5.8 which transform continuous scores of latent functions to discrete classification decisions. Unfortunately, final classification estimates can no longer be solved in closed-form and approximation techniques are required, *e.g.*, Laplace approximation by Williams and Barber (1998) or Expectation-Propagation by Minka (2001). We therefore introduced the technique of label regression as a computationally tractable alternative, which was proposed by Nickisch and Rasmussen (2008) and is based on regularized least-square classification as proposed by Rifkin and Klautau (2004). Let us now experimentally compare all three techniques.

7.2.1.1 Binary Classification Scenarios

We start with a focus on binary classification tasks.

Experimental Setup: As application scenario, we conduct experiments on the ImageNet dataset (see Section 7.1.1.1). We use publicly available image representations provided by ImageNet³. Thereby, images are represented 1,000 dimensional BoW histograms over densely sampled SIFT features. All histograms are L_1 -normalized. In consequence, the application of a histogram intersection kernel as defined in Eq. (2.109) as covariance function is well justifiable. In absence of further knowledge, we follow the zero mean assumption as motivated in Section 2.1.5. For training of classification models, we use the publicly available `gpm1` toolbox by Rasmussen and Nickisch (2010).

³<http://www.image-net.org/download-features>

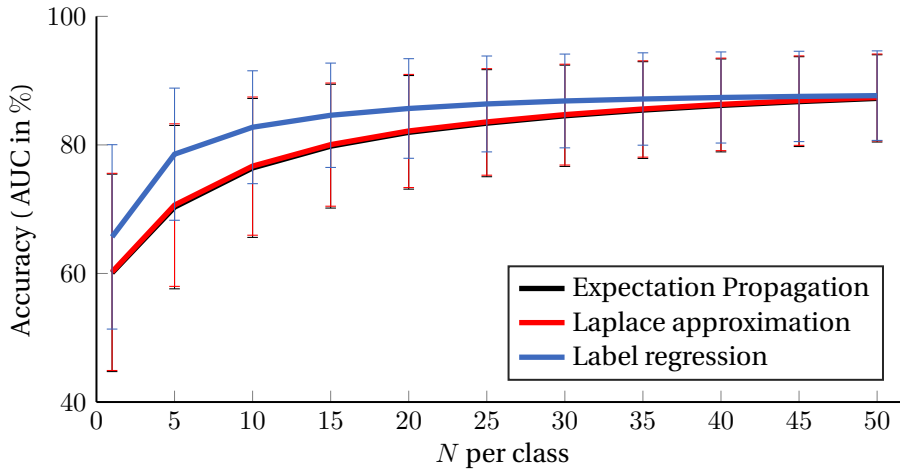


Figure 7.8: Comparing GP label regression (Nickisch and Rasmussen, 2008) with two prominent techniques for approximate inference in binary classification scenarios: Laplace approximation by Williams and Barber (1998) and Expectation-Propagation by Minka (2001). Results are averaged over 1,000 random binary tasks obtained from ImageNet with 10 random initializations per task. See text in Section 7.2.1 for details.

We randomly sample 1,000 binary tasks from ImageNet such that each class serves once as positive category and is to be separated from 9 of the remaining 999 classes selected at random. Thereby, we simulate a one-vs-all classification scenario. Model training for all three methods is done with different numbers of training examples per category. In detail, we train models with 1 and up to 50 examples for each category which have been selected at random from the official training set. To further increase statistical significance, random selection of training examples is repeated 10 times for each set of selected classes. For model evaluations, we use the official validation set and classified all examples corresponding to the selected classes of each binary task. Final results are obtained by averaging area under an ROC curve (AUC) scores and are shown in Fig. 7.8.

Experimental Results: Interestingly, the approximate inference methods perform poorly for extremely small number of training examples (the chance level of random guessing is 50% AUC score). Even for increasing number of examples, both techniques are significantly inferior to the simple label regression approach. For further increased dataset sizes, all techniques finally achieve comparable results. From a statistical point of view, label regression is still significantly superior to Laplace and EP ($p < 10^{-25}$ according to a Paired-sample t-test, see Section 7.1.2.3).

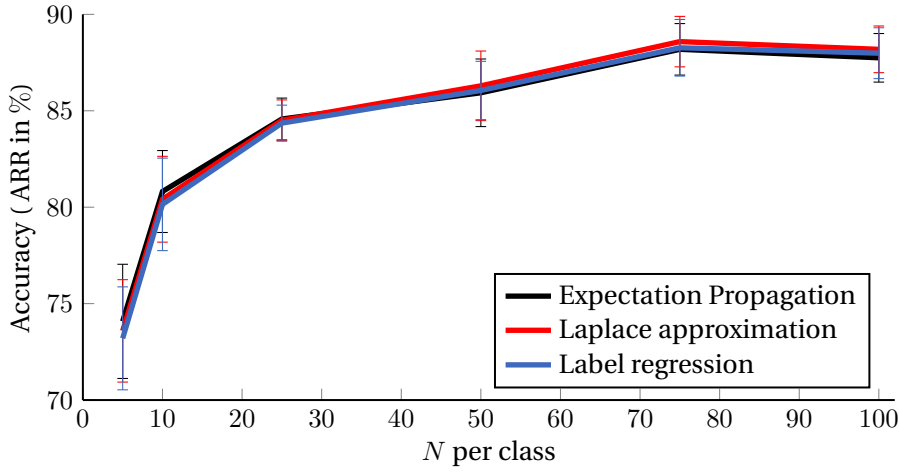


Figure 7.9: Comparing GP label regression (Nickisch and Rasmussen, 2008) with two prominent techniques for approximate inference in multi-class classification scenarios: Laplace approximation by Williams and Barber (1998) and Expectation-Propagation by Minka (2001). Results are obtained on the 15Scenes dataset and are averaged over 10 random initializations. See text in Section 7.2.1 for details.

Summary: In summary, GP label regression is on par or even superior to its theoretically sound competitors for binary classification tasks.

7.2.1.2 Multi-class Classification Scenarios

In addition to the previous evaluation, we also compare all three techniques in a multi-class classification scenario.

Experimental Setup: We chose the 15Scenes dataset due to its moderate size. Images are represented using CNN activations as shortly reviewed in Section 2.2.3. In detail, we use AlexNet and extract activations of layer `relu7` which are non-negative as required by κ_{HIK} (a more detailed analysis of CNN features is given in Section 7.2.2.3). During training, we use all 15 categories of the dataset and vary the number of selected training examples per category. We follow the one-vs-all principle as introduced in Section 2.1.5.8 and train a single model for each category. Again, we use the publicly available `gpm1` toolbox. For model evaluation, we additionally select 50 hold-out test examples for each category at random. Classification accuracy of trained models is measured as average recognition rate (ARR). As in the previous experiment, we average results over ten random splits. Results are shown in Fig. 7.9.

Experimental Results: In contrast to the previous experiment, we observe that all three techniques perform comparably for all training set sizes. According to a paired-sample t-test, differences in results of all three techniques are not statistically significant ($p > 10^{-2}$).

Summary: *In summary, one-vs-all GP label regression is well applicable as multi-class GP classification technique due to three reasons: it is easy to implement, it is fast to compute, and classification results are comparable to those of competing techniques.*

7.2.2 Experimental Analysis on Large-scale Classification

In the previous section, we empirically validated that GP label regression models are well suited for classification scenarios. Let us now investigate the benefits of our efficient GP techniques as introduced in Chapter 3. Therefore, we conducted experiments for the following questions:

- (i) Section 7.2.2.1: Can we train binary GP models on ImageNet using GP/HIK?
- (ii) Section 7.2.2.2: Can we train multi-class GP models on ImageNet using GP/HIK?
- (iii) Section 7.2.2.3: Can we successfully apply our GP/HIK on top of CNN activations as recent image representations?
- (iv) Section 7.2.2.4: Is the choice of conjugate gradients as iterative linear solver sensible?
- (v) Section 7.2.2.5: Is it advisable to interrupt the iterative linear solver before convergence?
- (vi) Section 7.2.2.6: How does the GP/HIK compare against sparse GP approximations?
- (vii) Section 7.2.2.7: When and how should we apply the quantization idea?

The source code used for the evaluations is publicly available at <https://github.com/cvjena/gp-hik-core.git>.

7.2.2.1 Large-scale Experiments with the ImageNet Dataset (1) - Binary Classification

In a first experiment, we are interested in applying GP/HIK to scenarios where training and test sets scale to sample sizes which are not longer manageable with naive

Table 7.1: *Evaluation on 200 binary classification tasks derived from the ImageNet database. Computation times are given as mean values of measurements for each task (learning) and each test example (classification).*

Method	10,090 examples ($\ell = 10$)			50,050 examples ($\ell = 50$)		
	AUC	learning time	classif. time	AUC	learning time	classif. time
GP with HIK (Cholesky)	0.836	>3.5 h	1.1 s	—*	—*	—*
GP with HIK	0.836	64 s	44 μ s	0.856	321 s	44 μ s

implementations. Therefore, we investigate two scenarios on the ImageNet dataset: binary classification (this section) and multi-class classification (Section 7.2.2.2). ImageNet is perfectly suited due to its sheer size of collected examples. We start with binary classification scenarios. Although these scenarios might be less important for life-long learning applications, they still come with two important advantages. First of all, they can be seen as an evaluation of single one-vs-all models. Besides, we can obtain at least some results for the GP baseline in affordable time.

Experimental Setup: We compare our efficient GP/HIK techniques against GP models in their plain formulation (see Eq. (2.99) in Section 2.1.5), *i.e.*, with explicit computation of the covariance matrix as well as all kernel vectors. Computing the weight vector $\alpha = K^{-1}y$ is done by relying on the numerically stable Cholesky decomposition as introduced in Section 4.2.2. *i.e.*, decomposing K and computing α with one forward- and backward-substitution. Details on training GP/HIK models are given in Section 3.2.3. For scalability, we apply the quantization trick as introduced in Section 3.2.2.2 with an equidistant quantization into $[0, 1]$ and $q = 100$ bins. The noise level for model regularization is set to $\sigma_n^2 = 1$.

As image representations, we use the precomputed and publicly available BoW features (see Section 2.2.2) with 1,000 dimensions. All computation times are measured on a single-core Intel 2.6 GHz machine to allow for fair comparisons.

Similar to Section 7.2.1, we derive 200 binary classification tasks by randomly selecting one category as positive and nine categories as negative classes for each task. From the ImageNet training set, we use all 100 examples of the positive class as well as $\ell = 10$ or $\ell = 50$ randomly selected examples of each negative class for model training. This results in 10,090 and 50,050 training examples in total for each task, respectively. Learned models are evaluated on the hold-out ImageNet validation set by computing AUC scores (see Section 7.1.2.2) on all examples of the selected categories. Final results are obtained by averaging all 200 results and are shown in Table 7.1.

Experimental Results: We can make the important observation that we can train GP classifiers within a few minutes on large-scale data. In contrast to that, scenarios with tens of thousands of training examples are beyond the limits of baseline GP models – due to computation time and memory demand. In fact, the baseline implementation with explicit computations of the kernel matrix $\mathbf{K}$ requested more memory than our work station offered ($\ell = 50$ results in a 19 GB kernel matrix).

Summary: *In summary, GP/HIK is significantly faster than the baseline GP for training and classification in practical applications. Since the required memory is additionally reduced, it allows for learning on large-scale datasets which are otherwise intractable for exact GP inference.*

7.2.2.2 Large-scale Experiments with the ImageNet Dataset (2) - Multi-class Classification

In a second experiment, we evaluate the applicability of GP/HIK to multi-class scenarios.

Experimental Setup: Since the GP baseline does not lead to results in affordable time, we compare against established SVM solvers which are publicly available. In detail, we use the LibSVM package by Chang and Lin (2011) as representative for non-linear SVMs. Note that the LibSVM does not support κ_{HIK} as kernel function. For an exact comparison, we would thus need to explicitly precompute kernel values and hand them over to LibSVM. However, this option is not possible due to memory limitations as in the previous experiments. Therefore, we follow provided suggestions and apply an RBF kernel as introduced in Section 2.1.4.3, Eq. (2.76). Furthermore, we use LibLinear by Fan et al. (2008) as commonly done for large-scale image classification (Deng et al., 2010). For both solvers, we use recommended default parameter settings.

For classification with GP/HIK, we apply an equidistant quantization into $[0, 1]$ and $q = 100$ bins. Again, we set a constant noise level of $\sigma_n^2 = 1$ for model regularization.

As in the previous experiment, images are represented using provided BoW features with $D = 1,000$. For each category, we randomly pick training examples from the ImageNet training set. To evaluate learned models, we additionally select 50 examples for each category from the ImageNet validation set. This process is repeated for ten random initializations as well as for different numbers of training examples N . Classification accuracy is measured in terms of Flat-5 error rates and Top-5 hierarchical costs as commonly done for ImageNet (see Section 7.1.2.1). Results are shown in Fig. 7.10.

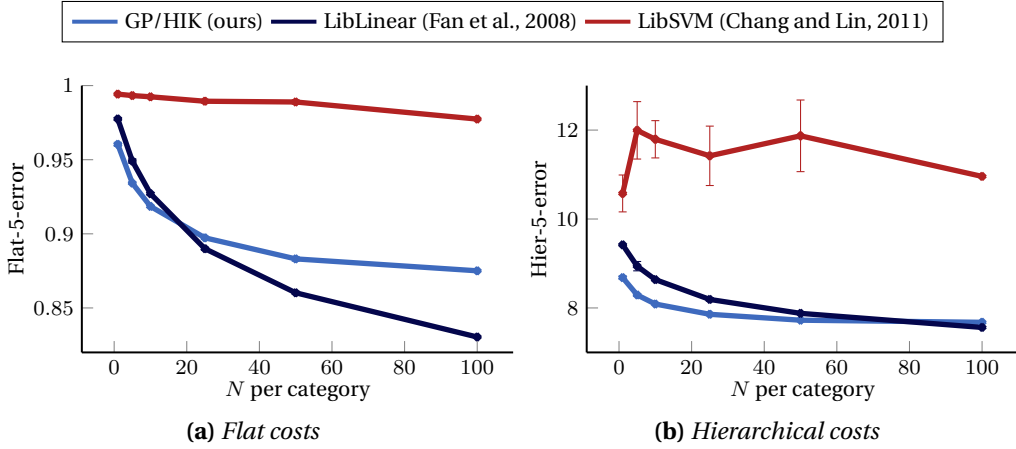


Figure 7.10: Multi-class classification using all 1,000 categories of ImageNet and provided BoW features. Left: Flat-5 error. Right: Hierarchical costs for top-5 predictions. See Section 7.2.2.2 for details.

Experimental Results: As we can see, GP/HIK is indeed able to handle multi-class classification scenarios with 1,000 categories. Interestingly, GP/HIK outperforms the linear SVM for small number of training examples, but is on par or even behind for increasing training sizes. We attribute this aspect to the fixed regularization parameter used in the experiment. Furthermore, we observe that the kernelized SVM results in extremely low accuracy values, which might also be attributed to the choice of recommended default values.

Summary: In summary, GP/HIK provides a useful alternative to established SVM solvers in multi-class classification scenarios with large numbers of training examples and categories.

7.2.2.3 GP/HIK versus SVM on CNN Activations Using 15Scenes

The results of previous experiments on ImageNet proved that we can perform exact GP inference on large-scale datasets. However, the BoW representations provided with the dataset are far from today’s state-of-the-art in image categorization. In contrast, recent successful classification systems rely on CNN activations as shortly reviewed in Section 2.2.3. Let us therefore investigate whether GP/HIK can also be applied as model on top of these representations.

Experimental Setup: Experiments in this section are conducted on the 15Scenes dataset. Thereby, even the largest image representation fit into our available memory ($D = 64,896$ for relu3 features). We extract CNN activations from the AlexNet

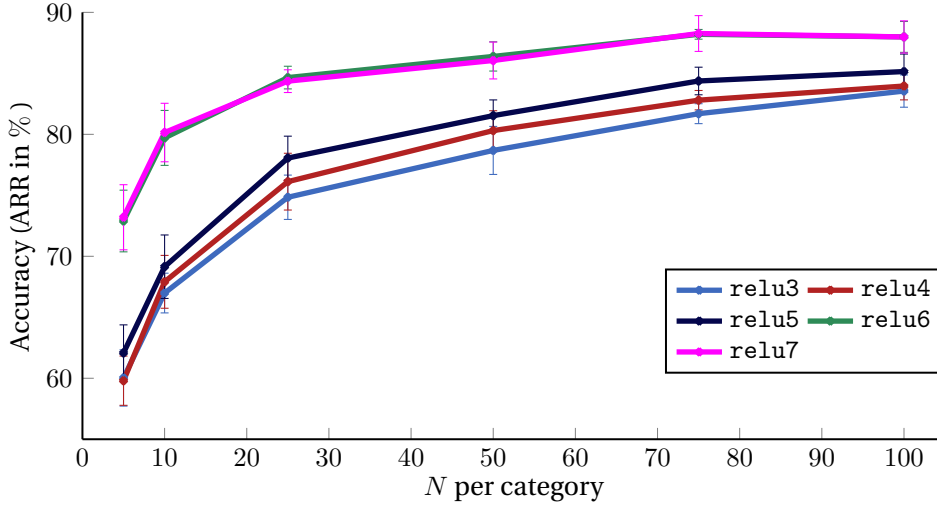


Figure 7.11: Performance of GP/HIK on 15Scenes with CNN features extracted from different relu layers and a varying number of training examples per class.

CNN trained on ImageNet as presented by Donahue et al. (2014) using the Caffe toolbox by Jia et al. (2014). The noise level σ_n^2 for model regularization is set to 0.1. All 15 categories of the dataset are used to form a small-scale multi-class scenario. As in previous experiments, we vary the number of training examples per category. In addition, we randomly select 50 hold-out test examples for each category. Classification accuracy is measured as average recognition rate. Results are finally averaged over ten repetitions.

We start by investigating GP/HIK trained on top of different CNN layers. To guarantee non-negative feature components as required by κ_{HIK} , we use layers relu3 to relu7. Results are shown in Fig. 7.11.

We additionally follow the previous experiments and compare GP/HIK against linear and non-linear SVM. Parameter settings and kernel functions are kept identical to allow for comparability. Given the high accuracy obtained with relu7, we use activations of this layers as image representations. To be consistent with previous evaluations on ImageNet, we also provide results when using BoW histograms instead. Therefore, we use BoW histograms, GIST features, and a spatial pyramid matching (SPMK) extension of BoW as representation. Furthermore, we additionally compare against discriminative part detectors by Sun and Ponce (2013). All results are shown in Table 7.2.

Experimental Results: As can be seen in Fig. 7.11, GP models with κ_{HIK} as covariance function can be successfully trained on top of CNN activations and lead to impressive results. In addition, we observe that high-level layers relu6 and relu7

Table 7.2: *Evaluation of GP/HIK on the 15Scenes dataset with standard BoW features (upper part) and CNN features (lower part).*

Method	Features	ARR
Linear SVM (Fan et al., 2008)	BoW	68.4%
SVM/HIK (Quattoni and Torralba, 2009)	BoW	64.1%
SVM/HIK (Quattoni and Torralba, 2009)	GIST	73.0%
SVM/HIK (Quattoni and Torralba, 2009)	SPMK	73.4%
GP/HIK (Ours)	BoW	70.8%
Linear SVM (Sun and Ponce, 2013)	learned patches	86.00%
Linear SVM (Fan et al., 2008)	AlexNet relu7	86.09%
GP/HIK (Ours)	AlexNet relu7	86.42%

lead to highest accuracy. For the identical setup, we showed in (Freytag et al., 2014d) that humans only obtain an accuracy level of 85.67% on this dataset. Thus, the accuracy of 87.98% with relu7 and the largest train size is indeed remarkable.

Interestingly, GP/HIK again results in higher classification accuracy than SVM baselines when the same features are used. Furthermore, we again observe a significant increase in classification performance resulting from replacing hand-crafted features with learned image representations. Clearly, this observation is consistent with a multitude of recent papers and experiments.

Let us finally note that we can also interpret the process of learning a GP on top of the relu7 layer as changing the network’s architecture by adding an additional, final layer which is fully connected. Training of the GP then equals fine-tuning of the network’s parameters on the new dataset but with fixed parameter values for all but the newly added layer. Although fine-tuning then happens with a different technique than the common gradient descent, *i.e.*, with the parameter estimation as described in Section 2.1.5.6 (the Weight-space view), this interpretation sheds new light on the similarity of model training from CNN activations and fine-tuning of a deep network – an observation we believe is worth being pointed out.

Summary: *In summary, we find that GP/HIK is well suited as classification model on top of CNN activations.*

7.2.2.4 Evaluation of Linear Solvers With Fast HIK Multiplications

Let us now investigate how strongly the choice of a specific iterative linear solver routine affects the learning time of GP/HIK.

Experimental Setup: As introduced in Section 3.2.3, our technique of choice will be the conjugate gradients method by Hestenes and Stiefel (1952) which we will compare against two coordinate descent technique. Following the previous setup, we apply an equidistant quantization of features in each dimension with $q = 100$ bins.

As first competitor, we compare against the coordinate descent method of Wu (2010). The technique was originally presented for fast SVM learning with HIK (see Section 3.1) and directly operates on the look-up table T as introduced in Section 3.2.2.2. In consequence, it can be easily transferred to the scenario of learning GP models equipped with HIK when the quantization trick is additionally applied.

In addition, we chose the greedy block coordinate descent (GBCD) approach presented by Bo and Sminchisescu (2008) to compare against. GBCD calculates parts of the kernel matrix on the fly to solve sub-problems. Thereby, GBCD is more general than our approach and does not place any assumptions on the applied kernel function. We set the size of the sub-problems to 10 and the number of components κ for greedy selection to 20. While we also tested other values, *e.g.*, a sub-problem size of 60 and $\kappa = 500$ as suggested by (Bo and Sminchisescu, 2008) we did not achieve a notable speed-ups.

To compare all three techniques, we follow experiments in Section 7.2.2 and select a binary classification task from the ImageNet dataset. Since GBCD turns out to run extremely slow, we set $\ell = 1$ to evaluate all three techniques in affordable time. All techniques are applied to compute the weight vector α by solving the linear system $K \cdot \alpha = y$. In Fig. 7.12, we visualized the residual of the linear system with respect to the computation time needed. We terminated each technique when the maximum-norm of the residual dropped below 10^{-6} .

Experimental Results: As first interesting observation, we note that the maximum-norm of the residual computed with the CG solution slightly increases within the first iterations. While this seems irritating on first glance, it is not conflicting with the convergence statement given in (Hestenes and Stiefel, 1952, Theorem 6:3) which states the series of approximations for α monotonically approach the exact solution. In fact, this behavior is well known (*e.g.*, see (Fong and Saunders, 2012, Table 5.1)) and does not negatively influence the final outcome.

We further note that there are orders of magnitude between convergence times of all three solvers. While our preferred conjugate gradients method converges after roughly four seconds, the coordinate descent method by Wu (2010) requires more than 30 seconds. Finally, the generic GBCD requires about 16 minutes. Thus, whenever κ_{HIK} is applied, we should prefer a specific implementation over a generic solver.

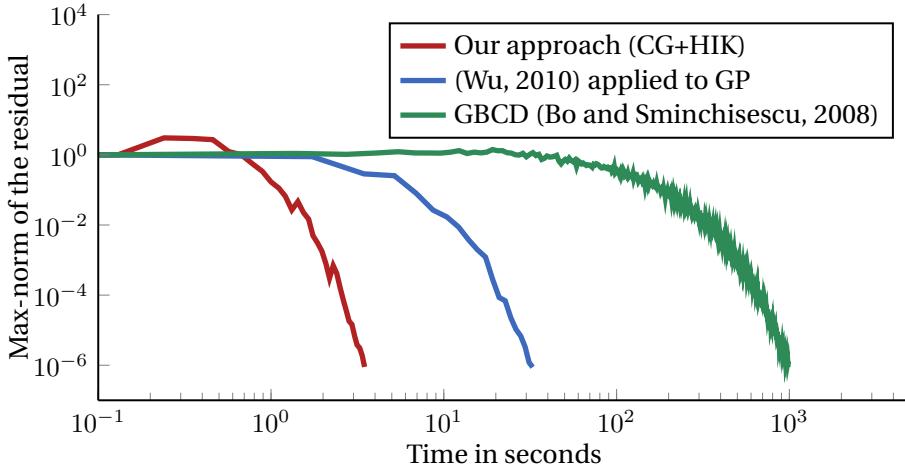


Figure 7.12: Evaluation of the computation time and convergence of linear solvers: (1) our conjugate gradients method, (2) the coordinate descent method of Wu (2010), and (3) greedy block coordinate descent of Bo and Sminchisescu (2008). Note that our approach and (Wu, 2010) exploit fast HIK matrix multiplications, while (Bo and Sminchisescu, 2008) can be applied for every kernel function.

Note that the large computation time of GCBBD is mainly caused by the necessary computations of kernel values between features of dimensionality 10,000. This behavior was less dominant in the experimental evaluation in (Bo and Sminchisescu, 2008). We attribute this difference to the low-dimensionality of features used by Bo and Sminchisescu (2008) ($D \leq 37$).

Summary: In summary, we conclude that the CG method can draw best advantage from our techniques for fast matrix vector multiplications.

7.2.2.5 Early Stopping of the Linear Solver

In the previous experiments, we trained GP/HIK models by running the ILS until convergence, *i.e.*, until the residual dropped below a pre-defined value close to zero or until a pre-defined iteration number was reached. As mentioned in Section 3.2.3.1 and shown by (Hestenes and Stiefel, 1952), we even have the guarantee that we found the exact solution for α after N iterations if round-off errors can be neglected. In some scenarios, however, we might not be interested in computing the exact solution, but rather a sufficiently close approximation thereof. Approximations can be obtained by interrupting the iterative linear solver before convergence, which is known as early stopping.

For learning of large-scale SVMs, Perronnin et al. (2012) demonstrated that early stopping results in advantages both with respect to computation time and

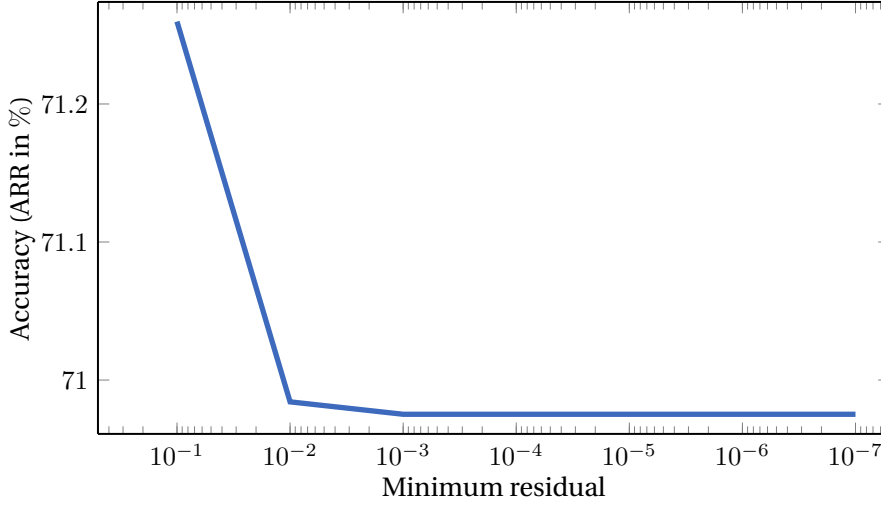


Figure 7.13: Evaluation of model regularization by early stopping of the GP/HIK conjugate gradient method. The x -axis displays stopping values for the iterative linear solver with respect to the infinity norm of the residual. Resulting recognition rate as a measure of generalization abilities are shown on the y -axis.

generalization abilities. According to their experiments, early stopping can be seen as a computational regularization technique with the additional benefit of lower computation costs during training. Similar to iterative SVM solvers, our ILS can be easily interrupted at any time step as well. Let us therefore investigate whether we can successfully transfer the effect of early stopping to GP/HIK models.

Experimental Setup: As in previous experiments, we use the 15Scenes dataset for evaluations. For each category, we select 100 examples for model training as well as 50 examples for model evaluation. The noise variance σ_n^2 is fixed to 0.1. The effect of early stopping is finally evaluated by interrupting the iterative linear solver after reaching different residuals and evaluating the resulting model on the hold-out test set. Length of residuals is measured by the infinity norm:

$$\|\mathbf{x}\|_\infty = \max_{d \in \{1, \dots, D\}} |\mathbf{x}(d)| . \quad (7.6)$$

Results are shown in Fig. 7.13.

Experimental Results: First of all, we notice that the classification accuracy converges with a growing number of iterations for the ILS (*i.e.*, with smaller minimum residuals which need to be reached for stopping). More importantly, this convergence happens quickly, *i.e.*, even if we stop the linear solver with an extremely high residual. In fact, we noted an exponential increase in the required number of itera-

tions with respect to a desired accuracy of the solution. Moreover, early stopping might even lead to higher accuracy. This observation is consistent with the findings of Perronnin et al. (2012) which relates early stopping to model regularization with improved generalization abilities.

Summary: *We conclude that early stopping of the iterative linear solver can significantly save computation time without negative implications on learned models.*

7.2.2.6 Comparison with GP Sparsity Methods

In Section 3.1, we reviewed general approaches for GP inference in large-scale scenarios. One important line of research consists of techniques which approximate exact GP inference by sparse models. Here, sparse refers to a reduction of relevant variables during training, e.g., the number of training examples.

In contrast to these techniques, our GP/HIK provides solutions for *exact* inference in large-scale scenarios at the cost of being restricted to κ_{GHK} as covariance function. In this section, we are interested in comparing our efficient techniques for exact inference against sparse GP approximations. A well-known and powerful technique in the family of sparse GPs is the Fully Independent Training Conditional (FITC) Approximation by Quiñonero Candela and Rasmussen (2005). Since FITC is implemented in the `gpm1` toolbox by Rasmussen and Nickisch (2010), it easily allows for comparisons.

Experimental Setup: We follow the previous experiments and use the 15Scenes dataset. As previously, images are represented using `relu7` features. For training, we sample different numbers of examples for each category. In addition, 50 hold-out examples of each category are used for model evaluations. Again, we average ARR values over ten random splits. For FITC, we evaluate different sizes of the inducing set. The noise level for all GP models was set to $\sigma_n^2 = 0.1$. Results are shown in Fig. 7.14.

Experimental Results: We observe that FITC leads to inferior results compared to exact inference, especially if N is small. Note that results for other CNN layers led to comparable conclusions. Furthermore, the decrease in accuracy is stronger for small inducing set sizes. However, small inducing set sizes are desired for large-scale scenarios,

Summary: *In summary, our techniques for exact GP inference outperform sparse GP approximation techniques.*

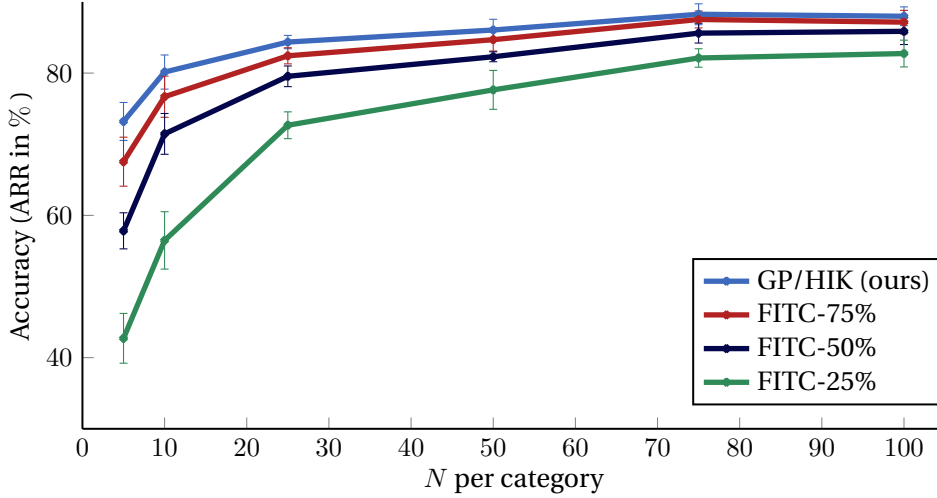


Figure 7.14: Comparison of our approach with the FITC method of Quiñero Candela and Rasmussen (2005) on the 15Scenes dataset with `relu7` features. For FITC, different relative sizes of the inducing set are used.

7.2.2.7 Evaluation of the Quantization

In previously shown experiments on ImageNet, we applied the quantization idea introduced in Section 3.2.2.2 to evaluate GP/HIK models in affordable time. As stated in Section 7.2.2.1, we always used a non-adaptive quantization, which was sensible due to the L_1 -normalized BoW features used. Furthermore, we spent $q = 100$ bins for each dimensions and empirically verified that the classification accuracy remained unchanged compared to the exact GP baseline. In this section, we are interested in analyzing the effects of varying number of bins. In addition, we evaluate the benefit of adaptive quantizations (see Section 3.2.2.3) for features which exhibit activations outside of $[0, 1]^D$.

Experimental Setup: As in previous experiments, we use the 15Scenes dataset and represent images with `relu7` features. For ten random splits, we randomly select 100 examples of each category for training as well as 50 hold-out examples of each category for testing. Classification accuracy is reported as average classification accuracy as done previously. Computation times are measured on a Intel Core i7-3930K CPU desktop computer with 3.2 GHz without any parallelization. In contrast to previously reported computation times, all computation times in this section include conversion overhead which arises from passing features from Matlab data structures to C++ pendants via Mex-interfaces. Results are shown in Fig. 7.15.

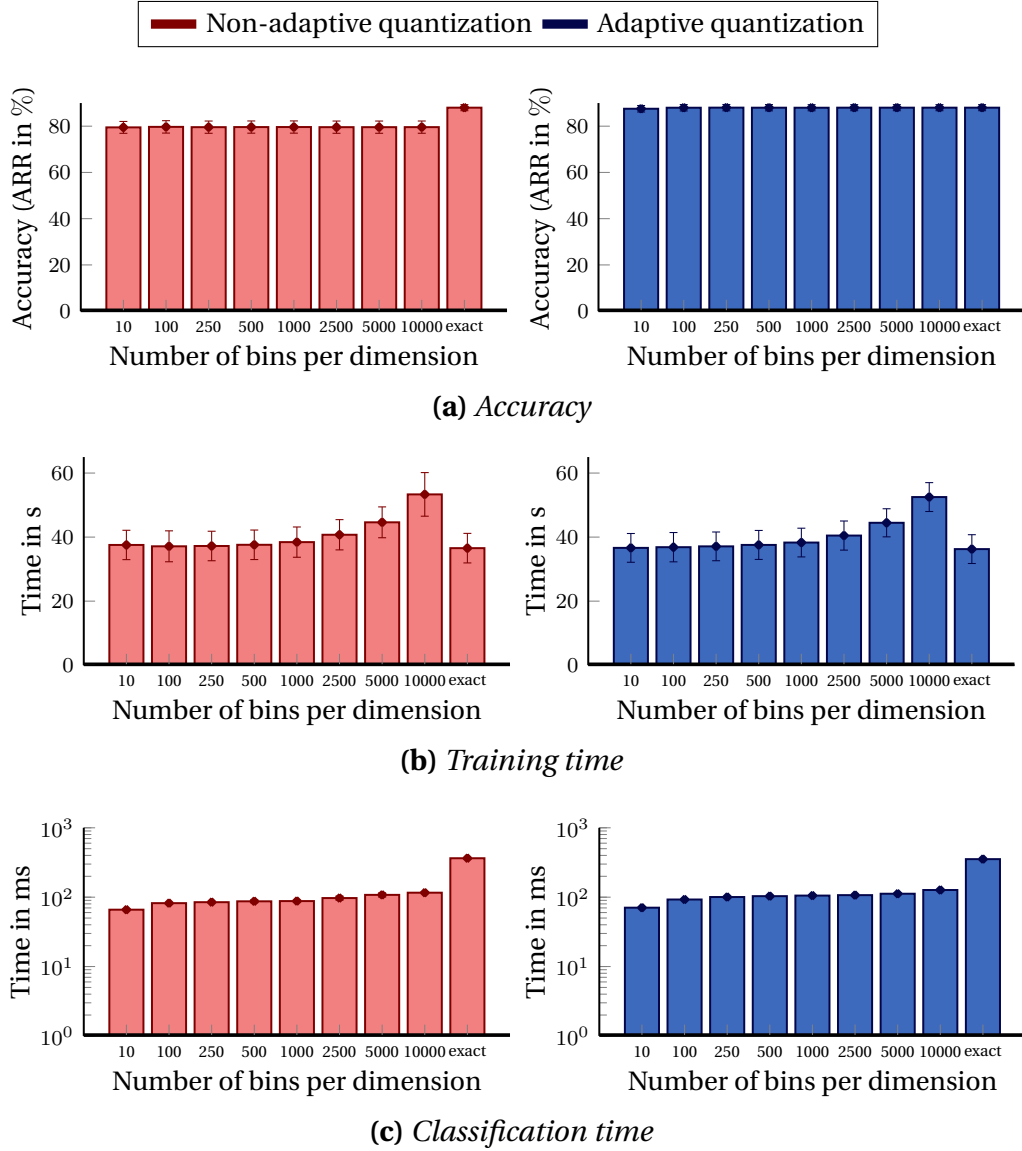


Figure 7.15: Effect of different quantization levels on classification accuracy (top), required computation times during training (middle), and for inference (bottom). Results are obtained on the 15Scenes dataset and averaged over ten random splits. See text for details.

Experimental Results: For the non-adaptive quantization shown left, we observe that the classification accuracy is almost unchanged for different numbers of q . However, it does not reach the accuracy level of exact inference. This is due to `relu7` features which span a larger range than $[0, 1]^D$. Thus, quantizing inputs into the unit interval maps a significant number of feature values to the interval border and thereby wastes valuable information. In contrast, the adaptive quantization shown right can easily resolve this drawback. In fact, with already 100 bins, classification results are exactly identical to those achieved with exact inference.

With respect to training times, we observe that the additional time required by both quantization techniques increases with the number of bins. Nonetheless, the induced overhead for moderately sized LUTs is negligible. A similar trend is visible for classification times: with an increasing number of bins, the computational advantage of the quantization trick is less dominant.

Summary: *In summary, the quantization of test inputs saves valuable classification time with an negligible overhead during training. Classification accuracy is unaffected for already 100 bins per dimension. Besides, the adaptive quantization is applicable to non-normalized features without significant extra costs.*

7.2.3 Experimental Analysis on Hyperparameter Optimization

As introduced in Section 2.1.5.7, the probabilistic formulation of Gaussian process models allow for more than just training and classification as evaluated in the previous section. One such aspect is the capability of optimizing hyperparameters by maximizing the marginal likelihood. We now investigate this aspect for GP/HIK models and our techniques for efficient computations thereof (see Section 3.3). In detail, we will investigate the following questions:

- (i) Section 7.2.3.1: How well does our upper bound match the exact log-likelihood?
- (ii) Section 7.2.3.2: Does optimization of hyperparameters for $\kappa_{\text{HIK-POLY}}$ and $\kappa_{\text{HIK-EXP}}$ improve the classification performance?
- (iii) Section 7.2.3.3: Can we perform hyperparameter optimization on large-scale data?
- (iv) Section 7.2.3.4: Is our optimization technique compatible with early stopping of the iterative linear solver?
- (v) Section 7.2.3.5: Can we apply our optimization technique to determine relevance of feature dimensions?

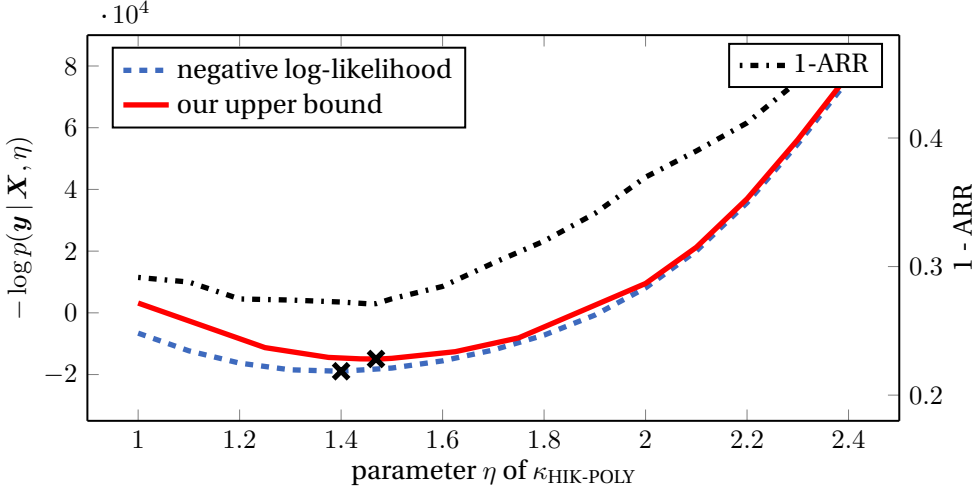


Figure 7.16: Comparison between upper bounded negative marginal log-likelihood, real negative marginal log-likelihood, and resulting average class-wise recognition rate (ARR). Different values for HIK-POLY are shown on the x-axis. The left y-axis shows upper bounded and exact values of the negative log-likelihood for the resulting models. On the right y-axis, we additionally show the resulting classification performance.

7.2.3.1 Verifying the Bound on the Negative Marginal Log-Likelihood

Computing the exact marginal likelihood turned out to be still a costly operation even with efficient kernel evaluations. We therefore introduced an upper bound for the negative log-likelihood in Section 3.3 which was computationally feasible. In a first experiment, we are interested in the quality of this bound, *i.e.*, we need to check if the bound is a close approximation to the exact log-likelihood and if resulting optima are approximately the same.

Experimental Setup: As in the previous experiment, we use the 15Scenes dataset for evaluation. During training and testing, we use 100 and 50 examples per category identically selected to the previous experiment. For different choices of the hyperparameter for HIK-POLY, we evaluate our upper bound after training the model and computed its classification accuracy on the test set. In addition, we train models for each setting following the plain GP formulation, *i.e.*, with explicitly computing the kernel matrix, its Cholesky decomposition, and solving the linear equation system to compute the weight vector α . For each model, we then compute the exact negative log-likelihood to serve as gold standard and also evaluate our upper bound approximation to compare against. Results are shown in Fig. 7.16.

Experimental Results: First of all, we see a surprisingly similar behavior of likelihood (blue curve) and resulting classification accuracy (black curve) which underlines the usefulness of likelihood maximization for optimization of hyperparameters. In addition, we observe that our upper bound (red curve) tightly matches the exact likelihood. Since extremes of both curves are close, we conclude that the bound is sufficient for optimization of hyperparameters in visual recognition scenarios. It is noteworthy that our bound is closer for higher values of η and converges to the exact solution when η is increased. This behavior is due to the fact that large values of η for HIK-POLY (see Eq. (3.2)) lead to well-conditioned kernel matrices. In consequence, the relative importance of the complexity term decreases (see Section 3.3.1) and the likelihood's value is dominated by the data term which we can compute without error even for large-scale datasets. Experiments on other datasets and settings confirmed this observation.

Summary: *In summary, our bound is a close approximation to the exact negative log-likelihood and has similar minima.*

7.2.3.2 Different Generalized HIK for Multi-class Classification

Let us now quantitatively evaluate the resulting benefits of optimizing hyperparameters for generalizations of κ_{HIK} .

Experimental Setup: As a first dataset of medium scale, we again choose the 15Scenes dataset. The setup is identical to the previous experiment with that dataset, *i.e.*, for each category, we select 100 examples for model training as well as 50 hold-out examples for testing. Again, the noise variance σ_n^2 is fixed to 0.1. We compare GP models equipped with κ_{HIK} and two generalizations thereof (HIK-POLY and HIK-EXP) when optimizing their hyperparameters using our upper bound. Linear and kernelized SVM models serve as baselines and results from the literature reported for different image representations are shown. As representations for images, we compare two general settings: one with BoW features and one with activations of the AlexNet CNN using the Caffe toolbox by Jia et al. (2014). Results for all settings are shown in Table 7.3.

Experimental Results: Let us start with the upper part of Table 7.3, where we focus on BoW features as image representations. First of all, we again observe a significant performance boost when optimizing hyperparameters of generalized histogram intersection kernels, both for $\kappa_{\text{HIK-POLY}}$ and $\kappa_{\text{HIK-EXP}}$. Surprisingly, results of $\kappa_{\text{HIK-EXP}}$ with GP models are even on par with the spatial pyramid matching kernel (SPMK) when plugged into an SVM, although no position information is included in the image representations the GP model is trained with. Thus, we

Table 7.3: *Evaluating the effect of optimizing hyperparameters for $\kappa_{\text{HIK-POLY}}$ and $\kappa_{\text{HIK-EXP}}$ on the resulting classification accuracy using the 15Scenes dataset. While the upper part shows results for standard BoW features, results of the lower part are computed from CNN activations. Compare also against non-optimization results in Table 7.2.*

Method	Features	ARR
GP/HIK	BoW	70.8%
GP/HIK-POLY	BoW	72.9%
GP/HIK-EXP	BoW	74.0%
GP/HIK	AlexNet-relu7	86.42%
GP/HIK-POLY	AlexNet-relu7	86.42%
GP/HIK-EXP	AlexNet-relu7	80.04%

can conclude that the generalizations of κ_{HIK} are well suited to serve as adjustable similarity measure in visual recognition scenarios.

In the lower part of Table 7.3, we report results on `relu7` extracted from AlexNet. Interestingly, we find that generalizations of κ_{HIK} do not result in additional performance gains anymore. We attribute this behavior to the learned representations which are already expressive enough on their own. Thus, they do not require additional flexibility introduced by adjustable similarity metrics.

Summary: *In summary, our optimization technique allows for tuning of hyperparameters if required by given data.*

7.2.3.3 Different Generalized HIK in Binary Classification Tasks

In the last section, we successfully evaluated the optimization of hyperparameters on a small-scale dataset. Let us now investigate the scalability of our hyperparameter optimization technique with respect to dataset sizes.

Experimental Setup: We therefore follow the same experimental setup as in Section 7.2.2 where we compared GP models in their plain formulation with our more efficient GP/HIK techniques (see Table 7.1). The only difference is that we again apply generalizations of κ_{HIK} and optimize their hyperparameters based on the bound for the negative log-likelihood. As in the previous experiment, we allow for adjustable polynomial transformations (GP/HIK-POLY) as well as an exponential transformation (GP/HIK-EXP). We additionally compare against the baseline κ_{HIK} as evaluated in Table 7.1. Results are shown in Table 7.4.

Table 7.4: *Evaluating the effect of optimizing hyperparameters for $\kappa_{\text{HIK-POLY}}$ and $\kappa_{\text{HIK-EXP}}$ on the resulting classification accuracy using the ImageNet dataset. As in Table 7.1, we report results averaged over 200 binary classification tasks derived from the ImageNet database. In contrast to Table 7.1, we now optimize hyperparameters for both kernel functions. Computation times are given as median values of measurements for each task (learning) and each test example (classification).*

Method	10,090 examples ($\ell = 10$)			50,050 examples ($\ell = 50$)		
	AUC	learning time	classif. time	AUC	learning time	classif. time
GP/HIK (see Table 7.1)	0.836	64 s	44 μ s	0.856	321 s	44 μ s
GP/HIK-POLY	0.865	435 s	44 μ s	0.883	2,815 s	44 μ s
GP/HIK-EXP	0.889	579 s	44 μ s	0.893	2,578 s	44 μ s

Experimental Results: We observe that our upper bound approximation of the negative log-likelihood is indeed applicable to datasets with tens of thousands of examples. Thereby, we can achieve additional performance gains which are statistically significant according to a Paired-sample t-test with $p < 10^{-7}$. This again underlines the benefit of adjustable similarity metrics.

Summary: *In summary, our introduced approximation of the negative log-likelihood is applicable to scenarios with tens of thousands of training examples.*

7.2.3.4 Early Stopping of the Linear Solver When Hyperparameters Are Optimized

We already investigated the effect of early stopping for κ_{HIK} in Section 7.2.2. In this section, we are interested in whether early stopping is still applicable if optimization of hyperparameters is activated. To this end, we investigate whether the found hyperparameter setting is stable even if the iterative linear solver for model training is not run until convergence.

Experimental Setup: We chose the experimental setup identical to the previous experiment to allow for a direct comparison. Thus, we use the 15Scenes dataset with 100 and 50 examples per category for training and testing, respectively. For different values of the ILS stopping criterion, we optimize hyperparameters of $\kappa_{\text{HIK-EXP}}$ without applying the quantization trick. Learned models are finally evaluated on the test set and accuracy values are plotted against the minimum residual for early stopping. Results are shown in Fig. 7.17.

Experimental Results: As for the first evaluation with the standard κ_{HIK} in Fig. 7.13, we again observe a fast convergence of classification accuracy with a growing num-

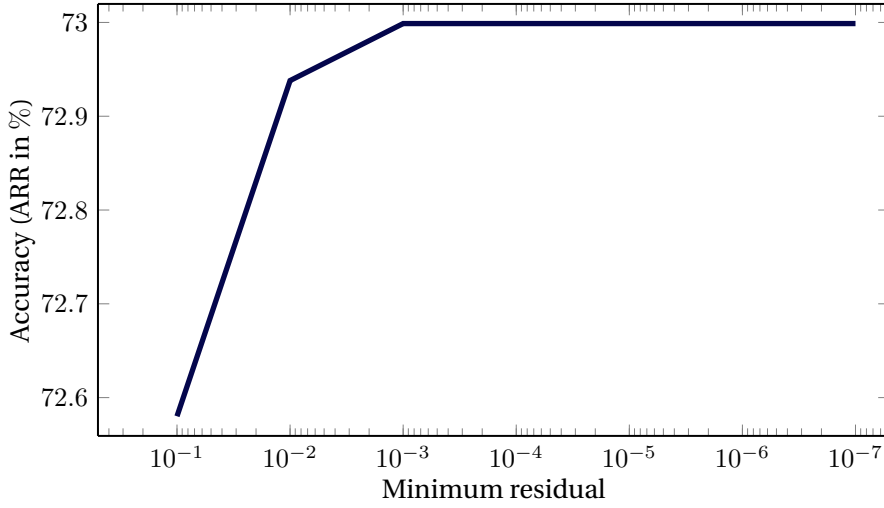


Figure 7.17: Evaluation of model regularization by early stopping of the GP/HIK conjugate gradient method. The experimental setup is identical to but the replacement of κ_{HIK} with $\kappa_{\text{HIK-POLY}}$ and optimization of its hyperparameter. Stopping values for the iterative linear solver with respect to the infinity norm of the residual are displayed on the x -axis, whereas the y -axis shows the resulting recognition rate as a measure of generalization abilities.

ber of iterations for the ILS. Interestingly, the optimized hyperparameter was exactly the same for all evaluated stopping criteria ($\eta \sim 1.48$ in our experiments). Thus, we can also speed-up training with involved optimization of hyperparameters by early stopping to save valuable computation time. In contrast to the previous experiment with fixed hyperparameters, we observed no additional increase in accuracy for smaller residuals.

Summary: In summary, early stopping is also applicable with activated hyperparameter optimization and saves valuable computation time during training.

7.2.3.5 Feature Relevance Determination

The previous experiments proved the suitability of our provided bound for the optimization of hyperparameters. Let us now apply the same techniques for adapting generalized histogram intersection kernels with dimension-specific weights. As introduced in Eq. (2.125), relative importance of feature dimensions can be estimated by optimizing dimension-specific weighting parameters. For the histogram intersection kernel, this is directly possible by applying dimension-specific weighting-functions as shown in Eq. (3.4).

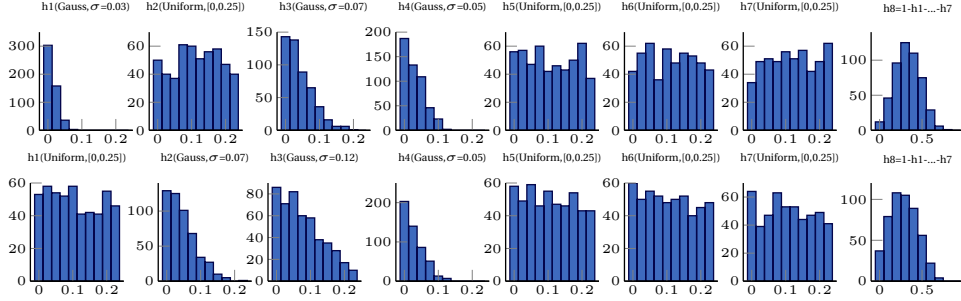


Figure 7.18: *Distributions of two categories (top and bottom) used to sample synthetic, eight-dimensional features as done by Ablavsky and Sclaroff (2011). For each category, 500 examples are sampled from the dimension-specific and class-specific distributions (specified in title texts) and statistics of drawn values are shown as histograms. Note that by design, only the first two dimensions are truly informative to tell both categories apart.*

Experimental Setup: We follow Cai et al. (2010) and Ablavsky and Sclaroff (2011) and conduct experiments in a controlled environment. Thereby, we can evaluate whether or not our optimization technique is able to figure out relevant dimensions and we can immediately observe the resulting benefit on the classification performance. In detail, we synthesize eight-dimensional features for two categories. We design the sampling process such that distributions of features differ significantly for the first two dimensions, differ slightly for the third dimension, and are identical for entries of dimension four to seven. Normalization properties are kept by making the 8th dimension dependent to sum up to one. We sample features from uniform distributions over a given interval or a Gaussian distribution with zero mean and given variance and return absolute values thereof. If a sample resulted in negative values for the last dimension, its sampling process is repeated to guarantee well-formed, histogram-like features. Parameters of all distributions as well as resulting statistics after sampling 500 examples for each category are shown in Fig. 7.18.

To evaluate the effect of automatic relevance determination, we sample five training examples for each of the two categories. With the resulting training sets, we learn GP models equipped with $\kappa_{\text{HIK-ARD}}$ and optimize hyperparameters using the Nelder-Mead method and our introduced upper bound for the negative log-likelihood. The entire process is repeated 500 times to decrease the influence of randomization. Learned weights averaged over all 500 runs are finally shown in Fig. 7.19 (red bars).

Besides the inspection of learned weights, we also evaluate learned models on 150 hold-out test examples per category. To allow for direct comparison with results of Ablavsky and Sclaroff (2011), we report mean classification accuracy as a measure of performance. Again, we average results over 500 iterations of dataset sampling,

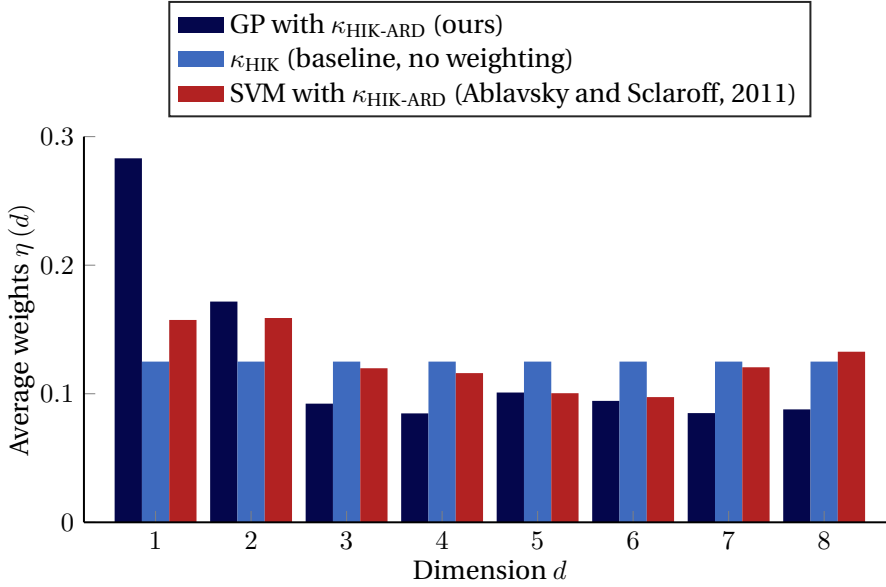


Figure 7.19: *Learned relative weights for each dimension for $\kappa_{\text{HIK-ARD}}$ in a synthetic experiment following the evaluation protocol of Ablavsky and Sclaroff (2011). Weights are averaged over 500 repetitions of feature sampling and model training.*

model training, and evaluation for each size of the training set. Classification accuracy as a function of the training size is visualized in Fig. 7.20.

Experimental Results: As we can see, our optimization technique consistently assigns higher weights to discriminative dimensions. In addition, we can compare against the weights reported by Ablavsky and Sclaroff (2011) which are shown in Fig. 7.19 as blue bars. These weights were obtained by training an SVM and optimizing hyperparameters with the optimization technique introduced by Ablavsky and Sclaroff (2011). Notably, our learned weights consistently reflect the actual relevance of all dimensions better than the weights learned by Ablavsky and Sclaroff (2011).

We further note that the accuracy increases with the number of training examples. Furthermore, we observe that estimating feature relevance consistently gives a performance boost for all training set sizes. While the improvement seems small in absolute numbers, it is highly statistically significant according to a paired-sample t-test (see Section 7.1.2.3) with $p < 10^{-7}$.

Summary: *In summary, the probabilistic formulation of GP models is beneficial for feature relevance determination. However, an extension to thousands of hyperparameters for handling high-dimensional features is still an open issue.*

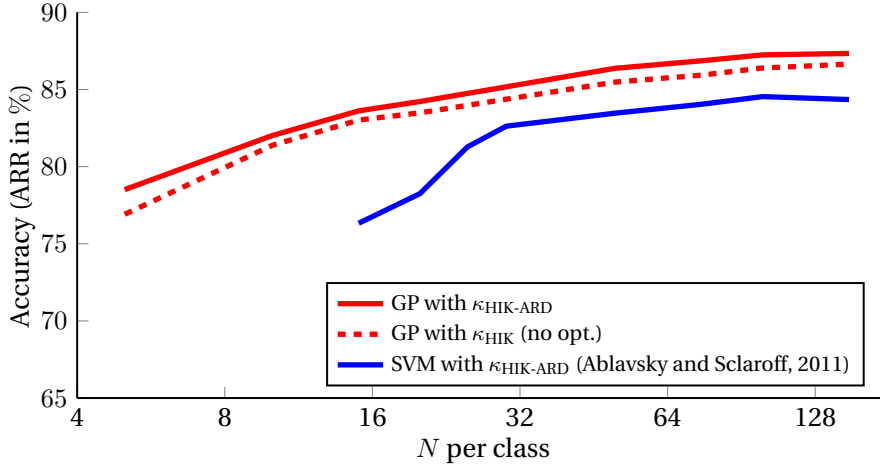


Figure 7.20: Accuracy as a function of the training set size when dimension-specific weights are optimized for $\kappa_{\text{HIK-ARD}}$. The experiment uses synthetic data following the evaluation protocol of Ablavsky and Sclaroff (2011).

7.2.4 Experimental Analysis on Uncertainty Prediction

In all evaluations shown so far, we were interested in training GP models, optimizing their hyperparameters, and applying them to classification scenarios. One important aspect which we did not consider so far is the possibility to compute uncertainty estimates along with pure classification decisions. As reviewed in Section 2.1.5, uncertainty estimates of GP models are directly related to the predictive variance.

In Section 3.4, we presented several techniques to compute or approximate the predictive variance by exploiting histogram intersection kernels. Similar to the previous experiments, we will therefore evaluate two important questions:

- (i) Section 7.2.4.1: What is the speed-up compared to the exact baseline?
- (ii) Section 7.2.4.2: How tight are the introduced approximations?

As for experiments in the previous section, source code is publicly available at <https://github.com/cvjena/gp-hik-core.git>.

7.2.4.1 Computation time Analysis for Variance Approximations

Since the predictive variance requires computation times quadratic in the number of known examples in their plain formulation, we introduced several approximations thereof in Section 3.4. From a theoretical perspective, these techniques allow for uncertainty prediction even in large-scale scenarios with tens of thousands of

Table 7.5: *Computation times needed for computing the predictive variance and approximations thereof. Experiments are conducted on the 15Scenes dataset (left) as well as ImageNet with 10 and 50 training examples per negative category (middle and right). As baseline serves a GP model in plain formulation with explicit computations. * not possible due to excessive memory demand.*

Method	Number of training examples		
	1,500	10,090	50,050
GP, plain formulation	60.36 ms	1.54 s	—*
PUP (Section 3.4.1)	2.11 s	22.92 s	> 1 min
FAPU (Section 3.4.2), $k = 8$	13.73 ms	105.63 ms	1.47 s
FAPU (Section 3.4.2), $k = 2$	13.35 ms	105.37 ms	1.15 s
CAPU (Section 3.4.3)	5.92 ms	62.07 ms	266.97 ms
q-CAPU (Section 3.4.3)	13.32 μ s	33.89 μ s	33.89 μ s

training examples. Let us now investigate which speed-up we can actually obtain by applying these techniques in practice.

Experimental Setup: We present experiments with identical setups as done in previous evaluations in Section 7.2.2 and Section 7.2.3. Thus, we use both the 15Scenes dataset as well as ImageNet. Note that the predictive variance is independent of actual class labels (see Eq. (2.100) in Section 2.1.5.3). Thus, training a single one-vs-all model is sufficient for this evaluation since we obtain identical uncertainty estimates as if all binary models would have been trained.

For the 15Scenes dataset, we again select 100 images of each category for training as well as 50 additional hold-out images of each category for testing. On ImageNet, we apply the same setup as used for previous binary experiments. Thereby, we use 100 examples of a single category as positive examples. In addition, we randomly select 10 or 50 examples for each of the remaining 999 categories as negatives.

After training, we compute predictive variance estimates and its approximations on all test images and average required computation times. Experiments are conducted on a 3.4 GHz CPU without any parallelization. All results are shown in Table 7.5.

Experimental Results: First of all, we note that the exact method PUP is extremely slow – even orders of magnitude slower than the baseline with explicit computations. Large computation times are mainly caused by the required number of iterations of the ILS. In fact, one computation of PUP requires approximately the same time as one run of standard model training for a binary scenario (without optimization of hyperparameters). Nonetheless, it is still computable even if the

sheer number of training examples would render the corresponding kernel matrix to be too large for the available memory. Thus, the precise method should only be chosen in scenarios where explicit computations are not possible due to memory limitations but exact uncertainty estimates are required.

In contrast to PUP, we observe significant speed-ups over the baseline technique for all introduced approximations. Especially in real-time applications, where not the exact value but only an estimate thereof might be sufficient, the CAPU technique is highly suitable with computation times in the order of microseconds.

Summary: *In summary, our introduced variance approximations come with different trade-offs regarding accuracy, computation time, and memory and can be chosen according to the current application constraints.*

7.2.4.2 Investigating the Tightness of Variance Approximations

In a second experiment, we investigate how strong our approximations deviate from exact variance estimates. To this end, two requirements are possible: (1) approximated values should be as close as possible to exact predictions (as for security applications with fixed thresholds), or (2) only the relative order is of interest, although the approximated scores can deviate from the exact values (as for active learning). We analyze both settings in the following.

Experimental Setup: As previously, we use the 15Scenes dataset for evaluation and represent images using `relu7` features. To analyze the effect of normalized features, we additionally report results after normalizing all features to L_1 -length one. For training, we select three categories with 50 randomly chosen examples of each category. All remaining data of the dataset serves for model evaluation. In consequence, the test set contains a small set of examples from known categories and a significantly larger pool of unknown categories. The noise level for model regularization is fixed to $\sigma_n^2 = 0.1$. As in the previous evaluation, we train a single one-vs-all model since we only care about scores of the predictive variance which are independent of provided class labels.

Besides the exact PUP technique, we apply FAPU with varying numbers of eigenvectors. To robustly estimate even 16 eigenvectors, we spend $T_3 = 100$ iterations for the Arnoldi technique. In addition, we apply CAPU without and with quantization. For activated quantization, we apply an adaptive quantization with $q = 100$ bins per dimension. We finally sort variance scores of the exact PUP technique increasing order. Predictions of the remaining techniques are re-arranged accordingly. Results are shown in Fig. 7.21. For visualization, only every 5th sample is plotted.

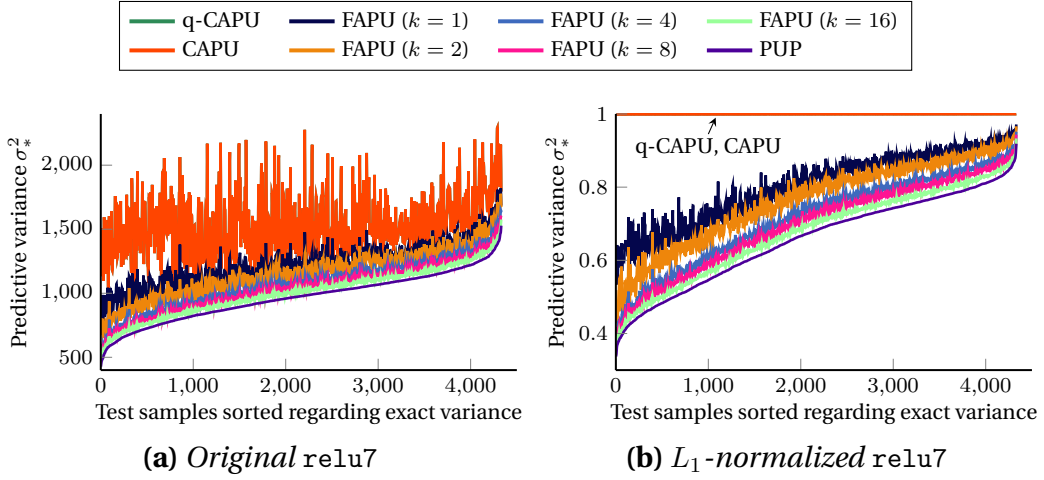


Figure 7.21: Tightness of GP/HIK variance approximations on the 15Scenes dataset with relu7 features. Scores of the exact method are increasingly ordered. Predictions of remaining techniques are re-ordered accordingly. Left: GP/HIK trained on original relu7 activations. Right: relu7 activations are additionally normalized to L_1 -length one. See text in Section 7.2.4.2 for details. Best viewed in color.

Experimental Results: As we can see, the exact but slow PUP technique is closely matched by the approximated values obtained with FAPU. Furthermore, the strict upper bound relation between PUP, FAPU with decreasing numbers of eigenvectors, and CAPU is clearly visible. For normalized features, we further note that variance estimates are mapped into $[0, 1 + \sigma_n^2]$. Thus, they can be approximately used as confidence measure in practice. Non-normalized features require further post-processing of values if probability-like values are intended. Besides, we observe that q-CAPU and CAPU lead to identical results which again underlines the usefulness of the quantization idea. Nonetheless, the absolute range of variance predictions is extremely small for both methods. We can thus conclude that FAPU should be chosen if close approximations of the exact values are required.

To better analyze whether the relative order of q-CAPU and CAPU is comparable to remaining approximations, we individually normalize variance estimates of all methods to the unit interval. In detail, we use smallest and largest estimates of each method on all test examples for visualization, which we call the normalized predictive variance. Results are shown in Fig. 7.22.

As in the previous evaluation, we again observe that FAPU is closely related to PUP. With respect to the coarsest approximation CAPU, we now find a similar trend. Thus, although absolute values are extremely different from exact estimates, the absolute order follows a similar distribution.

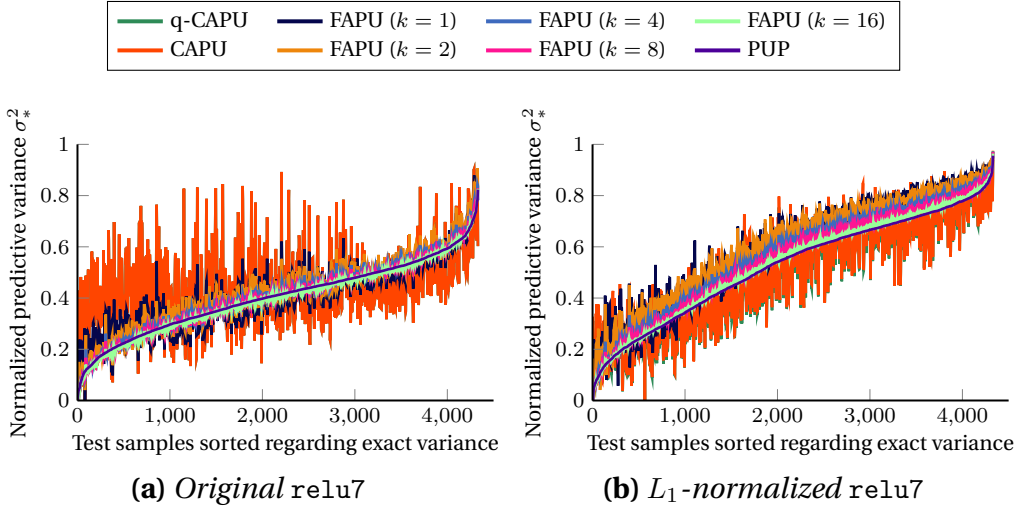


Figure 7.22: An investigation of the tightness of GP/HIK variance approximations on the 15Scenes dataset with `relu7` features. In contrast to Fig. 7.21, we normalized variance predictions of each method individually into $[0, 1]$ to better visualize the overall trend. The remaining setting is identical to Fig. 7.21. Best viewed in color.

Summary: In summary, FAPU leads to close approximations and should be used when approximation errors are undesirable. If only the relative order of variance estimates is required, CAPU is a reasonable choice.

7.2.5 Experiments on Semantic Segmentation with Scalable Gaussian Process Models

In a final experiment, we are interested in applying our GP/HIK models to a large-scale classification task which is as interesting as challenging: the task of semantic segmentation. We already gave an introduction to semantic segmentation in Section 3.5.1 and reviewed the framework of Csurka and Perronnin (2010) which we will apply within the following evaluations.

It should be emphasized again that this framework is *not* state-of-the-art (see (Long et al., 2015) for recent impressive with deep architectures). However, the intended purpose of this experiment is not to obtain state-of-the-art results. Instead, we are interested in directly comparing the non-linear GP/HIK against linear competitors as classifier for local features.

Experimental Pipeline: As visualized in Fig. 3.6, we need to specify five key ingredients. We will mainly follow the suggestions of Fröhlich et al. (2010). Thus, we apply the unsupervised graph-based segmentation by Felzenszwalb and Hut-

tenlocher (2004) in step (1). For a given image, local OpponentSIFT features are extracted on regular grids and multiple resolutions of the original image in step (2). Feature enhancement in step (3) is done as introduced in Section 3.5.2 by converting local features into a vector over assignment probabilities of GMM components which have been trained previously. During the learning phase, we rely on a set of images which are labeled pixelwise according to the category each pixel belongs to (see most right column of Fig. 7.23 as a visual example). Thereby, we can easily compute tens of thousands of local features as training examples with only several labeled images. We then train both the GMM for feature enhancement as well as a GP/HIK as final classifier. As outlined in Section 3.5.2, enhanced features allow us to select κ_{HIK} as covariance function and thus to apply our techniques for efficient training and classification.

For test images, steps (1) to (3) are performed identically to the training step. In step (4), all resulting features are classified with the previously trained GP/HIK. Classification results are combined in step (5) by averaging class scores over all pixels which belong to the same region computed in step (1). For each region, the category with highest score determines the region's class label. This label is assigned to all corresponding pixels within that region to obtain the final segmentation output.

Experimental Details: As for the choice of the semantic segmentation framework, we again follow Fröhlich et al. (2010) regarding the experimental setup to allow for fair comparisons. Thus, we use the datasets eTRIMS and LabelMeFacade as introduced in Section 7.1.1.8. As suggested by Yang and Förstner (2011b), we use 10 random splits of eTRIMS into 40 and 20 images for training and testing, respectively. For the LabelMeFacade dataset, we use the official split as suggested by (Fröhlich et al., 2010). All other settings for steps (1) to (3) as well as step (5) are identical to the setup described in Fröhlich et al. (2010). Consequently, we only differ in the chosen classifier which allows for meaningful comparisons of applied models. As adjustable kernel function, we follow our previous evaluations and apply $\kappa_{\text{HIK-POLY}}$ as introduced in Eq. (3.2).

We additionally found that the numbers of training examples is highly imbalanced among existing categories. As an example, pixels of category “door” occur rarely, whereas “building” is significantly over-represented. To avoid sample bias, we select all examples of the smallest category and randomly sample as many examples of each remaining category. Note that due to the sheer number of examples available for training, applying a GP model in its plain formulation is not possible.

For both datasets, we compare against baselines reported in the literature. Most similar to our setup are the approaches by Fröhlich et al. (2010) and Csurka and Perronnin (2010). Both follow the same framework but apply a random decision

forest classifier (Fröhlich et al., 2010) or a sparse logistic regression (Csurka and Perronnin, 2010; Fröhlich et al., 2010) instead of our GP/HIK model. The effect of post-processing RDF results with a conditional random field as well as its hierarchical extension was investigated by Yang and Förstner (2011b) and Yang and Förstner (2011a), respectively. Finally, we also compare against (Fröhlich et al., 2012a) where a multitude of additional features is introduced. Most notably, additional context features are presented to capture high-order dependencies of categories for nearby pixels.

Since all pixels of all test images serve as test examples, we report accuracy as overall recognition rates (see Eq. (7.1) in Section 7.1.2.1). Furthermore, to reflect imbalanced class distributions in test sets, we additionally report average recognition rates (see Eq. (7.2) in Section 7.1.2.1).

Experimental Results: Qualitative results for both datasets are shown in Fig. 7.23. Hold-out test images and segmentation outputs of the pipeline by Fröhlich et al. (2010) with a GP/HIK model as classifier are given in the left and center column, respectively. Classification results are color coded with color-class-correspondences as shown in the bottom of Fig. 7.23.

Let us compare our results with the ground truth obtained from human annotators (right column). As we can see, our non-linear classifier is able to reliably capture the overall layout of the scene. This is indeed remarkable since only local features are classified independently in the utilized framework. Furthermore, we can nicely see limitations of the current approach: the house's roof in the second row is classified as car which can be corrected when additional context information is included as proposed by Fröhlich et al. (2012a). Interestingly, also the human-provided ground truth segmentation contains significant errors in both datasets. Typical examples are missed objects (*e.g.*, bottom left window in first image or majority of distant windows in fourth image), incorrect label assignments (*e.g.*, lanterns labeled as vegetation in third image), and missing labels (*e.g.*, the garden gate in the first image should be labeled as door).

Additionally, we present quantitative results of the evaluations in Table 7.6 for both datasets. Let us first look into the results on the eTRIMS dataset. With respect to the ARR measure, our results are clearly superior to those of previous methods which follow the same framework. Interestingly, also the incorporation of context features (ICF) does not improve over this result by its own and only the addition of several other steps can result in significant improvements (ICFHS-). At the same time, all approaches perform comparable with respect to the biased ORR measure.

For the more challenging LabelMeFacade dataset, results are slightly different. Here, our GP/HIK models leads to significant performance gains with respect to both measures. We thus conclude that the non-linear GP/HIK is able to handle the



Figure 7.23: Qualitative results for semantic segmentation with GP/HIK models. We parse unseen test images (left) through the pipeline as introduced in Section 3.5. The resulting labeling with respect to each pixel's category as one of eight distinct classes is color coded (middle). The color-category-relation for all categories is shown at the bottom. As ground truth serve human annotations of the provided scenes shown in the right column. First two rows: images from the eTRIMS dataset. Remaining two rows: images from the LabelMeFacade database.

Table 7.6: Quantitative results of semantic segmentation on two different datasets: *eTRIMS* (upper part) by Korc and Förstner (2009) and *LabelMeFacade* (lower part) by Fröhlich et al. (2010).

Approach	ARR	ORR
eTRIMS		
RDF (Fröhlich et al., 2010)	58.67% (± 3.14)	64.54% (± 1.37)
SLR (Fröhlich et al., 2010)	65.57% (± 2.47)	71.18% (± 2.69)
CRF (Yang and Förstner, 2011b)	49.75%	65.80%
HCRF (Yang and Förstner, 2011a)	61.63%	69.00%
GP/HIK (ours)	68.68% (± 2.07)	68.02% (± 2.10)
ICF Fröhlich et al. (2012a)	68.61% (± 1.71)	70.81% (± 1.32)
ICFHS- Fröhlich et al. (2012a)	72.26% (± 3.25)	76.10% (± 1.24)
LabelMeFacade		
RDF (Fröhlich et al., 2010)	44.08% (± 0.45)	49.06% (± 0.52)
SLR (Fröhlich et al., 2010)	42.81% (± 0.89)	48.46% (± 1.58)
GP/HIK (ours)	51.18% (± 0.08)	54.01% (± 0.21)
ICF Fröhlich et al. (2012a)	49.39% (± 0.48)	60.68% (± 0.72)
ICFHS- Fröhlich et al. (2012a)	57.82% (± 0.19)	64.76% (± 0.78)

label noise present in training data well enough to result in useful segmentation results. Again, we observe that context features on their own do not significantly improve over our results, but the incorporation of additional features and post-processing steps does.

Summary: *In summary, we can apply GP models to semantic segmentation tasks without the necessity of sparse approximation techniques. Furthermore, replacing classifiers in existing frameworks for semantic segmentation with GP models can pay off and result in potential performance gains.*

7.3 Experiments on Active Learning for Visual Object Recognition

In the previous evaluations, we assumed all training data to be given and evaluated the scalability with respect to increasing dataset sizes. We now investigate scenarios where the initial set of training examples is potentially small but increases over time. Particularly, we evaluate active learning techniques as introduced in Chapter 5. Thereby, the learning process is steered towards new examples which are expected to contain additional information.

As in the theoretical analysis in Chapter 5, we divide experiments into two main application scenarios. In Section 7.3.1, we cover active learning for binary classification as introduced in Sections 5.3 to 5.5. Our techniques for multi-class classification with potentially unnameable instances are covered in Section 7.3.2 and are based on Section 5.6. We made the source code used in this section publicly available at <https://github.com/cvjena/gpal.git>.

7.3.1 Experiments in Binary Classification Scenarios

Let us start our investigations of active learning in binary classification tasks with corresponding query functions. In detail, we will investigate the following questions:

- (i) Section 7.3.1.1: Do query strategies select intuitively useful examples in controlled setups?
- (ii) Section 7.3.1.2: What is the accuracy gain of active learning techniques on ImageNet?
- (iii) Section 7.3.1.3: What is the accuracy gain of active learning techniques on Caltech-256?
- (iv) Section 7.3.1.4: Are selected examples from real-world datasets qualitatively sensible?
- (v) Section 7.3.1.5: How much does EMOC benefit from perfectly estimated class labels?
- (vi) Section 7.3.1.6: Should we especially focus on dense regions of input space?

7.3.1.1 Visual Inspection in a 2D Problem

We start by visually analyzing preferred examples of different active learning strategies. Our goal is to get a first intuitive understanding of inherent advantages and disadvantages of each technique. This analysis extends the visualizations shown in Fig. 5.1, Fig. 5.4 and Fig. 5.6 of Chapter 5.

Experimental Setup: As we will show in Section 7.3.1.4 and Section 7.3.2.5, investigating selected images in *real-world* active learning scenarios can only provide limited intuitive understanding of a query strategy. This non-interpretability is caused by the complex pipeline of data compression which is required to compress raw images into feature vectors (which are usually still high-dimensional). Therefore, we start our analyses in a controlled setup.

In detail, we create a synthetic scenario with 2D data and two categories. We design data distributions of known and unlabeled examples to reflect several properties relevant for real-world scenarios:

- *Few initially known examples*: The number of labeled data is too small for creating a representative dataset. Thus, initial decision boundaries are far from optimal.
- *Initially unknown clusters*: Data distributions of both categories are spread over multiple clusters. Only few clusters are initially known.
- *Existence of outliers*: In addition to examples of both categories, measurement errors lead to outliers far off the main sample distribution.

With respect to few labeled examples, we decided to start with only a single labeled example of each category. Thereby, we can visually inspect if an active learning criterion is able to improve initial decision boundaries to classify known regions more reliable.

To analyze the ability of discovering new clusters of data, we added two unknown clusters to the unlabeled pool. Furthermore, nine outliers are added to investigate the preference for exploration. In total, we obtain 120 unlabeled examples to query from.

As in real applications, we restrict the labeling budget to a small fraction of the available unlabeled pool. In detail, each criterion is allowed to select five queries which visualizes preferred regions in space.

Data points are represented as two-dimensional data and plotted according to their x and y coordinates to allow for optimally visualizing the feature space. Due to the thesis' focus, we apply Gaussian process label regression models as classification technique. An RBF kernel as introduced in Section 2.1.4.3, Eq. (2.76) serves as covariance function. The noise variance for model regularization is fixed to $\sigma_n^2 = 0.1$. For reproducing this analysis, see the Matlab script `visualizeAL2D.m` in the released source code.

We analyze active learning criteria for Gaussian process models as introduced in Chapter 5. In detail, we investigate the explorative technique based on large predictive variances ($Q_{GP-\sigma^2}$, see Section 5.3.1, Eq. (5.14)), as well as the exploitative criterion Q_{GP-unc} (see Section 5.3.1, Eq. (5.15)). Furthermore, we apply the estimated empirical risk minimization strategy transferred to GP Q_{GP-ERM} (Eq. (5.12), Section 5.2.4) with the log-loss as suggested by Roy and McCallum (2001).

Additionally, we analyze our introduced techniques based on estimated model changes $Q_{GP-\Delta\theta}$ (Section 5.4.2, Eq. (5.22)) and estimated model output changes $Q_{GP-\Delta f}$ (Section 5.5.2, Eq. (5.44)), respectively. The oracle gold standard is represented by $Q_{GP-AUC-Oracle}$. To this end, we assume all labels of unlabeled examples to

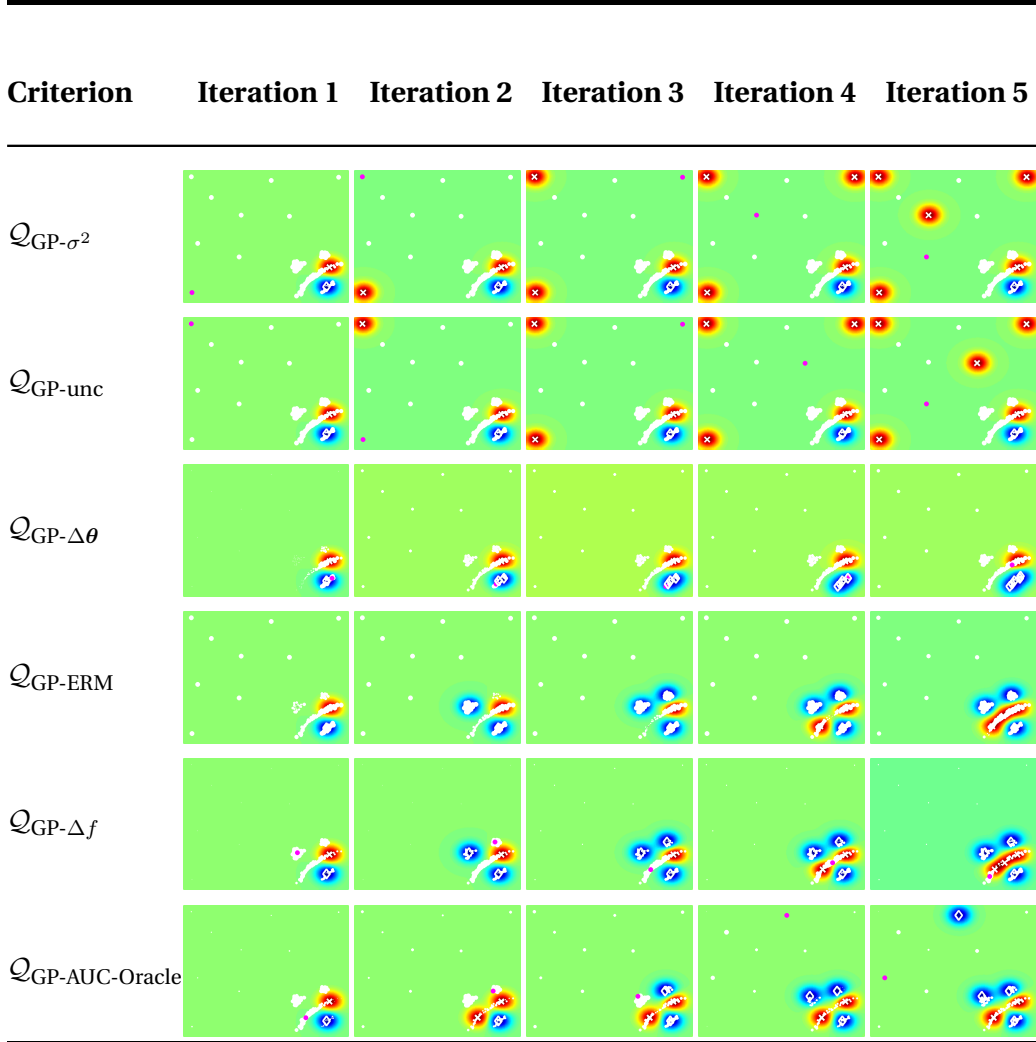


Figure 7.24: Visual analysis of active learning criteria in a 2D scenario. Known examples of positive and negative category are shown as cross and diamond, respectively. Classification scores derived from the currently trained model are color-coded using Matlab's *Jet colormap*. White dots represent unlabeled examples. The diameter indicates the preference of a specific criterion (large diameters indicate strong preference). The example with best score in every iteration is chosen and marked in purple. Best viewed in color and by zooming in.

be known during selection and prefer examples with strongest improvements of AUC values estimated on all available data. Results after five query steps are shown in Fig. 7.24.

Experimental Results: Regarding the predictive variance strategy (first row in Fig. 7.24), we observe a strong focus on outliers. The explorative design of this strategy is clearly reflected in the chosen examples. We thus conclude that using the predictive variance as indicator for gainable information is only useful if outliers can rarely be expected.

As already stated in Section 5.3.1, also the classification uncertainty is distracted by outliers (second row in Fig. 7.24). Although this strategy is motivated as improving decision boundaries, the zero mean assumption leads to large decision boundaries far off the main data distribution for the applied models. See also the explanation given in Section 5.3.1, Fig. 5.2 regarding this phenomenon. Note that this effect is also possible for different classification models, *e.g.*, Kernel-SVMs as used by Tong and Koller (2002).

In contrast to the criteria introduced by Kapoor et al. (2010), focusing on examples which result in large estimated model changes overcomes drawbacks of outliers (third row in Fig. 7.24). It should be noted that this only holds for classification models which are sufficiently complex. A counter-example would be a simple least-square fit of a linear regression model, where outliers are over-emphasized due to the quadratic loss. For GP regression, we can instead observe that new clusters are hardly discovered. Instead, samples near to the current decision boundary and close to known samples are preferred.

A better focus on new clusters is visible for the estimated risk minimization criterion (fourth row in Fig. 7.24) and estimated model output changes (fifth row in Fig. 7.24). Note that both techniques come with increased computational costs compared to remaining criteria (see also Chapter 5 for details). Nonetheless, all relevant clusters are known after only three iterations. Subsequent queries are then used for boundary refinement. It should be further noted that the selected examples are almost identical to the oracle gold standard with access to ground truth labels (last row in Fig. 7.24).

Since both techniques are resistant to outliers, can discover new clusters, and can improve current decision boundaries, they represent our methods of choice if available computation resources are sufficient.

Summary: *In summary, we can intuitively observe advantages and limitations of different active learning criteria by visually inspecting them in a controlled 2D setup. All findings are summarized in Table 7.7.*

7.3.1.2 Quantitative Comparisons on ImageNet

After the qualitative analysis in the previous section, we present quantitative analyses to compare active learning criteria in more realistic scenarios. We start with binary classification tasks derived from ImageNet (see Section 7.1.1.1).

Table 7.7: *Summary of visually analyzing active learning criteria on 2D data. We marked properties on which each criterion puts a visible focus.*

Criterion	Improves decision boundary	Resistant to outliers	Discovers new clusters
Random			
$\mathcal{Q}_{\text{GP-}\sigma^2}$ (Eq. (5.14))			(✓)
$\mathcal{Q}_{\text{GP-unc}}$ (Eq. (5.15))			(✓)
$\mathcal{Q}_{\text{GP-}\Delta\theta}$ (Eq. (5.22))	✓	(✓)	
$\mathcal{Q}_{\text{GP-ERM}}$ (Eq. (5.12))	✓	✓	✓
$\mathcal{Q}_{\text{GP-}\Delta f}$ (Eq. (5.44))	✓	✓	✓

Experimental Setup: We obtain binary classification task by following the experimental setup in Section 7.2.1. Thus, we create one-vs-all tasks from the 1,000 available ImageNet categories. In detail, we select 100 sets of 10 randomly chosen classes. The first selected category of each set serves as positive category which needs to be separated from remaining 9 categories.

From the official ImageNet training set, we randomly select one example of each category for training. All remaining examples build the unlabeled pool with a resulting size of 990 samples. Model evaluation is done on the official validation set with 50 examples of each category. In this manner, we create ten different splits into labeled and unlabeled for each set of ten classes. Consequently, we conduct 1,000 individual one-vs-all experiments. We evaluate the models' accuracy after every query using the area under an ROC curve (AUC) as introduced in Section 7.1.2.2.

As in Section 7.2.1, we represent images using the publicly available BoW histograms. The resulting 1,000 dimensional histograms are additionally L_1 -normalized. We again apply a histogram intersection kernel κ_{HIK} following our previous evaluations in Section 7.2.

The noise level for model regularization is fixed to $\sigma_n^2 = 0.1$. We also tested optimizing σ_n^2 by maximizing the marginal likelihood as introduced in Section 2.1.5.7 and applied in Section 7.2.3. In detail, we tested fixed noise levels after an initial optimization as well as constantly optimizing the noise level after five or ten queries but found noticeable impact on classification accuracies.

As query strategies, we follow the previous evaluation and compare criteria for Gaussian process models. In detail, we apply $\mathcal{Q}_{\text{GP-}\sigma^2}$ (see Eq. (5.14) in Section 5.3.1), $\mathcal{Q}_{\text{GP-unc}}$ (see Eq. (5.15) in Section 5.3.1), and $\mathcal{Q}_{\text{GP-ERM}}$ (see Eq. (5.12) in Section 5.2.4). These baseline are compared against our introduced techniques $\mathcal{Q}_{\text{GP-}\Delta\theta}$ (see Eq. (5.22) in Section 5.4.2) and $\mathcal{Q}_{\text{GP-}\Delta f}$ (see Eq. (5.44) in Section 5.5.2). Finally, we investigate the two approximations of EMOC $\mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}$ and $\mathcal{Q}_{\text{GP-}\Delta\theta\text{-pde}}$

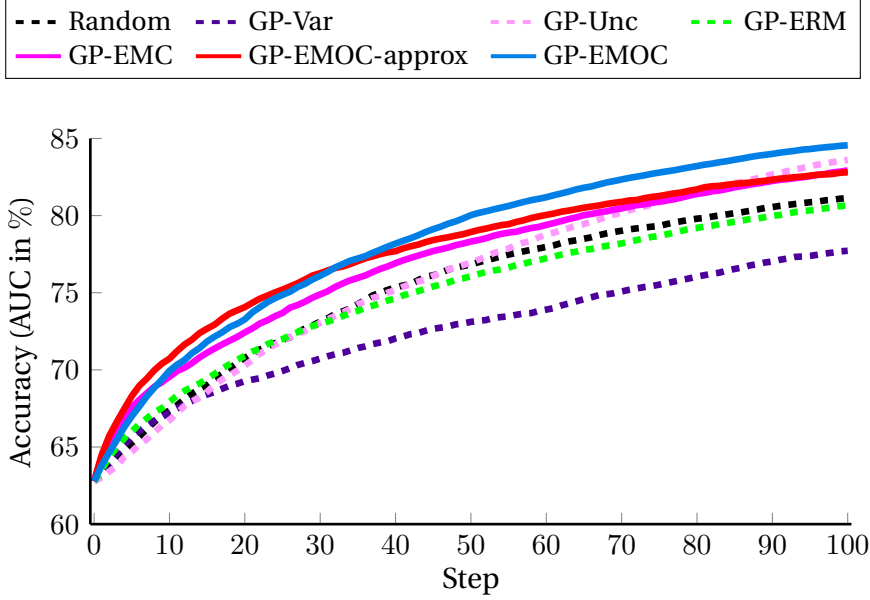


Figure 7.25: Evaluating active learning criteria in binary classification tasks derived from ImageNet. Results are averaged over 100 different class splits with 10 random initializations per split. Best viewed in color.

(see Eq. (5.47) and Eq. (5.54) in Section 5.5.3).

Results for all labeling iterations are shown in Fig. 7.25. In addition, we present obtained accuracies after 100 queries in Table 7.8. Statistical significance of differences between $\mathcal{Q}_{\text{GP-}\Delta f}$ and remaining techniques is estimated according to the Wilcoxon signed rank test (see Section 7.1.2.3).

Experimental Results: First of all, we observe that pure exploration is worse than random selection. This again indicates the preference of $\mathcal{Q}_{\text{GP-}\sigma^2}$ for outliers which is consistent with our results from the previous evaluation. After 100 queries, remaining techniques are almost on par, although $\mathcal{Q}_{\text{GP-unc}}$ is inferior in early stages of learning. It is noteworthy that $\mathcal{Q}_{\text{GP-}\Delta f}$ leads to the highest accuracy with small but significant improvements over remaining techniques.

In contrast to the results on 2D data, also the expected risk minimization strategy leads to results inferior to passive learning. We believe that this results from complex data distributions in high-dimensional feature spaces. Consequently, initial class estimates are likely error prone which leads to unstable estimates of empirical risk reduction. Loosely spoken, aiming for “decisions turning for the better” ($\mathcal{Q}_{\text{GP-ERM}}$) is likely too ambitious, since “the better” can be wrongly estimated too easily, especially with only few labeled data available.

Table 7.8: Quantitative results from experiments shown in Section 7.3.1.2 and Section 7.3.1.3 regarding active learning in binary classification scenarios. Results are reported as AUC scores (in %) after 100 queries and averaged over 10 random initializations of 100 random splits. Additionally shown are confidence intervals for estimating statistical significance of differences to $\mathcal{Q}_{\text{GP-}\Delta f}$ computed with a Wilcoxon signed rank test (see Section 7.1.2.3).

Strategy	ImageNet	Caltech-256
Random	81.14 ($10^{-74.7}$)	85.76 ($10^{-101.6}$)
$\mathcal{Q}_{\text{GP-}\sigma^2}$	77.72 ($10^{-107.7}$)	82.18 ($10^{-121.5}$)
$\mathcal{Q}_{\text{GP-unc}}$	83.60 ($10^{-0.7}$)	89.25 ($10^{-0.2}$)
$\mathcal{Q}_{\text{GP-ERM}}$	80.67 ($10^{-22.0}$)	88.24 ($10^{-6.4}$)
$\mathcal{Q}_{\text{GP-}\Delta\theta}$	82.93 ($10^{-23.9}$)	89.04 ($10^{-5.8}$)
$\mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}$	82.81 ($10^{-35.4}$)	88.85 ($10^{-13.6}$)
$\mathcal{Q}_{\text{GP-}\Delta f}$	84.56	89.79

Summary: In summary, we found that samples minimizing the estimated empirical risk or with large predictive variances are not feasible for active learning from real-world data. Remaining techniques lead to comparable results.

7.3.1.3 Quantitative Comparisons on Caltech-256

To verify previous findings, we repeat the same investigation on the Caltech-256 dataset.

Experimental Setup: In a first experiment, we rebuild the previous analysis on ImageNet with identical settings. Thus, we compare the active learning criteria as described previously. Number of class splits and random initialization is kept identical. The only difference is the procedure for generating splits. Since no official test split is available, we randomly select from each category one example for training, 30 hold-out examples for testing, and keep remaining examples as unlabeled data.

Images are represented as BoW histograms over densely extracted SIFT features. We use the publicly available features by Tuytelaars et al. (2010)⁴. Thereby, we again obtain L_1 -normalized histogram features of dimension $D = 1,000$. Results are shown in Fig. 7.26 and Table 7.9.

In addition, we follow the setup of Kapoor et al. (2010) and create a second one-vs-all experiment with increased amounts of negative data. To this end, we use every category which has already been present in the predecessor Caltech-101

⁴http://homes.esat.kuleuven.be/~tuytelaa/unsup_features.html

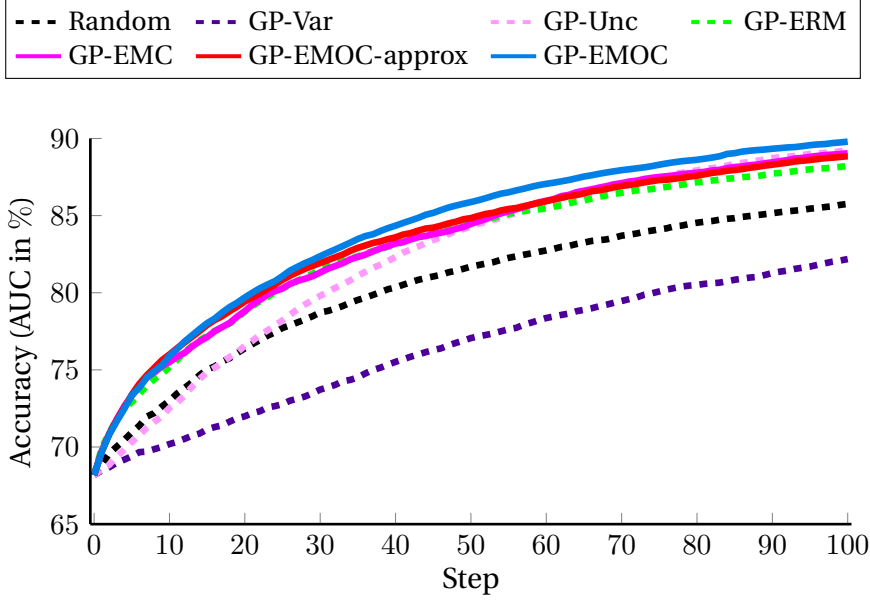


Figure 7.26: Evaluating active learning criteria in binary classification tasks derived from Caltech-256. Results are averaged over 100 different class splits with 10 random initializations per split. Best viewed in color.

to serve once as positive class. The corresponding class IDs of Caltech-256 are [15, 20, 22, 36, 52, 57, 63, 64, 66, 91, 100, 102, 114, 121, 123, 127, 129, 134, 140, 145, 172, 179, 201, 204, 230, 235, 240, 251, 252, 253] which results in 30 one-vs-all settings. For every task, we use all remaining 255 categories as negative data.

We start with a single example of every category for training, *i.e.*, one positive and 255 negative examples. Model evaluation is done on 10 hold-out examples of every category. All remaining examples are accessible as unlabeled pool. Selection of labeled, unlabeled, and test data is repeated 10 times to average results. We fix the labeling budget to 20 iterations. As criterion for active query selection, we apply the EMOC criterion $\mathcal{Q}_{\text{GP-}\Delta f}$ which lead to best results in the previous quantitative analysis.

The improvement of $\mathcal{Q}_{\text{GP-}\Delta f}$ over random selection is shown in Fig. 7.27 for individual categories. Improvement is measured as difference in area under AUC curves.

Experimental Results: Regarding the results in Fig. 7.26 with an identical setup as on ImageNet, we find similar trends. Pure exploration is clearly unfeasible even after 100 queries. Similarly, $\mathcal{Q}_{\text{GP-unc}}$ is not able to improve over random selection in early stages of learning, but is on par with remaining techniques for larger query budgets. Interestingly, $\mathcal{Q}_{\text{GP-ERM}}$ leads to notable accuracy gains over random selec-

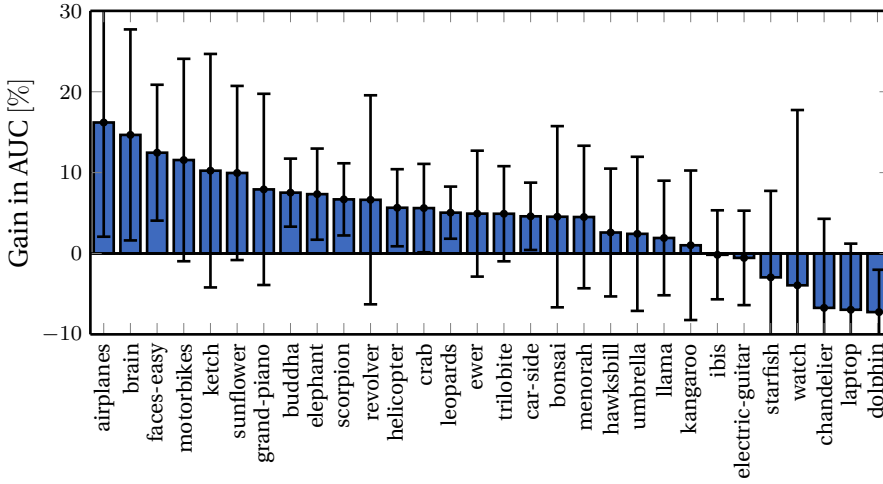


Figure 7.27: Evaluating active learning with $Q_{GP-\Delta f}$ in one-vs-all binary classification tasks derived from Caltech-256 as done by Kapoor et al. (2010). Gain in AUC is measured as improvement over models trained with random selection of queries. Results are averaged over 10 random initializations per task.

tion. Currently, we have no explanation why results of Q_{GP-ERM} differ so strongly in our experiments on ImageNet and Caltech-256. As previously, $Q_{GP-\Delta f}$ outperforms remaining approaches by small but significant margins ($p < 10^{-3}$, paired-sample t-test, see Section 7.1.2.3).

With respect to the one-vs-all evaluation, we observe that almost all binary tasks benefit from active query selection with our introduced $Q_{GP-\Delta f}$ criterion.

Summary: *In summary, we can verify previous findings for active learning in binary classification experiments. An exception is Q_{GP-ERM} which leads to improved results. Best classification performance is again obtained using criteria based on estimated model output changes.*

7.3.1.4 Qualitative Analysis of Selected Examples

The previous analysis was purely quantitative. Now, we visually analyze chosen examples for intuitively informative patterns.

Experimental Setup: We randomly select a single run of the previous one-vs-all experiment on Caltech-256. In detail, we visually analyze the first 20 selected queries of one run using Caltech-256’s airplane category as positive data. Note that the setup is kept identical to the previous experiment. The chosen examples along with their ground truth category are shown in the top part of Fig. 7.28. Furthermore,

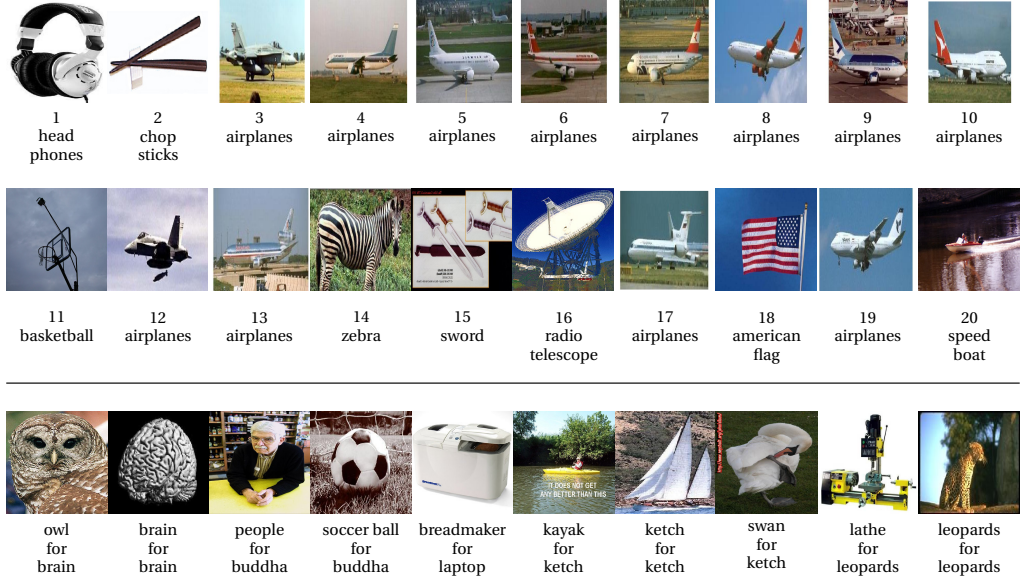


Figure 7.28: Visual analysis of selected queries in one-vs-all experiments on Caltech-256. Top: the first 20 queries of $Q_{GP-\Delta f-\text{approx}}$ along with their ground truth class label. The positive category is airplane. Bottom: Some interesting queries of $Q_{GP-\Delta f-\text{approx}}$ for several one-vs-all tasks. The notation “class A for class B” denotes that class A is the ground truth label of the query and class B the one-vs-all task. Images are resized to quadratic aspect ratios for visualization purpose.

we overlaid the chosen examples with the corresponding classification accuracy in Fig. 7.29.

In addition, we manually selected several interesting queries made in the remaining one-vs-all experiments. These queries along with their ground truth class are shown in the bottom part of Fig. 7.28. Regarding the notation, “class A for class B” denotes that “class A” is the ground truth category of the queried image, whereas “class B” is the positive class of the one-vs-all experiment.

Experimental Results: When inspecting the chosen examples in Fig. 7.28, we observe that airplane examples and non-airplane examples are chosen equally frequent. We thus conclude that $Q_{GP-\Delta f-\text{approx}}$ provides a suitable trade-off between exploration and exploitation. Furthermore, examples of the positive category are preferred in early stages. Given the highly imbalanced initial class distribution of 1 : 255, this preference is intuitively useful to adequately model the positive category. Subsequent queries are from visually similar categories to resolve class confusions and to refine class boundaries.

Queries along with corresponding classification accuracies in Fig. 7.29 confirm this intuition. The initial queries lead to largest gains in classification performance.

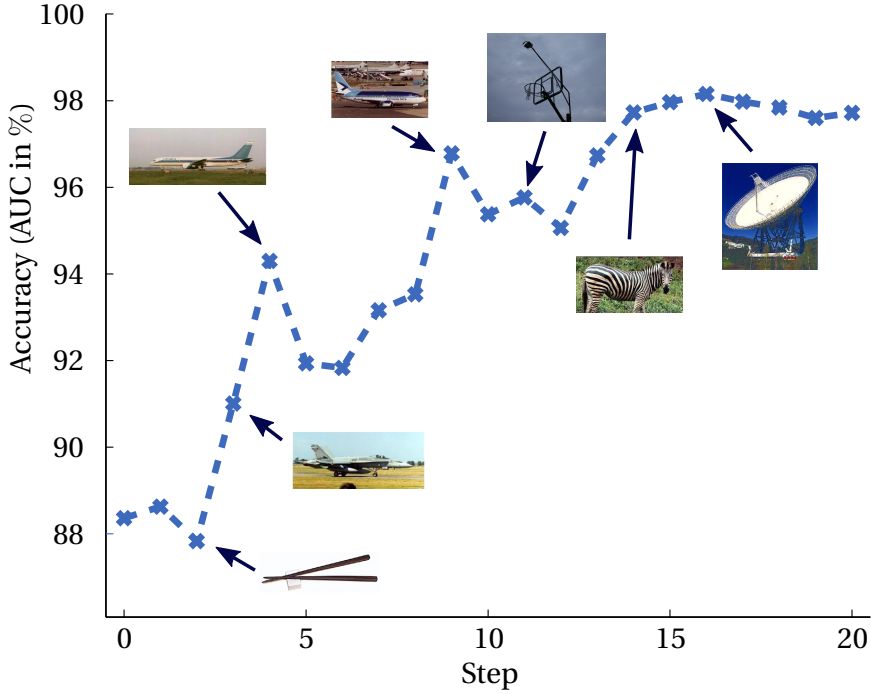


Figure 7.29: Qualitative analysis of a single one-vs-all experiments on Caltech-256. The positive class is airplane, and all queried examples have already been shown in Fig. 7.28. We displayed queries which lead to noticeable gains of the airplane detector’s classification performance.

This is caused by the increased access to representative data for the positive category. After ten queries, the accuracy increased by $\sim 8\%$. The remaining gains are rather incremental but important for further boosting discrimination abilities. Besides these insights, we find no intuitive explanation why some examples lead to larger improvements than others by just inspecting them visually.

The selection of likely mixed-up categories is further shown in the bottom part of Fig. 7.28. For several tasks, we observe that $\mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}$ queries examples which are useful to differentiate between visually similar categories.

Summary: In summary, examples selected by $\mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}$ are intuitively useful upon visual inspection. Queries are selected to improve the representation of positive data and to improve discrimination abilities with visually similar negative data.

7.3.1.5 EMOC with Perfect Label Estimates

In previous results, we found that EMOC-based criteria are highly suited for active sample selection in binary classification tasks. One crucial part in EMOC compu-

tations is the expectation operator regarding possible class labels of an unlabeled example (see Eq. (5.44) in Section 5.5.2). For every possible label assumption, the model is updated and output changes are computed. Finally, scores are computed as weighted average using the probability of each class label. We now investigate the resulting accuracy gain if label estimates would be perfect but still applied within the EMOC framework.

Experimental Setup: We apply the identical setup as for the evaluations on ImageNet in Section 7.3.1.2. Thus, we average results over 100 randomly selected tasks with 10 random initializations per task. Each task consists of a single positive category as well as nine negative classes. We again select one example of each category for training from the official training set, and use remaining data for active selection. All examples contained in the official validation set of the corresponding categories are used for model evaluation. As previously, the labeling budget is fixed to 100 iterations.

For the exact EMOC criterion $\mathcal{Q}_{\text{GP-}\Delta f}$ (see Eq. (5.44)), we estimate class labels as previously. In addition, we replace class estimates with ground truth class labels. Thus, instead of a weighted average over possible labels, only output changes induced by the correct label are computed. This reflects the upper bound of gainable information using EMOC criteria for active sample selection.

In addition, we also provide ground truth labels to the estimated risk minimization strategy shown in Eq. (5.12). Since we evaluate models using area under ROC curve values, this results in greedily selecting examples which maximally increase AUC scores (corresponds to $\mathcal{Q}_{\text{GP-AUC-Oracle}}$ in the visual analysis of the 2D problem shown in Section 7.3.1). Note that this is the upper bound for any myopic active learning criterion. Besides, this artificial setup can also be seen as data selection problem. Results are shown in Fig. 7.30.

Experimental Results: Regarding EMOC, we see that relying on correct label estimates further improves the quality of selected queries and resulting learning rates. In practice, however, improved class estimates for active learning resembles a chicken-and-egg problem, since active learning is used to rapidly obtain better class estimates. Nonetheless, an improved estimation of class labels especially in early stages of learning is beneficial and should be aimed at.

Notably, learning curves can be further improved by greedily maximizing the gain in AUC computed with ground truth class labels. After only ~ 15 query steps, learning curves are saturated and additional information is hardly added. This interesting result shows the full potential of active learning. Furthermore, the results show once again that ERM is theoretically a great strategy. Nonetheless, it suffers from practical limitations which makes it inferior to EMOC criteria in

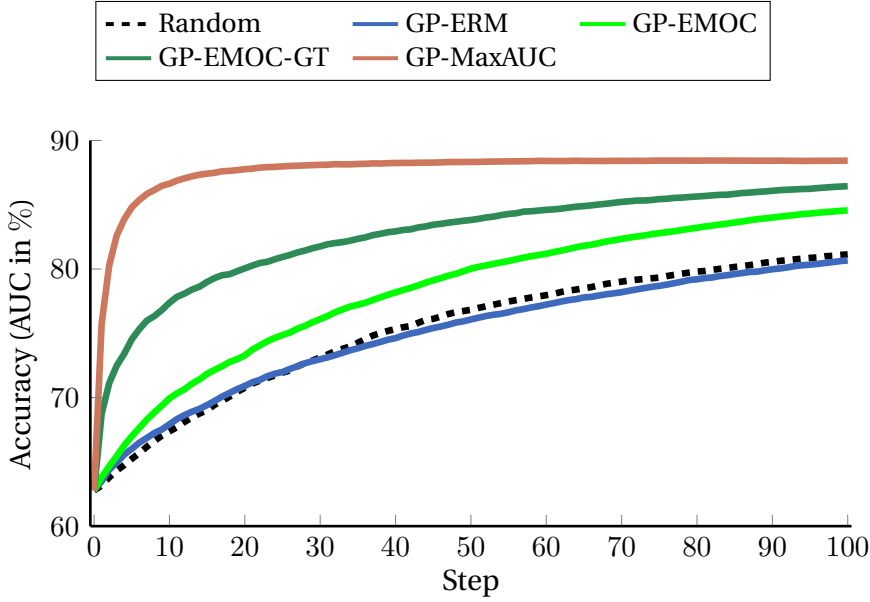


Figure 7.30: Evaluating benefits of accessible ground truth labels when computing active learning scores. Experiments are conducted on ImageNet with an identical setup as in previous evaluations. Results are averaged over 100 tasks with 10 random initializations. Best viewed in color.

realistic applications.

Summary: We conclude that EMOC can benefit from improved label estimates.

7.3.1.6 Importance of Density-based Re-weighting

As mentioned in Section 5.5.3, density-based re-weighting can be applied to arbitrary active learning criteria to focus on dense regions in feature space. In the following, we quantitatively evaluate the resulting benefits.

Experimental Setup: We again apply the identical setup as done for the quantitative analyses on ImageNet in Section 7.3.1.2. As criteria for active learning, we apply $\mathcal{Q}_{\text{GP-}\sigma^2}$ (see Eq. (5.14) in Section 5.3.1), $\mathcal{Q}_{\text{GP-unc}}$ (see Eq. (5.15) in Section 5.3.1), and $\mathcal{Q}_{\text{GP-}\Delta\theta}$ (see Eq. (5.22) in Section 5.4.2). We exclude $\mathcal{Q}_{\text{GP-ERM}}$ and $\mathcal{Q}_{\text{GP-}\Delta f}$ since both techniques already contain an expectation operator with respect to input data (see Eq. (5.12) and Eq. (5.22), respectively). Scores of every criterion are multiplied with Parzen density estimates of the corresponding unlabeled example as shown in Eq. (5.56). Note that for $\mathcal{Q}_{\text{GP-}\Delta\theta}$, re-weighting leads to $\mathcal{Q}_{\text{GP-}\Delta\theta\text{-pde}}$ as motivated earlier.

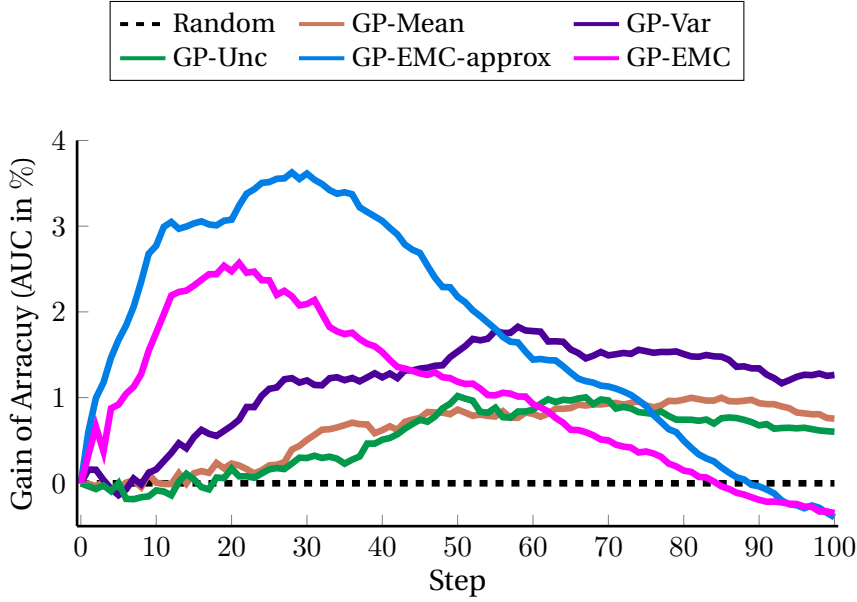


Figure 7.31: Quantitative benefits of density-based re-weighting for active learning criteria on ImageNet. The y-axis represents the gain in AUC resulting from multiplying active learning scores with density estimates as shown in Eq. (5.56). Results are averaged over 100 tasks with 10 random initializations.

For each criterion, we report the accuracy gain over the corresponding counterparts without density-based re-weighting. Results are shown in Fig. 7.31.

Experimental Results: We observe that the incorporation of data density into active sample selection is beneficial in almost all cases. This confirms the findings reported by Settles (2008). Surprisingly, the benefit for the explorative technique $\mathcal{Q}_{\text{GP-}\sigma^2}$ is less prominent as expected. Besides, the estimated model change criterion $\mathcal{Q}_{\text{GP-}\Delta\theta}$ strongly benefits in early phases of learning, but no additional advantage can be drawn in later stages. The causes for these phenomena are not yet understood. A solution could exist in adaptive weighting schemes, *e.g.*, by applying adaptive combination techniques as proposed by Ebert et al. (2012).

Summary: We conclude that the incorporation of data density into active sample selection is beneficial.

7.3.2 Experiments in Multi-class Scenarios with Unnameable Instances

In contrast to previous experiments conducted for binary classification, we now investigate active learning in multi-class scenarios as introduced in Section 5.6. Thus, we are interested to see whether the findings presented in previous experiments are still valid. Furthermore, we focus on scenarios with unnameable instances as motivated in Section 5.6.3 (see also Fig. 5.7). We evaluate our proposed active learning techniques in three multi-class classification scenarios with unnameable instances to answer the following questions:

- (i) Section 7.3.2.1: Can we confirm results of binary scenarios in simple multi-class settings?
- (ii) Section 7.3.2.2: Is active learning applicable to face identification tasks?
- (iii) Section 7.3.2.3: Can we successfully perform active learning on object proposals?
- (iv) Section 7.3.2.4: Do we obtain similar conclusions if the selection time matters?
- (v) Section 7.3.2.5: Are queried examples visually striking?
- (vi) Section 7.3.2.6: Which loss functions are sensible to compute EMOC?
- (vii) Section 7.3.2.7: How should we estimate probabilities when computing EMOC scores?

7.3.2.1 Proof-of-Concept on USPS Handwritten Digits

We start our evaluations with the task of recognizing handwritten digits. Therefore, we use the USPS digit dataset as introduced in Section 7.1.1.4.

Experimental Setup – Data: We use the provided disjoint datasets for training and testing, respectively. A single setup is obtained by randomly selecting two digit categories to be initially known. For training, we randomly select 5 examples of each category from the official training dataset. All remaining examples serve as unlabeled data. From the official test set, we chose 30 hold-out examples of each category for model testing. In consequence, the test set consists of categories which are unknown to the initially trained model. Thus, active learning as well as discovery of unknown categories is required. As in previous experiments, each setup is repeated with ten random initializations. We finally average results over ten setups with their ten respective random initialization. Thus, we obtain conclusions

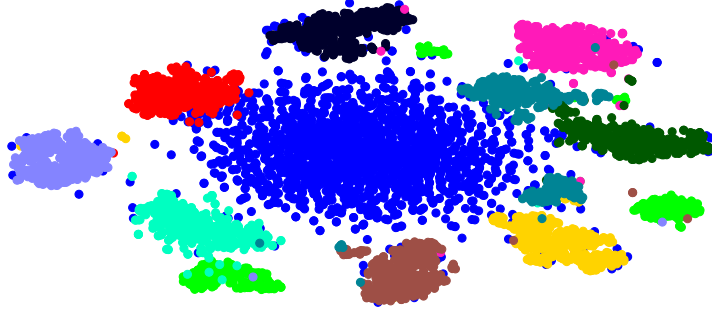


Figure 7.32: Visualization of USPS handwritten digits by Bache and Lichman (2013) with unnameable instances. Examples are non-linearly projected to $D = 2$ using the unsupervised t-SNE method by Van der Maaten and Hinton (2008). After projection, each data point is colored according to its category label. Unnameable instances are indicated in dark blue.

from 100 experiments. Model accuracy is measured as average recognition rate (ARR) with respect to all available categories.

To additionally investigate the effect of unnameable instances, we augment the unlabeled pool with digits where all pixels are randomly re-arranged. Since resulting images are hardly recognizable, this leads to non-categorical examples as motivated in Section 5.6.3. Thereby, we increase the size of the unlabeled pool $\mathcal{U}$ by $\sim 50\%$ by randomly re-arranging pixels of every second example from the USPS dataset. The resulting dataset is shown in Fig. 7.32 after projecting it into two dimensions using the unsupervised t-SNE method by Van der Maaten and Hinton (2008).

Experimental Setup – Query functions: As query functions, we compare our three multi-class variants of EMOC as introduced in Section 5.6.2 and Section 5.6.3. In detail, we denote with GP-EMOC_{MC} the multi-class variant of our EMOC principle ($\mathcal{Q}_{\text{GP-}\Delta f\text{-MC}}$ as introduced in Section 5.6.2). We apply an $\mathcal{L}_{|\cdot|}$ -loss (see Eq. (5.60)) and Monte-Carlo probability estimates (see Eq. (5.62)). Furthermore, GP-EMOC_{PDE} denotes the additional incorporation of the data density ($\mathcal{Q}_{\text{GP-}\Delta f\text{-MC-PDE}}$ in Eq. (5.64)). Finally, GP-EMOC_{PDE+R} additionally accounts for possible rejections of unnameable instances ($\mathcal{Q}_{\text{GP-}\Delta f\text{-MC-PDE-R}}$ derived in Eq. (5.65)).

We compare our EMOC criteria against a set of established techniques. For Gaussian process models, we follow Kapoor et al. (2010) and select examples with large predictive variances (GP-var) or large uncertainties (GP-unc) – see Section 5.6.1 for details. In addition, we transfer the best-vs-second-best strategy proposed by Joshi et al. (2009) to GP models (GP-1-vs-2). Furthermore, we apply the estimated empirical risk minimization (GP-ERM) approach of Roy and McCallum (2001) to GP models as introduced in Eq. (5.12). Finally, we also include the baseline of random

query selection (Random). An RBF kernel serves as covariance function for all criteria which rely on GP models.

To also compare against other classification techniques, we apply a probabilistic KNN classifier with the query strategy PKNN as proposed by Jain and Kapoor (2009). Source code of their method is publicly available⁵. As last competitor, we apply the Dirichlet process expected accuracy (DPEA) technique by Hospedales et al. (2012) with publicly available source code⁶.

All selected examples which were successfully labeled are immediately used to update the current classification model. If a strategy selected an unnameable instance which is rejected from labeling, we follow Jia et al. (2013) by adding it as additional negative example for all one-vs-all models of known categories.

Results are shown in Fig. 7.33. We further show the upper bound of classification accuracy obtained from training one-vs-all GP models from all training examples of the training set (denoted as dashed pink line). Finally, we present an overview of classification accuracies after 100 queries in Table 7.9.

Experimental Results: First of all, we find that all EMOC variants are able to discover all unknown categories after only 20 queries. In direct consequence, classification accuracy rises from 20% to $\sim 85\%$ after only 20 queries which serves as solid proof-of-concept. Interestingly, we do not observe clear differences between all three methods, which is likely due to the simplicity of the dataset (see also the t-SNE visualization in Fig. 7.32).

From the remaining techniques, the DPEA method is slightly inferior to EMOC and needs ~ 40 queries to discover all classes. This result is especially interesting, since the existence of potentially new categories is explicitly incorporated into DPEA. In direct comparison, EMOC reduces the required query budget by roughly 50% without the necessity of explicitly modeling new categories.

Surprisingly, all remaining techniques perform worse than random query selection. This observation holds both for the frequency of discovered categories as well as for the resulting classification accuracy. We attribute this behavior to the missing aspect of discovering new categories (in contrast to DPEA). Furthermore, the purely explorative method GP-Var is not able to gather any information at all. The reason is the overwhelming amount of unnameable instances which are rejected on annotation request. This observation confirms our previous evaluations regarding the distractedness of explorative methods by outliers.

⁵Source code for PKNN obtained from <http://research.microsoft.com/en-us/um/people/akapoor/cvpr2009/>

⁶Source code for DPEA obtained from <http://www.eecs.qmul.ac.uk/~tmh/>

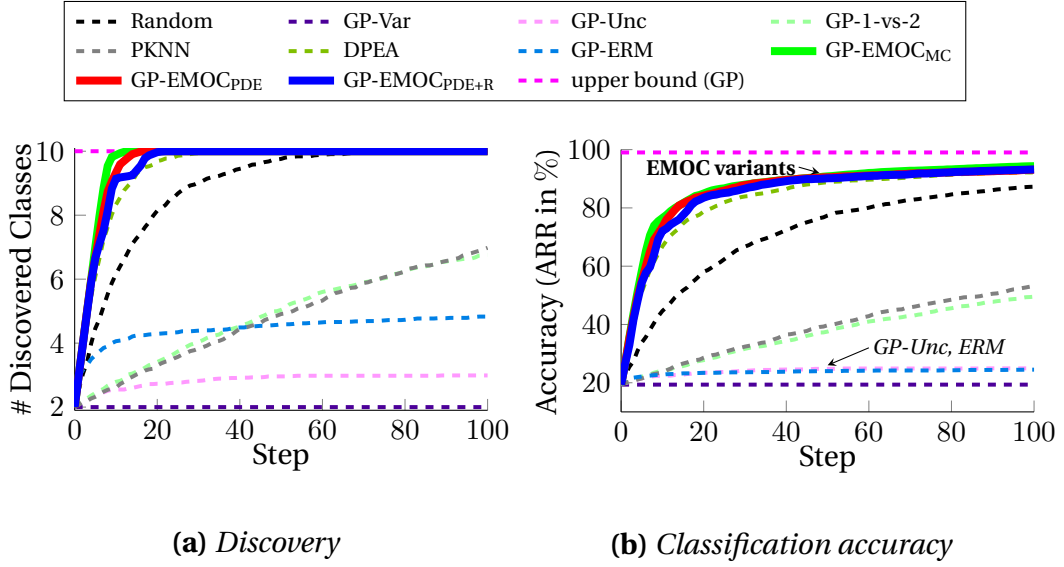


Figure 7.33: Results of active learning and class discovery for multi-class digit recognition using the USPS dataset by Bache and Lichman (2013). Each run consists of 100 queries (x-axis). Results are averaged over 10 random initializations. Left: number of known categories after each iteration. Right: average classification accuracy after each iteration evaluated on hold-out test data. Overlapping learning curves of multiple strategies are additionally labeled. Best viewed in color.

Visualization of Query Selection: Similar to the analysis in Section 7.3.1.1, we further visualized the query process to visually investigate the preferences of every method. This is possible by projecting all data into $D = 2$ using t-SNE as done in Fig. 7.32. Labeled training examples are indicated with larger diameters. Known categories are assigned the same unique colors as for the visualization of the entire dataset in Fig. 7.32.

Every unlabeled point is color-coded according to the most likely class estimate at each time step. The color saturation is chosen according to the classification probability such that saturated colors indicate clear decisions. The resulting video is available at <https://youtu.be/AEIrYqMHH74>. A snapshot is shown in Fig. 7.34.

Given the video, we can again see the benefit of our introduced EMOC method which rapidly discovers new clusters. In addition, non-categorical noise images in the middle of the data distribution are ignored by EMOC which leads to a non-saturated grayish color coding. Finally, we observe that EMOC is able to select representative instances of each cluster due to the marginalization over available data. In consequence, few queried examples are sufficient to model all categories adequately. The visual analysis thereby confirms our previous quantitative observations. We refer to the YouTube video for the investigation of remaining techniques.

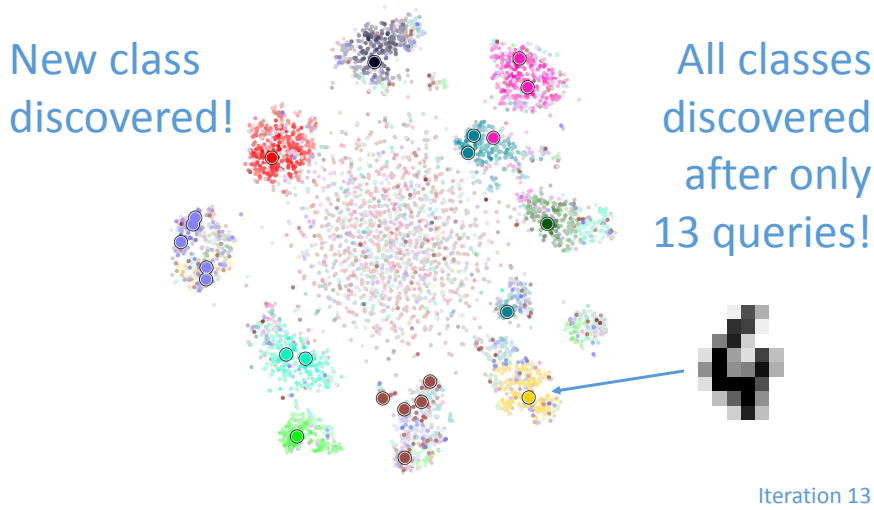


Figure 7.34: A snapshot of the video to visualize the process of active query selection in the presence of unnameable instances. The entire video is available on YouTube at <https://youtu.be/AEIrYqMHH74>.

Summary: In summary, our EMOC criterion is well suited for multi-class active learning and discovery of simple handwritten digits even without modeling rejections of unnameable instances.

7.3.2.2 Face Identification

As second benchmark, we evaluate multi-class active learning and discovery for the challenging task of face identification. In these scenarios, it is likely that unnameable instances are requested. A typical reason is the false detection of non-face-regions as faces which are rejected by any annotator as non-categorical. Furthermore, an annotator could be interested in labeling only a subset of collected person photos (e.g., only his friends, and not all possible faces in his holiday snapshots). Finally, human annotators are likely not omniscient oracles and thus need to reject annotation of a requested image if the contained person is unknown. Thus, modeling possible rejections is intuitively useful.

Experimental Setup: We use the “Labeled faces in the Wild” dataset introduced in Section 7.1.1.5 as benchmark dataset. Images are represented using the face features provided by Guillaumin et al. (2009). This yields 27 SIFT descriptors which are extracted from nine landmarks and three different scales. Descriptors are finally concatenated and L_1 -normalized. Thereby, the resulting face representations are histograms, which justifies a histogram intersection kernel as covariance function.

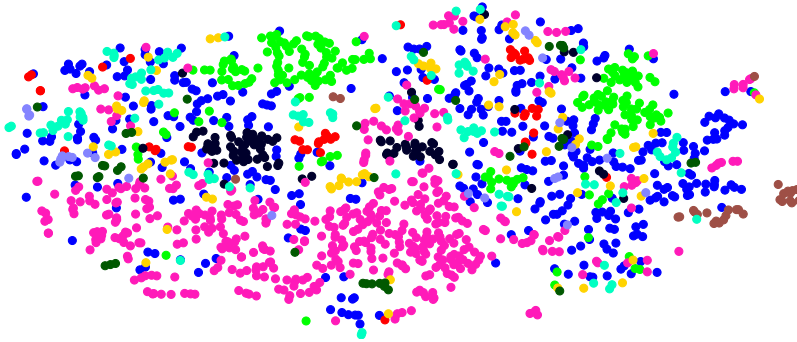


Figure 7.35: Visualization of human faces from the Labeled Faces in the Wild (LFW) dataset by Huang et al. (2007). Examples are non-linearly projected to $D = 2$ using the unsupervised t -SNE method by Van der Maaten and Hinton (2008). After projection, each data point is colored according to its category label. Ten persons are assumed to be known, whereas all other persons serve as unknown and are color-coded in dark blue. Unnameable instances are indicated in dark blue.

From the entire dataset, we use the nine categories which contain at least 55 examples as domain of interest. In each task, we randomly select two of the nine categories to be initially known. As previously, five randomly selected instances for each of the two categories serve for initial model training. For model evaluation, we select 30 hold-out examples for each of the 9 categories. All remaining instances serve as unlabeled data. In addition, we randomly select 400 individuals with only a single image to serve as unnameable instances. Thus, any image of these persons will be rejected during annotation. The resulting dataset is visualized in Fig. 7.35 after being projected into $D = 2$ using t -SNE.

The remaining setup is identical to evaluations in Section 7.3.2.1 on USPS. Thus, results are averaged over ten random class selections with their ten respective random initializations of training examples. Techniques for active learning and discovery are kept identical. Results are shown in Fig. 7.36. As previously, we show the upper bound of classification accuracy obtained by learning one-vs-all GP models from all training examples of the training set (denoted as dashed pink line). In addition, classification accuracies after 100 queries are shown in Table 7.9.

Experimental Results: Let us start analyzing the ability of discovering new categories over time. We observe that the difference between random selection and the remaining techniques is less clearly than on USPS (see previous evaluation in Section 7.3.2.1). Nonetheless, relying only on the predictive GP distribution (GP-Var or GP-Unc) leads to results slightly inferior to passive learning. This also holds for the computationally expensive expected risk minimization strategy (GP-ERM). All remaining techniques are roughly on par and lead to small but noticeable im-

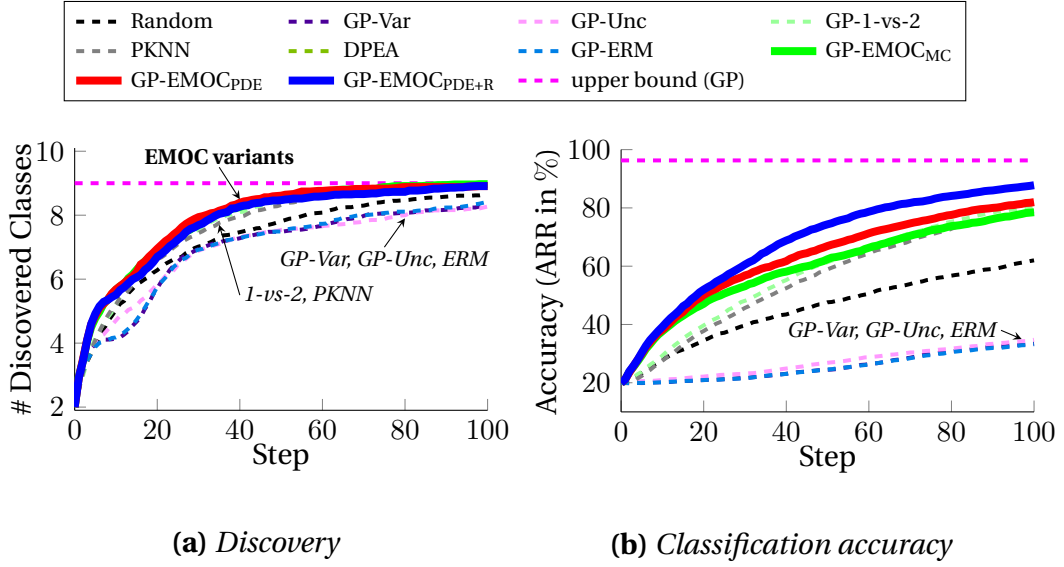


Figure 7.36: Results of active learning and class discovery for multi-class face identification using the LFW dataset by Huang et al. (2007). Each run consists of 100 queries (x-axis). Results are averaged over 10 random initializations. Left: number of known categories after each iteration. Right: average classification accuracy after each iteration evaluated on hold-out test data. Overlapping learning curves of multiple strategies are additionally labeled. Best viewed in color.

improvements over random selection. Note further that the provided source code for DPEA, which was the closest competitor of EMOC on USPS, is not applicable in this scenario due to the increased number of examples and feature dimensions.

We finally note that no method is able to discover all categories after 50 queries. Even after 100 iterations, only GP-EMOC_{MC} was able to discover all categories in all experiments. This observation is likely caused by the increased difficulty of the dataset. As shown in Fig. 7.35, data points are severely cluttered in space and less clustered compared with USPS which hinders the discovery of new categories.

Regarding improvements of classification accuracy, we observe similar trends as in the experiments on USPS. Again, GP-var, GP-unc, and GP-ERM query a significant amount of unnameable examples which are rejected. In consequence, the resulting classification accuracy improves only slightly over time ($\sim 15\%$ after 100 iterations).

All remaining techniques significantly outperform passive learning. In contrast to the previous experiment, we now observe remarkable accuracy gains of GP-EMOC_{PDE+R} compared to the simple GP-EMOC_{MC} baseline (87.8% versus 78.5%). Thus, explicitly modeling unnameable instances pays off in this scenario. PKNN and GP-1-vs-2 are inferior in early stages of learning but lead to accuracies compa-

nable with GP-EMOC_{MC} after all 100 iterations.

Summary: *We conclude that modeling unnameable instances is not required for person discovery in face identification scenarios. However, it substantially helps to query informative examples important for accurate person identification.*

7.3.2.3 Active Discovery with Object Proposals

We finally investigate a multi-class classification scenario where categorical and non-categorical unnameable examples occur frequently. In detail, we investigate the task of learning animal classifiers from automatically extracted sub-regions in images. Since the process of generating candidate regions is far from perfect, a large number of possible regions will cover non-animal objects or even no objects at all. Since these scenarios are common in automated biodiversity analysis tasks, we evaluate the applicability of the presented active learning techniques in this context.

Experimental Setup: As benchmark, we use the MS-COCO dataset by Lin et al. (2014) as introduced in Section 7.1.1.6. Since we focus on simulating biodiversity applications, we restrict the domain of acceptable categories to the ten animal classes contained in MS-COCO: bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, and giraffe. Candidate regions are extracted using geodesic object proposals by Krähenbühl and Koltun (2014) which is an unsupervised object proposal technique.

From all candidate regions extracted from the MS-COCO training set, we accept those which have an intersection-over-union value of at least 0.9 with ground truth regions of any of the ten categories. The surrounding bounding box finally determines the sub-image for each example. From all remaining segments, we additionally select 10,000 to build the set of unnameable instances. Thereby, we cover object proposals from non-animal categories (categorical rejections) as well as background segments (non-categorical rejections). Exemplar regions for the three types of instances are shown in Fig. 7.37

We represent images by computing CNN activations on the bounding box of each segment. In detail, we use pool5 features from the AlexNet model by Krizhevsky et al. (2012). Feature extraction is done using the publicly available Caffe framework by Jia et al. (2014). Since these activations are high-dimensional ($D = 9,216$), we apply a linear kernel as covariance function for GP models. All remaining settings are kept identical to the previous evaluations in Section 7.3.2.1 and Section 7.3.2.2.

As in previous evaluations, we randomly select two initially known categories and average over ten splits. Each split is repeated with ten random initialization.



Figure 7.37: Object proposals extracted from the MS-COCO dataset by Lin et al. (2014) as described in Section 7.3.2.3. Top: segments accepted as nameable instances of the animal domain. Middle: segments representing categorical rejections (unrelated objects). Bottom: segments representing non-categorical rejections (no objects).

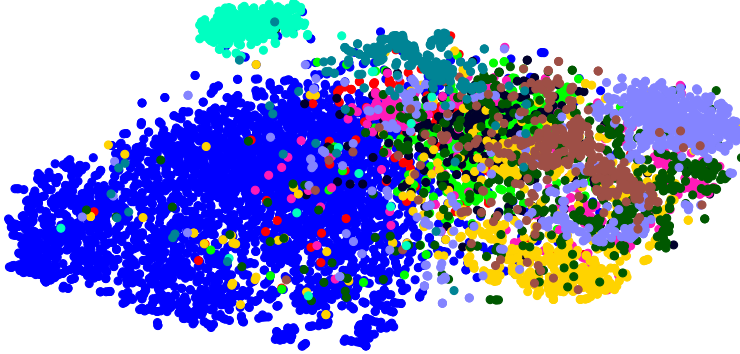


Figure 7.38: Visualization of object proposals computed from MS-COCO by Lin et al. (2014). See text for details of the proposal generation process. Resulting examples are non-linearly projected to $D = 2$ using the unsupervised t-SNE method by Van der Maaten and Hinton (2008). After projection, each data point is colored according to its category label. Unnameable instances are indicated in dark blue.

In each resulting experiment, we randomly select five examples of every known category for training. From all animal categories, we additionally select 30 hold-out examples of each category for testing. All remaining animal examples as well as all 10,000 unnameable instances build the unlabeled pool. The resulting dataset is shown in Fig. 7.38 after an unsupervised projection to $D = 2$ using t-SNE.

Results of active learning and class discovery are shown in Fig. 7.39. Note that the upper bound of the classification accuracy is left out for visualization purpose

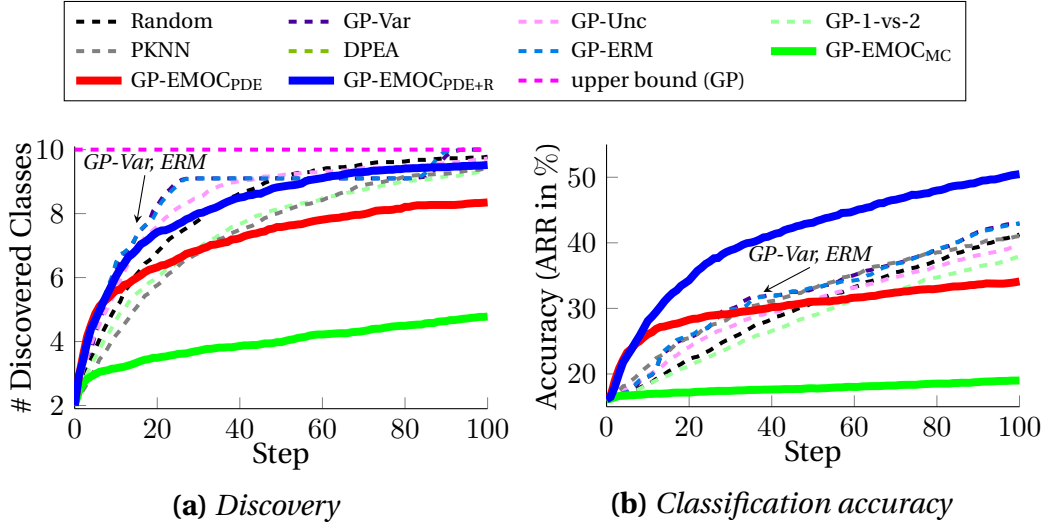


Figure 7.39: Results of active learning and class discovery for multi-class animal categorization using object proposals computed from MS-COCO by Lin et al. (2014). Each run consists of 100 queries (x-axis). Results are averaged over 10 random initializations. Left: number of known categories after each iteration. Right: average classification accuracy after each iteration evaluated on hold-out test data. Overlapping learning curves of multiple strategies are additionally labeled. Best viewed in color.

(training one-vs-all GP models from all training examples of the training set results in $\sim 76\%$ ARR). Classification accuracies after 100 queries are finally shown in Table 7.9.

Experimental Results: The quantitative results lead to several interesting observations. First of all, findings of previous experiments do not exactly match the outcome for learning to classify object proposals. In fact, the previously inferior techniques GP-Var, GP-Unc, and GP-ERM lead to the fastest discovery rates. On the contrary, 1-vs-2, the simple multi-class EMOC version GP-EMOC_{MC}, as well as its extension towards non-categorical rejections GP-EMOC_{PDE} perform significantly worse than random selection regarding class discovery. We attribute this behavior to the large amount of unnameable instances of both types, categorical and non-categorical. In addition, the large dimensionality of image representations hinders the discovery of consistent clusters in space (curse of dimension). Note that DPEA is again not applicable due to the large number of unlabeled examples and feature dimensions.

With respect to classification accuracy, we observe that the majority of techniques can hardly improve over the baseline of random selection. This further stresses the difficulty of the tackled scenario. Again, GP-Var and GP-ERM lead to

Table 7.9: Quantitative results from experiments shown in Sections 7.3.2.1 to 7.3.2.3 regarding active learning and class discovery in multi-class scenarios. Corresponding datasets are USPS (Bache and Lichman, 2013), LFW (Huang et al., 2007), and MS-COCO (Lin et al., 2014), respectively. Results are reported for average class-wise recognition (in %) after 100 queries and averaged over 100 random initializations and splits. (*) not possible due to excessive memory demand. See experimental setup description in corresponding sections for details.

Strategy	USPS	LFW	COCO
Random	87.27	62.01	41.27
GP-Var (Kapoor et al., 2010)	19.35	33.45	42.94
GP-Unc (Kapoor et al., 2010)	24.94	34.74	39.49
GP-1-vs-2 (Joshi et al., 2009)	49.54	80.58	37.87
PKNN (Jain and Kapoor, 2009)	53.26	78.88	41.03
DPEA (Hospedales et al., 2012)	92.20	—(*)	—(*)
GP-ERM (Roy and McCallum, 2001)	24.49	33.21	42.98
GP-EMOC _{PDE+R}	93.21	87.79	50.48

surprisingly good results compared with the previous observations on USPS and LFW. Nonetheless, the only technique which leads to clear accuracy gains over random selection is GP-EMOC_{PDE+R}. Thus, we conclude that modeling categorical and non-categorical rejections is beneficial for these real-world scenarios.

For additional qualitative impressions, we show the first five queries of passive learning, PKNN, and GP-EMOC_{PDE+R} in Fig. 7.40. As we can see, PKNN spends a remarkable ratio of the labeling budget on unnameable instances which does not lead to gainable information. In contrast, GP-EMOC_{PDE+R} puts a strong focus on nameable instances.

Summary: *In summary, modeling categorical and non-categorical rejections is crucial for real-world active learning tasks similar to biodiversity applications.*

7.3.2.4 Improvement versus Selection Time

In all previous evaluations, we considered the gain of classification accuracy or class discovery after spending a fixed labeling budget. This analysis is useful if the annotator is available upon request and the labeling cost is the only limiting resource. However, a similarly important aspect might be the time between annotation and selection of the next query. During that time period, the human-in-the-loop is idle which might waste valuable resources. Let us therefore repeat the previous evaluations with a focus on selection time.



Figure 7.40: Qualitative results of active learning and class discovery for multi-class animal categorization. For a single run, we show the first selected instances of passive learning (top), PKNN (middle), and GP-EMOC_{PDE+R} (bottom). We visualize corresponding regions in red, resulting bounding boxes in green, and the original image each region is extracted from. Animal instances which were successfully labeled are indicated with a green halo, whereas rejected samples are marked in orange.

Experimental Setup: We use the same experimental setup as in previous evaluations on USPS (Section 7.3.2.1), LFW (Section 7.3.2.2), and MS-COCO (Section 7.3.2.3). All splits and random initialization are kept identical, which leads to the same queried examples. In contrast to the previous evaluations, however, we now additionally record the query time of each method. Thereby, we can measure classification accuracy and class discovery as a function of passed time – including query selection and annotation.

Since our evaluations only simulate a human annotator, we investigate the influence of different annotation times. In detail, we assume annotation times of 1 s, 10 s, or 50 s for a single query. As previously, results are averaged over ten random splits with ten random initializations each. Results are shown in Fig. 7.41 for USPS, Fig. 7.42 for LFW, and Fig. 7.43 for MS-COCO.

Experimental Results: First of all, note that some curves end earlier than others due to the limited number of 100 queries. In all three evaluations, we observe that the expected risk minimization strategy GP-ERM is clearly unfeasible when the time

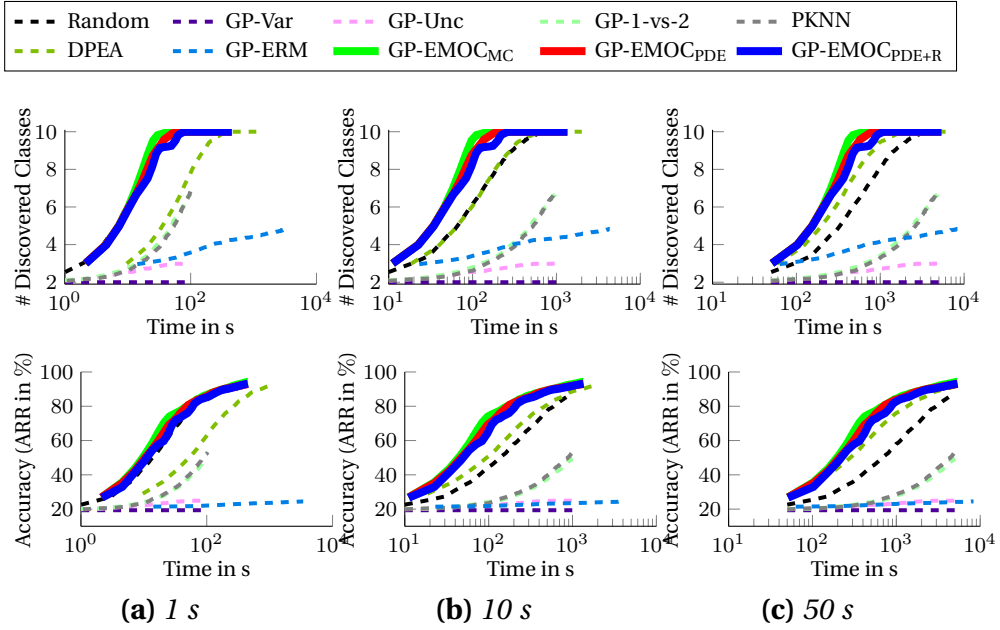


Figure 7.41: Active learning and class discovery versus total time spent on USPS. Results are identical to the evaluation shown in Fig. 7.33 but with the additional consideration of time spent for query selection and annotation. Total times are obtained for different assumptions of required labeling times per query: 1 s (left), 10 s (middle), and 50 s (right). Best viewed in color.

for query selection is limited. Especially for short annotation times of one second, the majority of time is spent for query selection using GP-ERM. This observation is consistent with the statements of Roy and McCallum (2001) who initially proposed the ERM strategy. DPEA is slightly faster than GP-ERM, although still an order of magnitude slower than PKNN. Note that random selection represents the lower bound for required time since the involved time for query selection is zero.

Regarding the EMOC variants, we conclude that they clearly require more computation time than the criteria by Kapoor et al. (2010) who rely on the predictive distribution without the necessity of model updates. Nonetheless, the difference is less visible for increased annotation times. Due to the superior results of EMOC in terms of classification accuracy and class discovery, we conclude that the additional selection time is well justifiable.

Summary: We conclude that EMOC requires an acceptable computational overhead given the superior classification and discovery results.

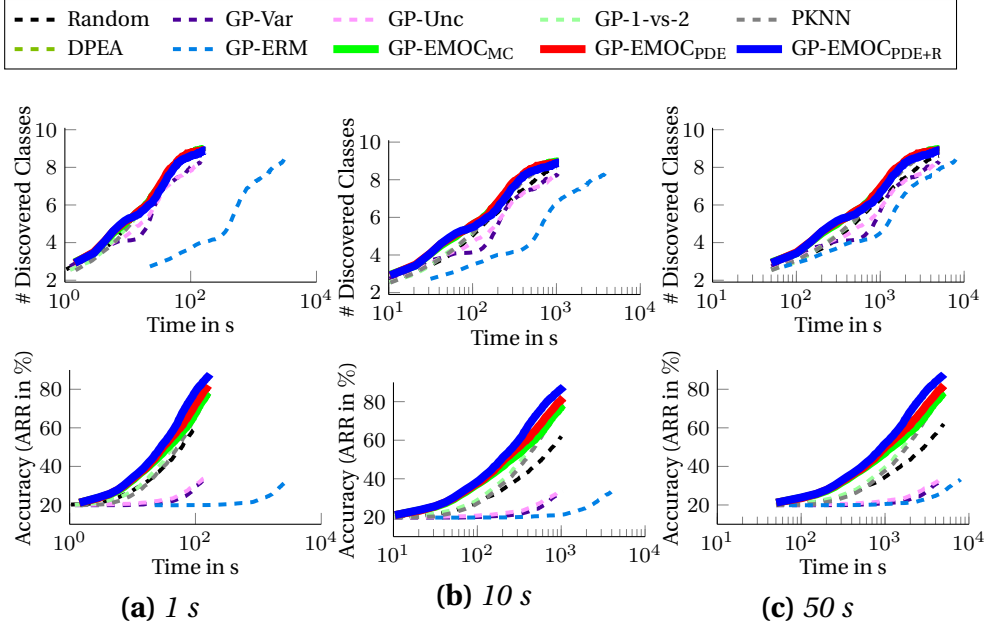


Figure 7.42: Active learning and class discovery versus total time spent on LFW. Results are identical to the evaluation shown in Fig. 7.36 but with the additional consideration of time spent for query selection and annotation. Total times are obtained for different assumptions of required labeling times per query: 1 s (left), 10 s (middle), and 50 s (right). Best viewed in color.

7.3.2.5 Qualitative Analysis of Active Learning for Multi-class Classification

In previous evaluations, we were mainly interested in quantitative statements regarding the suitability of different query strategies. A notable exception was the qualitative analysis in Section 7.3.1.1, where we analyzed inherent properties and limitations of different query strategies in a $D = 2$ toy example. This setup came with the clear advantage of being controllable, visualizable, and interpretable. In addition, we visually inspected queried examples in Section 7.3.1.4 and found them intuitively useful. We now provide a similar investigation for real-world multi-class data as used in the previous experiments.

Experimental Setup: In Section 7.3.2.3, we reported active learning results for multi-class classification scenarios on the MS-COCO dataset. For statistical significance, we averaged results over multiple random splits of data into labeled and unlabeled examples. Regarding discovery and classification accuracy, the GP-EMOC_{PDE+R} turned out to be highly suitable for the chosen scenario.

We now inspect the queried examples for two unique random splits: those

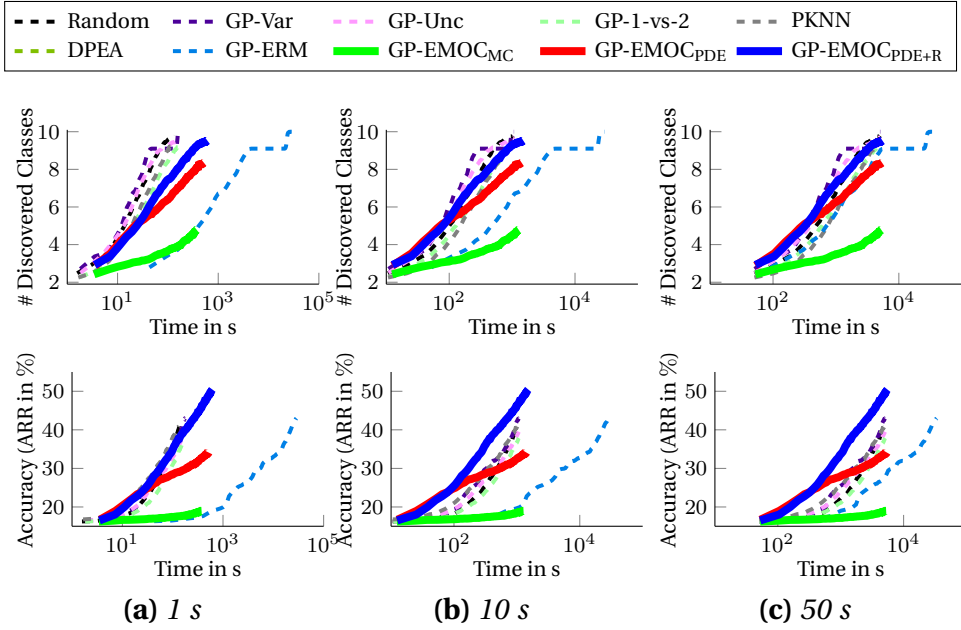


Figure 7.43: Active learning and class discovery versus total time spent on MS-COCO. Results are identical to the evaluation shown in Fig. 7.39 but with the additional consideration of time spent for query selection and annotation. Total times are obtained for different assumptions of required labeling times per query: 1 s (left), 10 s (middle), and 50 s (right). Best viewed in color.

which lead to smallest and highest classification accuracy after 100 queries. For each example, we show the corresponding unsupervised region segmentation obtained with geodesic object proposals (GOP) by Krähenbühl and Koltun (2014). Additionally, we display the resulting bounding box which serves for feature extraction. All settings are identical to the evaluation presented in Section 7.3.2.3.

We highlight selected examples which lead to strongest changes in classification accuracy. Thereby, we aim at visually inspecting characteristic queries. To allow for visual comparison, we further highlight examples which did not provide additional information. Results for best and worst scenario are shown in Fig. 7.44. We further display all training examples as well as all chosen queries for the best split in Fig. 7.45.

Experimental Results: Besides their visual beauty, we are not able to see any specific characteristics for these images that might give reason for the performance difference of the best and the worst run.

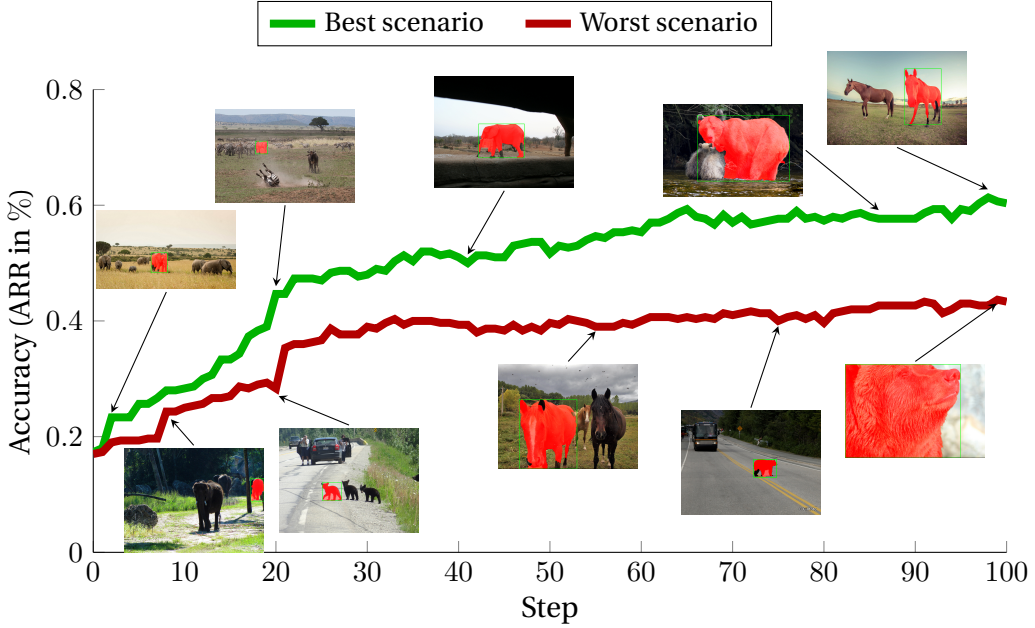


Figure 7.44: Qualitative analysis of individual runs in multi-class active learning experiments. Results show the best and worst run of $\text{EMOC}_{\text{PDE+R}}$ on the MS-COCO dataset as reported in Section 7.3.2.3. We additionally displayed chosen examples which either lead to strong information gains or which did not contain novel information at all. Red segmentations correspond to unsupervised object proposals computed with GOP by Krähenbühl and Koltun (2014). The bounding box which serves for feature extraction to represent the unlabeled example is shown in green.

Summary: In summary, we argue that active learning techniques should mainly be compared quantitatively.

7.3.2.6 Detailed Analysis for EMOC (1): Loss Functions

In Section 5.5.1, Eq. (5.25), we introduced the EMOC criterion for arbitrary classification scenarios. For the case of multi-class classification, we then derived specific computations in Section 5.6.2. The derivations were based on specifying the space of possible labels $\mathcal{Y} = \{1, \dots, M\}$ and corresponding loss functions to compute changes of model outputs. In this section, we evaluate the suitability of different loss functions.

Experimental Setup: We follow the previous experimental setup used in Section 7.3.2.3 for learning from object proposals on MS-COCO. For comparability, we fix all settings and only change the loss function to compute EMOC scores.

Changes of discrete class labels are counted using the 0/1-loss introduced in



Figure 7.45: Qualitative analysis of the best split obtained in the multi-class active learning experiment on MS-COCO. Top row shows all training examples which cover the initially known categories giraffe and bird. Subsequent rows show selected examples for labeling in consecutive order. As in Fig. 7.44, red segmentations correspond to unsupervised object proposals computed with GOP by Krähenbühl and Koltun (2014). Green rectangles indicate the obtained bounding box which serves as unlabeled example for feature extraction. Figure is best viewed in color and by zooming in.

Eq. (2.2). A less stringent alternative is the $\mathcal{L}_{|\cdot|}$ as motivated in Section 5.6.2 (see Eq. (5.60)). We follow previous evaluations and use classification accuracy as well as the number of discovered categories as performance indicator. The labeling budget is fixed to 100 iterations. We average results over the same 10 random splits used in previous evaluations. Results are shown in Fig. 7.46.

Experimental Results: As can be seen, rewarding only discrete changes of most probable class labels is clearly inferior to changes of continuous classification scores. We attribute this behavior to two aspects. First of all, continuous scores are often more reliably estimated compared to hard decisions. In fact, only tiny changes in continuous scores can result in label flips, especially if the classification uncertainty is high. These scenarios are typical for early stages of learning where

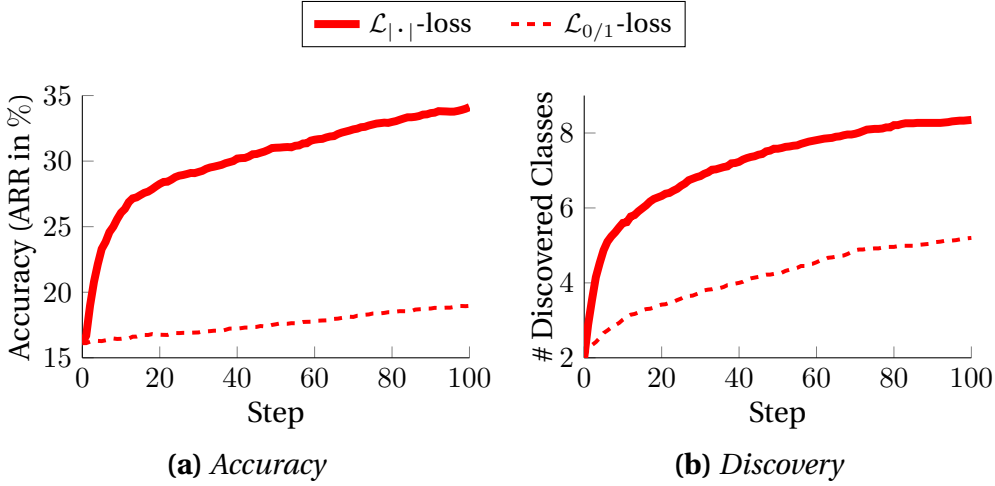


Figure 7.46: Evaluating different loss functions within EMOC for multi-class scenarios. Discrete changes of estimated labels are reflected using a 0/1-loss (see Eq. (2.2)). Changes of continuous model outputs are measured using the $\mathcal{L}_{|\cdot|}$ -loss (see Eq. (5.60)). Accuracy after each iteration is measured as average recognition rate (left) and number of discovered classes (right).

available labeled data is insufficient for reliable class estimates.

Furthermore, changes in hard decisions do not reward samples that would confirm current class estimates. In contrast, the $\mathcal{L}_{|\cdot|}$ -loss also rewards score changes which result in decreased classification uncertainties. Thus, the 0/1-loss can be seen as complementary to commonly used entropy-reduction techniques (see Section 5.2.4) whereas $\mathcal{L}_{|\cdot|}$ is smoothly located between both extremes.

Summary: In summary, a soft $\mathcal{L}_{|\cdot|}$ -loss for multi-class scenarios is preferable when evaluating EMOC.

7.3.2.7 Detailed Analysis for EMOC (2): Probability Estimates

To evaluate EMOC scores in multi-class scenarios, we proposed to follow Fröhlich et al. (2013) by computing class probabilities using a Monte-Carlo technique (see Eq. (5.62)). We now investigate whether the choice is advisable for our scenarios.

Experimental Setup: As for the previous comparison of different loss functions, we again follow the experimental setup used in Section 7.3.2.3 for learning from object proposals on MS-COCO. For comparability, we fix all settings and only evaluate the effect of changing techniques for estimating multi-class probabilities.

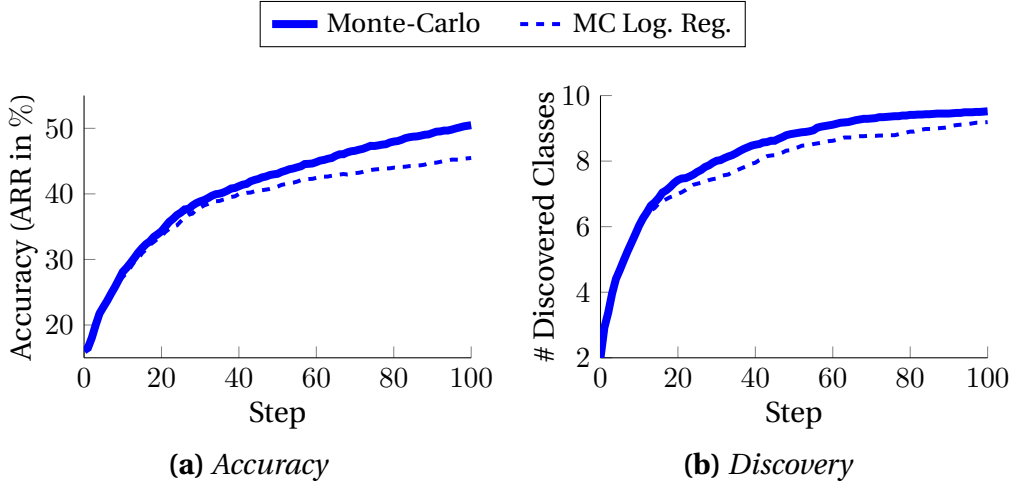


Figure 7.47: Evaluating different techniques to estimate multi-class probabilities within EMOC for multi-class scenarios. Monte-Carlo refers to the sampling strategy by Fröhlich et al. (2013) as used in previous experiments (see Eq. (5.62)). MC Log-Reg is the multi-class logistic regression approach introduced in Eq. (5.61). Accuracy after each iteration is measured as average recognition rate (left) and number of discovered classes (right).

The baseline is the sampling technique by Fröhlich et al. (2013) as used in previous evaluations. We compare against the multi-class logistic regression introduced in Section 5.6.2, Eq. (5.61). Class-specific parameters α_m and β_m are estimated using Eq. 5 and Eq. 6 of (Duan et al., 2003). During the parameter estimation, classification scores are obtained from training data using leave-one-out estimates (see (Rasmussen and Williams, 2006, Sect. 5.4.2) for a similar discussion). Thereby, estimated parameters are better suited to fit the training data compared to estimates obtained from a separate validation dataset.

As in previous evaluations, we use the classification accuracy as well as the number of discovered categories as performance indicator. The labeling budget is fixed to 100 iterations. We average results over the same 10 random splits as used in previous evaluations. Results are shown in Fig. 7.47.

Experimental Results: In direct comparison, we find that both probability estimates are comparable and show only marginal performance differences. For larger numbers of queries, the sampling technique leads to slightly superior results. Thus, both techniques would be well applicable which justifies using the sampling approach for the previous evaluations.

Summary: *In summary, Monte-Carlo estimates for class probabilities are well suited to evaluate EMOC.*

7.4 Experiments on Learning of Mid-level Representations

In all previous evaluations, we assumed image representations to be given. This assumption covered handcrafted representation (BoW on low-level SIFT features, HOG, or ColorNames) as well as activations of deep neural networks which have been trained on auxiliary data. However, this assumption might be unfeasible in lifelong learning scenarios. In this section, we go one step further and investigate our techniques for feature learning and localized feature extraction as introduced in Chapter 6. We will focus on fine-grained recognition tasks (see Section 6.1.4) since discriminative and localized features are highly important for these scenarios. In consequence, the Caltech-UCSD Birds-200-2011 (CUB-2011) dataset as introduced in Section 7.1.1.7 serves as perfect benchmark throughout the entire section.

7.4.1 Part Annotation Transfer for Fine-grained Recognition

We start with experiments regarding the transfer of semantic annotations from training images to previously unseen test images. Thus, we assume that part annotations for all training images are given. For the CUB-2011 dataset, we already showed two examples in Fig. 7.6.

In Section 6.4, we introduced a technique to transfer part annotations from birds in similar pose to the current object of investigation. If the annotation transfer was successful, we can extract localized features from these semantic positions to build a semantically rich description of the bird. We thus need to evaluate three aspects of the presented technique:

- (i) Section 7.4.1.1: How precise is the localization accuracy of transferred parts?
- (ii) Section 7.4.1.2: Which benefit comes with localized feature extraction from transferred parts?
- (iii) Section 7.4.1.3: Which annotation accuracy is required for fine-grained recognition?

We made the source code used in this section publicly available at <https://github.com/cvjena/finegrained-cvpr2014.git>. An introducing video can additionally be found at <https://youtu.be/mKd7GnEFs6w>.

7.4.1.1 Localization Error Analysis (1) – Comparison with DPMs

Let us start with an experiment regarding the localization of discriminative parts. As in all following evaluations, we crop images to provided bounding boxes and rescale them to $256\text{px} \times 256\text{px}$. Note that the assumption of a bounding box is not too strong for practical applications given the recent success of object detection algorithms (*e.g.*, latest results of the detection challenge on ImageNet (Russakovsky et al., 2015)).

Experimental Setup: Since our technique is non-parametric, we compare against a parametric counterpart. In detail, we use all examples of the training set to train a DPM by Felzenszwalb et al. (2010). Training is done using the publicly available source code of Girshick et al. (2014b). Details of DPMs are given in Section 2.2.4. We train 3 and 5 DPM components for the CUB-2011/14 and CUB-2011/200 dataset, respectively, to reflect different bird poses and aspect ratios. As initialization for DPM training, we use the mean position of each provided part. Thereby, each part of the otherwise semantic-free DPM can be associated with a semantically meaningful bird part. For remaining settings, we use the default values provided with the source code.

Training images for our part annotation transfer technique are represented by arrays of HOG features using the provided source code of Hariharan et al. (2012). We follow common data augmentation techniques and add flipped versions of all training images. Thereby, we allow for matches between birds in complementary orientations. We evaluate the localization accuracy for three parts distributed over the entire body: back, nape, and left leg. Accuracy of predicted part locations is measured as Euclidean distance between ground truth location and inferred location in the resolution of original input images. Since sizes of bounding boxes and aspect ratios vary significantly over the dataset, we report Euclidean errors normalized to the bounding box length. Nonetheless, we additionally show the localization errors in pixel difference for completeness. For visualization, we rank test images according to an increasing error individually for each method and part. Results are shown in Fig. 7.48 and Fig. 7.49. The visualization for the large dataset shows only every 5th example. Note that a qualitative example is given in Section 6.4.2, Fig. 6.14.

Experimental Results: First of all, we note that our non-parametric approach consistently leads to smaller localization errors than the DPM baseline. This observation holds for both the small 14 class subset as well as the entire dataset with 200 species. The absolute localization accuracy of our approach is surprisingly high with mean Euclidean errors of only 5.4%, 5.3%, and 7.8% on CUB-2011/200 for back, nape, and left leg, respectively. Besides, we observe strong variations of

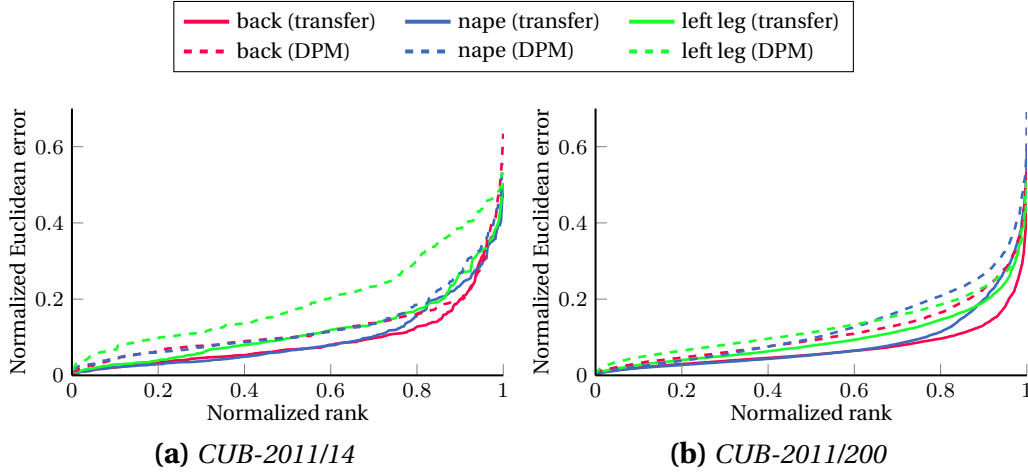


Figure 7.48: Normalized localization error for part localization with DPMs or part annotation transfer (denoted with transfer). Best viewed in color.

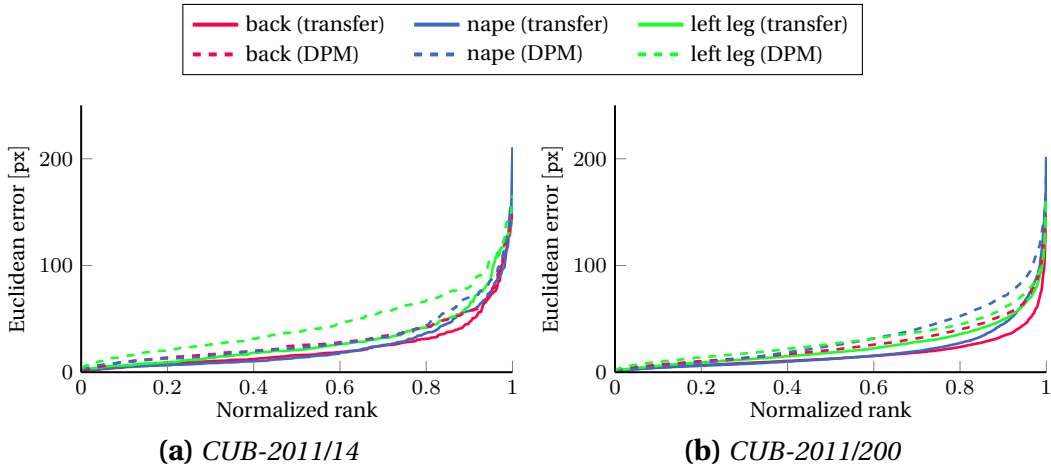


Figure 7.49: Localization error for part localization with DPMs or part annotation transfer (denoted with transfer) in absolute pixel differences. Best viewed in color.

localization accuracies among different parts. While back and nape can be robustly estimated, the errors for legs are clearly larger. This effect is likely caused by the strong variations of parts like leg and wing, which in turn is due to the strong pose variations birds usually show. A similar observation was reported before by the authors of the dataset (Wah et al., 2011b).

With respect to errors in pixel distances (Fig. 7.49), we can confirm the previous findings. Note further that the mean Euclidean error in pixel is 12.7 px, 12.4 px, and 18.4 px on CUB-2011/200 for back, nape, and left leg, respectively.

Summary: *In summary, our part annotation transfer has a better part localization accuracy than the DPM baseline while being simple and easy to implement.*

7.4.1.2 Fine-grained Recognition with Transferred Part Annotations

Based on localized part positions, we can evaluate the entire recognition pipeline as introduced in Section 6.4.2, Fig. 6.15.

Experimental Setup: To represent bird images, we use global and part information. For the latter, we represent each part with BoW histograms. Local features are extracted from each part and part-specific codebooks are created using K -means with 100 components. To capture color and texture information, we extract ColorName features with source code by Van De Weijer et al. (2009) as well as OpponentSIFT descriptors using the VLFeat library (Vedaldi and Fulkerson, 2008). Each resulting BoW histogram is L_1 -normalized and histograms for each part are concatenated.

Global information is captured by computing additional BoW histograms from local features which cover the entire bounding box. We reflect the larger amount of features by computing 1,000 codebook prototypes. To add little spatial information, we additionally apply spatial pyramid pooling by Lazebnik et al. (2006) with 3 levels (*i.e.*, the entire box as well as two divisions). Since a bounding box often covers significant amount of background, we additionally mask estimated background regions using GrabCut by Rother et al. (2004). All features are finally concatenated to train a linear SVM using LibLinear (Fan et al., 2008) with $C = 10$. To allow for non-linear separations, we further apply explicit feature transformations as presented by Vedaldi and Zisserman (2012) to approximate a χ^2 -kernel with parameter $\gamma = \frac{1}{2}$.

At test time, we apply the same steps on estimated part locations, either from the DPM baseline or using our part annotation transfer technique. If a part could not be localized or transferred, we use zero imputation for the corresponding feature vector (Saar-Tsechansky and Provost, 2007). Note that we also investigated more sophisticated imputation methods. However, we did not observe significant difference in terms of the resulting recognition accuracy. For our annotation transfer technique, we further repeat the classification pipeline for the K nearest neighbors and average classification scores to obtain final class estimates.

We compare against baselines which have been published at the same time, *i.e.*, before CNNs become omnipresent. In detail, we compare against deformable part descriptors (DPD) by Zhang et al. (2013), part-based one-vs-one features (POOF) by Berg and Belhumeur (2013), pooling-invariant image feature learning (PDL) by Jia et al. (2013), the template learning approach of Yang et al. (2012), and the patch discovery with style-awareness by Lee et al. (2013).

Table 7.10: Mean accuracy results on CUB-2011.

Approach	CUB-2011/14	CUB-2011/200
SVM + BoW (Wah et al., 2011b)	—	10.25%
Style-awareness (Lee et al., 2013)	—	38.31%
PDL (Jia et al., 2013)	—	38.91%
Template learning (Yang et al., 2012)	—	43.67%
DPD (Zhang et al., 2013)	—	50.98%
POOF (Berg and Belhumeur, 2013)	70.10%	56.78%
DPM part localization	67.35%	39.30%
Part annotation transfer ($K = 1$)	69.71%	54.94%
Part annotation transfer ($K = 5$)	73.40%	57.99%
CNN + part localization (Zhang et al., 2014)	—	76.37%
CNN + part constellation (Simon and Rodner, 2015)	—	81.01%

Finally, we also add recent results obtained with fine-tuned and post-processed CNN models. Zhang et al. (2014) presented how CNN features extracted at localized head and body region can boost recognition performance dramatically. Simon and Rodner (2015) introduced a technique to discover unsupervised part constellations to extract CNN features from. However, it should be noted that both approaches are not directly comparable to remaining techniques due to the extremely descriptive CNN features used. Thus, they are meant to show how fast the state-of-the-art shifted within several months due to the availability of discriminatively learned deep image representations. Results are shown in Table 7.10. Classification accuracy is reported as mean average recognition (ARR) for CUB-2011/14 and CUB-2011/200. Note that we reported ORR values in our earlier publication (Göring et al., 2014). However, we chose ARR to be comparable with reported results of cited competitors. See also Section 7.1.2.1 for an introduction to both metrics.

Experimental Results: First of all, it is interesting to note that already a simple DPM for part localization coupled with standard BoW features is able to compete with sophisticated techniques for discriminative feature learning. Besides, we find that our part annotation transfer further boosts the classification accuracy and even outperforms the computationally demanding competitors. This is indeed remarkable given the simplicity of the underlying idea for part annotation transfer. Last but not least, recent results with CNN architectures demonstrate impressively the importance of learning deep image representations from visual data.

In addition to quantitative numbers, we also provide a qualitative comparison

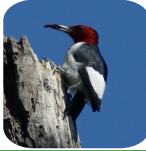
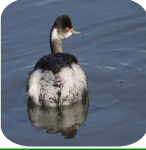
	Est. Class: Ours: 'Red faced Cormorant' DPM: 'Ruby throated Hummingbird'	Score: 0.7628 -0.8140	Result: ✓ ✗
	Est. Class: Ours: 'Mourning Warbler' DPM: 'Nashville Warbler'	Score: 0.3917 -0.4545	Result: ✓ ✗
	Est. Class: Ours: 'Hooded Oriole' DPM: 'Yellow Headed Blackbird'	Score: -0.4184 0.2345	Result: ✗ ✓
	Est. Class: Ours: 'Red Headed Woodpecker' DPM: 'White breasted Nuthatch'	Score: 0.2970 -0.8739	Result: ✓ ✗
	Est. Class: Ours: 'Eared Grebe' DPM: 'Clark Nutcracker'	Score: 0.3808 -0.6085	Result: ✓ ✗
	Est. Class: Ours: 'Lazuli Bunting' DPM: 'Blue headed Vireo'	Score: 0.3172 -0.6267	Result: ✓ ✗

Figure 7.50: Qualitative comparison of part detection with DPM and our part annotation transfer. We selected individuals for visualization where the DPM baseline and our approach resulted in strongest differences.

Table 7.11: Analyzing impact of individuals steps of the fine-grained recognition pipeline. Results are shown on the CUB-2011/14 dataset. Accuracy is measured as average class-wise recognition rates (ARR).

Method	CUB-2011/14
NN part transfer	69.71%
... without GrabCut masking only	68.43%
... without part features only	66.25%
... without global features only	61.24%

of classification results when parts are estimated with DPMs or our transfer technique. Fig. 7.50 shows six cases of CUB-2011/200 where both techniques resulted in strongest differences of classification scores. As we can see, part annotations can be reliably transferred in the majority of scenarios which allows for estimating the bird's category correctly. A typical scenario where our transfer fails is the middle entry of the left column. Here, the background is heavily cluttered which misleads our transfer technique.

As final evaluation, we investigate the impact of several steps in the classification pipeline using the CUB-2011/14 dataset. Results are shown in Table 7.11. We notice that the combination of global and local information adds valuable information. Furthermore, masking of background using GrabCut results in additional little improvement.

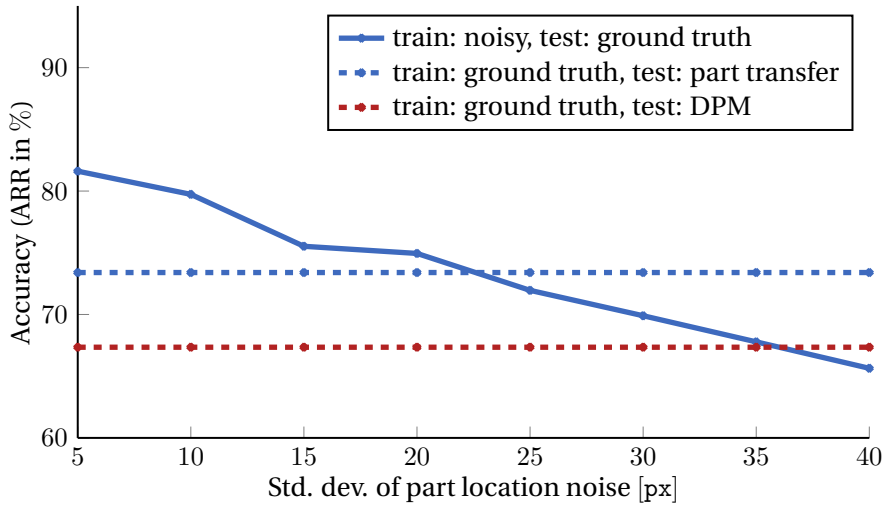


Figure 7.51: Analyzing the influence of part annotation errors on the final classification decisions. *X*-axis shows standard deviation in pixel of Gaussian noise added to part locations in training images. Results are obtained on the CUB-2011/14 dataset.

Summary: In summary, a simple part transfer and standard BoW features are comparable or even superior to sophisticated feature learning techniques of the same era. Furthermore, deep architectures again significantly boost performance. Hence, replacing BoW features with CNN representations is likely to further improve classification accuracy.

7.4.1.3 Localization Error Analysis (2) – Noisy Annotations

Our approach for transferring part annotations heavily relies on manually specified part positions in training images. Since human annotators can vary in the provided annotations, we investigate the effect of annotation noise on the resulting classification accuracy.

Experimental Setup: We use the classification pipeline as described in the previous experiments. To simulate varying precision of part annotation during training, we add i.i.d. Gaussian noise to the ground truth locations of training images. Decreasing annotation accuracy is reflected by increasing standard deviations of the additive noise. For test images, we use ground truth locations to simulate perfect part localization. As previously, we conduct experiments on the CUB-2011/14 dataset. Results are shown in Fig. 7.51.

Experimental Results: It is clearly visible that precisely annotated parts are highly beneficial for fine-grained recognition. Thus, special care should be taken when annotating training examples, *e.g.*, by special training or averaging over multiple annotators as done by Wah et al. (2011b).

Furthermore, we observe that our non-parametric technique for transferring exact training locations to test images leads to comparable results as if ground truth test locations are used for feature extraction and training locations are disturbed $\sigma = 20\text{px}$. Thus, the existing pipeline could benefit from an further improved annotation transfer technique. A possible extension could replace HOG by CNN activations for improved matching. Complementary, spatial refinement as commonly done in image retrieval scenarios is likely beneficial (Philbin et al., 2007).

Summary: *We conclude that accurate part annotations in training images are beneficial for fine-grained recognition.*

7.4.2 Mid-level Features for Fine-grained Recognition

In the previous section, we evaluated the impact of localized features extraction when human annotations are available. Since annotations are still costly, we introduced a technique for learning of localized mid-level features in Section 6.1.3. In the following, we will evaluate this technique with respect to three issues:

- (i) Section 7.4.2.1: How discriminative are learned representations?
- (ii) Section 7.4.2.2: What is the benefit of individual steps during patch discovery?
- (iii) Section 7.4.2.3: Do learned features correspond to localized parts?

As in the previous section, we conduct all experiments on the CUB-2011 dataset. We made the source code used in this section publicly available at <https://github.com/cvjena/patchDiscovery.git>.

7.4.2.1 Bird Identification Using Mid-level Features

Experimental Setup: We follow previous suggestions and crop images to provided bounding boxes enlarged by 10% and finally rescale them to a fixed size of $256\text{px} \times 265\text{px}$. During seeding, we mask background regions using the image masks provided with the dataset. We also experimented with estimating foreground masks using GrabCut by Rother et al. (2004) but found no clear difference in produced seeding results. Background statistics Σ_0 and μ_{-1} for LDA models are computed in advance using the entire training set and the source code by Hariharan et al. (2012). We published an extended version of their source code with flexible interfaces at <https://github.com/cvjena/whoGeneric.git>.

Table 7.12: *Classification accuracy of mid-level features for fine-grained recognition on the CUB-2011 dataset.*

Approach	CUB-2011/14	CUB-2011/200
SVM + BoW (Wah et al., 2011b)	—	10.25%
Mid-level features (unsupervised)	54.01%	—
Mid-level features (supervised)	58.01%	39.35%
Style-awareness (Lee et al., 2013)	—	38.31%
PDL (Jia et al., 2013)	—	38.91%
Template learning (Yang et al., 2012)	—	43.67%
DPD (Zhang et al., 2013)	—	50.98%
POOF (Berg and Belhumeur, 2013)	70.10%	56.78%
Part annotation transfer, Section 7.4.1	73.40%	57.99%

Our technique supports bootstrapping of patch detectors in a supervised or unsupervised fashion and we evaluate both settings. During bootstrapping, we fix the number of added examples per iteration to $m = 1$. The parameter γ for extracting bootstrap candidates was set to $\gamma = 0.3$. We stop the process of bootstrapping after at most five iterations. With trained patch detectors, we encode all training images using the largest activations as described in Section 6.2.4. As final classifier on detector responses serves a linear SVM using LibLinear (Fan et al., 2008). We also tested non-linear SVMs using LibSVM or explicit embeddings by Vedaldi and Zisserman (2012) as in the previous section but found no clear advantage. All remaining settings for the individual steps are specified as default values in the released source code. As baselines serve the same techniques used in the previous evaluation in Section 7.4.1.2. Results are shown in Table 7.12.

Experimental Results: First of all, the supervised bootstrapping is superior to its unsupervised pendant on the small dataset CUB-2011/14. For the large dataset, the computation time for the unsupervised approach even exceeded our available budget. However, the implementation was run on a single core without optimized implementations. Thus, the result of the unsupervised setting can be seen as proof of concept that even without labels, learning of mid-level features is possible without a dramatic loss in discrimination abilities.

Besides, we note that our approach clearly outperforms the simple baseline of BoW and SVM by Wah et al. (2011b). Furthermore, it is on par with the technique of Lee et al. (2013) who also presented a pipeline for discovery of mid-level features. However, incorporating a discrimination criterion into feature learning as done by

Berg and Belhumeur (2013) significantly boosts performance (POOF). Similarly, our previously investigated part annotation transfer with localized feature extraction is clearly superior in terms of classification accuracy.

Summary: *In summary, unsupervised discovery of mid-level features is possible but is computationally demanding. Our supervised technique leads to classification results comparable to techniques for learning of mid-level features. However, it does not obtain state-of-the-art results on its own.*

7.4.2.2 Analysis of Individual Steps

The introduced pipeline for fine-grained recognition with discovery of mid-level features consists of several steps. While we analyzed the overall accuracy in the previous experiment, we now investigate the impact of individual steps in more detail.

Experimental Setup: The entire setup is identical to the previous evaluation. For simplicity, we restrict the investigation to the small dataset CUB-2011/14.

First of all, we evaluate our seeding technique against human provided ground truth locations. With respect to bootstrapping, we investigate the effect of our technique by comparing it against no bootstrapping at all, *i.e.*, detectors trained from a single seeding block only (bootstrapping: none). Furthermore, we compare our convolutional bootstrapping against the approach suggested by Juneja et al. (2013) which allows only seeding blocks to be added. With respect to additional selection of potentially relevant detectors, we compare our no-selection strategy against two alternatives: the entropy-rank criterion with additional merging of similar filters as suggested by Juneja et al. (2013) and the removal of detectors with too few positive examples as introduced by Juneja et al. (2013). Results are shown in Table 7.13.

Experimental Results: Regarding bootstrapping, we observe that the evaluation of all possible locations and scales as candidates for bootstrapping is beneficial. Interestingly, detectors trained from seeding blocks alone lead only to a minor decrease in accuracy. With respect to selection of potentially useful detectors, we find that the entropy-rank criterion leads to dramatic accuracy decrease in our scenario. In contrast, removing singletons after bootstrapping slightly increases performance which is consistent with the findings of Singh et al. (2012). Finally, our seeding technique leads to comparable results as if patch detectors are trained from human part annotations. Thus, the expensive acquisition of part annotations can be circumvented.

Table 7.13: *Influence of individual steps during patch discovery on the classification accuracy.*

Approach	CUB-2011/14
Bootstrapping: on seeding blocks only (Juneja et al., 2013)	46.93%
none	55.64%
Selection: entropy-rank criterion with merging (Juneja et al., 2013)	26.96%
representative, without singletons (Singh et al., 2012)	58.62%
Our approach (globally discovered parts, convolutions, no selection)	58.01%
Seeding: <i>human annotated semantic parts</i>	57.89%

Summary: We conclude that our choices for each step fit well together. Thereby, no human part annotation nor a heuristic for part selection is required.

7.4.2.3 Activation Analysis of Learned Filters

In a final experiment, we visually analyze whether learned patch detectors correspond to intuitive parts of birds.

Experimental Setup: We use all discovered patch detectors from the previous experiment on CUB-2011/14 after bootstrapping. Patch detectors are evaluated on all positions and scales for training and test images, respectively. In contrast to the training phase, we evaluate patch detectors on original images without being cropped to their bounding boxes to see possible limitations of our approach. Images are further scaled to squared aspect ratio. Resulting detection response maps for all filters are summed up individually for each image to obtain a single activation map. We finally visualize activation maps using Matlab’s Jet colormap where warm colors indicate strong responses.

First of all, we are interested to see whether our learned patch detectors correspond to discriminative regions in images they were discovered from. Therefore, activation maps of six training images are shown in Fig. 7.52. In addition to the analysis on training images, it is interesting to see whether learned patch detectors generalize to previously unseen test images. We therefore repeat the same analysis on hold-out test data. Again, we show activation maps for six images in Fig. 7.53.

Experimental Results (1) – Known Images From The Training Set: As we can see, activations of learned detectors are especially focused on regions which are intuitively useful for discrimination. As an example, the striped head pattern in the top left image is sufficient to identify the bird as an Pileated Woodpecker in



Figure 7.52: Visualizing patch detector responses on training images. Left and middle column show successful activations whereas the right column shows failure cases. Activation maps are color coded using Matlab's *Jet colormap*. Warm colors indicate strong responses. Best viewed in color.



Figure 7.53: Visualizing patch detector responses on unseen test images. Left and middle column show successful activations whereas the right column shows failure cases. Activation maps are color coded using Matlab's *Jet colormap*. Warm colors indicate strong responses. Best viewed in color.

CUB-2011/14. Similarly, the dotted breast is a typical feature for Northern Flickers. However, in case of severe occlusions or extensive background structures, strong activations are mislead. Thus, the mentioned cropping to bounding boxes helps in these scenarios.

Experimental Results (2) – Unknown Images From The Test Set: We observe that the findings from training images similarly hold for previously unseen test data. As an example, in the bottom left image detectors are most strongly activated on olive-green upperparts and the white eye region, which in combination allows for identifying the bird correctly. We again see failure cases in case of prominent background structures which can be circumvented by cropping to bounding box regions.

Summary: *In summary, we find that our discovered patch detectors are most strongly activated on intuitively useful object parts and generalize well from training to test images.*

7.4.3 Exemplar-specific Representations via Local Learning

In Section 6.3, we extended our discovery technique to exemplar-specific representations. Thereby, computational limitations during training can be circumvented at the cost of additional overhead at test time. Let us now investigate the resulting impacts of local learning regarding the discovery of mid-level patches in fine-grained recognition scenarios. Therefore, we will analyze four aspects:

- (i) Section 7.4.3.1: How accurate is the matching for retrieving the neighborhood of a test image?
- (ii) Section 7.4.3.2: What is the final classification accuracy of exemplar-specific patch discovery?
- (iii) Section 7.4.3.3: How compact are exemplar-specific representations?
- (iv) Section 7.4.3.4: Do exemplar-specific patch detectors correspond to intuitive parts?

We again focus our experiments on the CUB-2011 dataset as running example. All experiments were conducted with our released source code which is publicly available at <https://github.com/cvjena/patchDiscovery.git>.

7.4.3.1 Matching Accuracy for Local Learning

Local learning of representations and classifiers can only be successful if at least one example of the correct category is among the local neighbors. In a first evaluation, we thus investigate the precision of nearest neighbor matching as introduced in Section 6.3.2.

Experimental Setup: Similar to the evaluations regarding part annotation transfer in Section 7.4.1, we represent images with a combination of HOG and ColorNames to capture texture, shape, and color information. Note that using HOG alone is likely insufficient. Instead, we require an accurate retrieval of visually similar instances to then discover subtle differences among them. However, using HOG alone was justifiable for capturing similar poses in Section 7.4.1 as required for the part annotation transfer. HOG features are extracted using the source code of Hariharan et al. (2012). Color information is captured using ColorName features extracted with the source code of Van De Weijer et al. (2009). Spatial information is

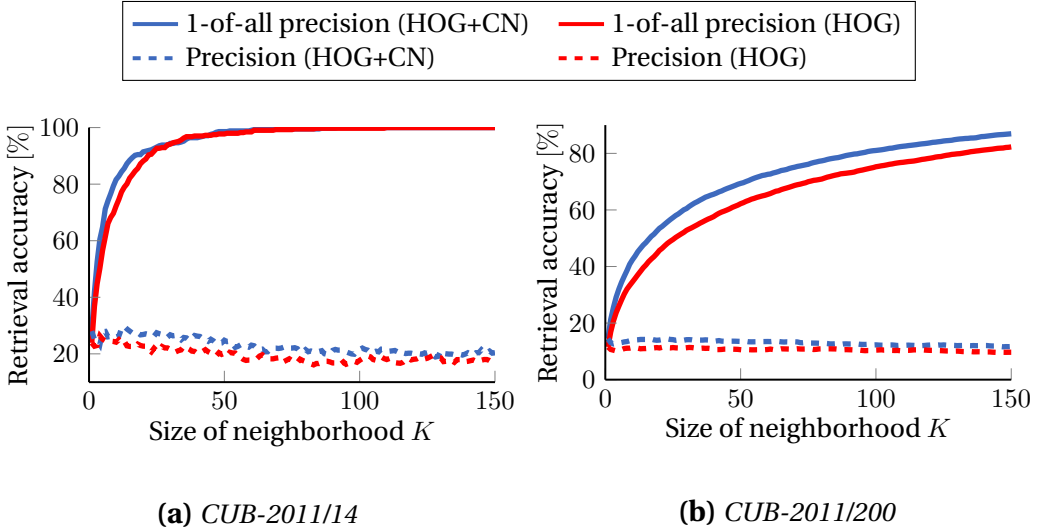


Figure 7.54: Analyzing the matching accuracy for local learning of patch detectors. Matching accuracy is measured as 1-of-all precision and precision, respectively.

kept by spatially dividing images into arrays and extracting features for each cell separately as usually done for HOG. Both features are concatenated to form the final representation.

Matching of test images and training images is done by computing Euclidean distances between global representations and returning the K most similar examples.

We evaluate the accuracy of matching using two metrics introduced in (Freytag et al., 2015, Sect. 4.1). *Precision* reflects the ratio between retrieved examples of the same category as the test image and the total amount of retrieved images. In contrast, *1-of-all precision* can be seen as an oracle evaluation and return one if at least one retrieved image is of the correct category. Both metrics are evaluated for each test image with varying sizes of neighborhoods and averaged over all test images. We compare our matching against the baseline of using HOG alone as done for the part annotation transfer. Results are shown in Fig. 7.54.

Experimental Results: As we can see, incorporating color information into the retrieval process is beneficial. Regarding CUB-2011/14, we observe that already small neighborhoods of $K = 20$ examples contain at least one example of the correct category in $> 90\%$. For the large CUB-2011/200 dataset, significantly larger K are required. In fact, not even $K = 150$ results in 90% statistical probability of containing at least a single correct example. Thus, better matching techniques are likely required as shown in (Freytag et al., 2015). Improved precision can either be obtained using better features such as CNN activations, or using post-processing

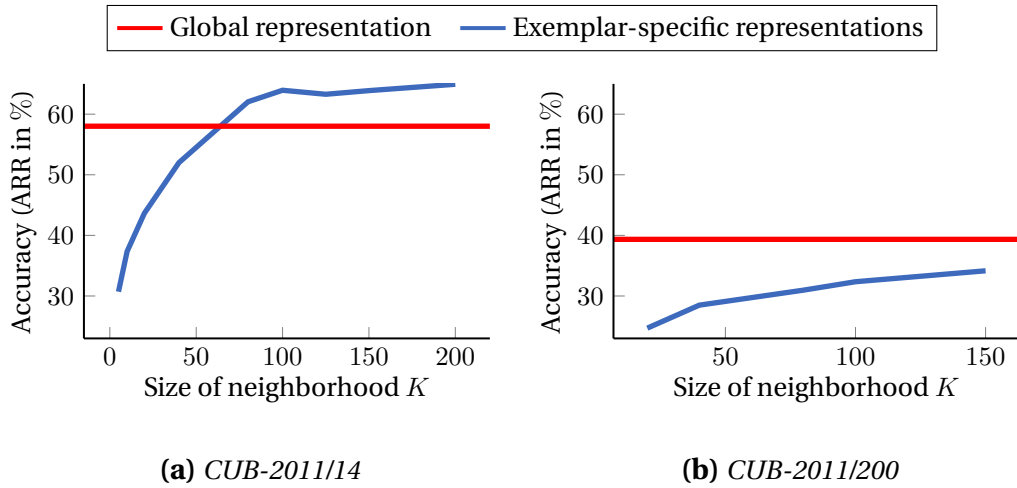


Figure 7.55: Dependency of classification accuracy and neighborhood sizes for exemplar-specific mid-level features on CUB-2011/14 (left) and CUB-2011/200 (right).

techniques as mentioned in Section 7.4.1.3.

Summary: *In summary, we find that the combination of texture, shape, and color is relevant to compute useful neighborhoods for local learning. However, the matching would benefit from further improvements presented in the image retrieval community.*

7.4.3.2 Fine-grained Recognition with Local Learning

In the last experiment, we observed that the matching accuracy for local learning might be sufficient if K is chosen reasonably large. Let us now investigate the resulting classification accuracy of exemplar-specific mid-level features for fine-grained recognition.

Experimental Setup: In Section 7.4.2.1, we evaluated the discovery of mid-level representations using the entire training dataset. Here, we use the identical experimental setup but perform all steps individually for every test image using only its K nearest neighbors for discovery and training. We refer to both settings as *global* and *local*, respectively. Due to superior results in previous evaluations, we only perform supervised bootstrapping. Results for varying sizes of K are shown in Fig. 7.55. A comparison to previously shown results is given in Table 7.14.

Table 7.14: *Classification accuracy of exemplar-specific mid-level features for fine-grained recognition on the CUB-2011 dataset.*

Approach	CUB-2011/14	CUB-2011/200
SVM + BoW (Wah et al., 2011b)	—	10.25%
Mid-level features (unsupervised), Section 7.4.2	54.01%	—
Mid-level features (supervised), Section 7.4.2	58.01%	39.35%
Local Mid-level features ($K = 150$)	63.95%	34.16%
Style-awareness Lee et al. (2013)	—	38.31%
PDL (Jia et al., 2013)	—	38.91%
Template learning (Yang et al., 2012)	—	43.67%
DPD (Zhang et al., 2013)	—	50.98%
POOF (Berg and Belhumeur, 2013)	70.10%	56.78%
Part annotation transfer, Section 7.4.1	73.40%	57.99%

Experimental Results: From the results, we can draw two conclusions. First of all, we observe that the quality of the nearest neighbor matching strongly influences the resulting classification accuracy. On the CUB-2011/14 dataset, the matching accuracy is already sufficient as shown in the last evaluation. For these scenarios, exemplar-specific representations can lead to a valuable boost in classification accuracy by focusing on relevant features to discriminate extremely similar individuals. However, the resulting accuracy is still inferior to discriminative feature learning (POOF) or human annotations to focus feature extraction (Part annotation transfer).

For less accurate matching results as on CUB-2011/200, classification accuracy decreases compared to global representations. Since 13.07% test examples did not contain at least a single example of the correct category among their K neighbors, these cases are guaranteed to result in wrong class estimates.

Summary: *In summary, exemplar-specific mid-level features can lead to improved discrimination abilities. However, the success of local learning heavily depends of the matching accuracy.*

7.4.3.3 Dimensionality of Learned Representations

Our introduced technique for the discovery of mid-level features initialized a set of detectors using results of an unsupervised seeding step. In contrast to previous work, we perform no selection of discriminative detectors. Thus, after several iterations of bootstrapping, all discovered patch detectors are used to encode

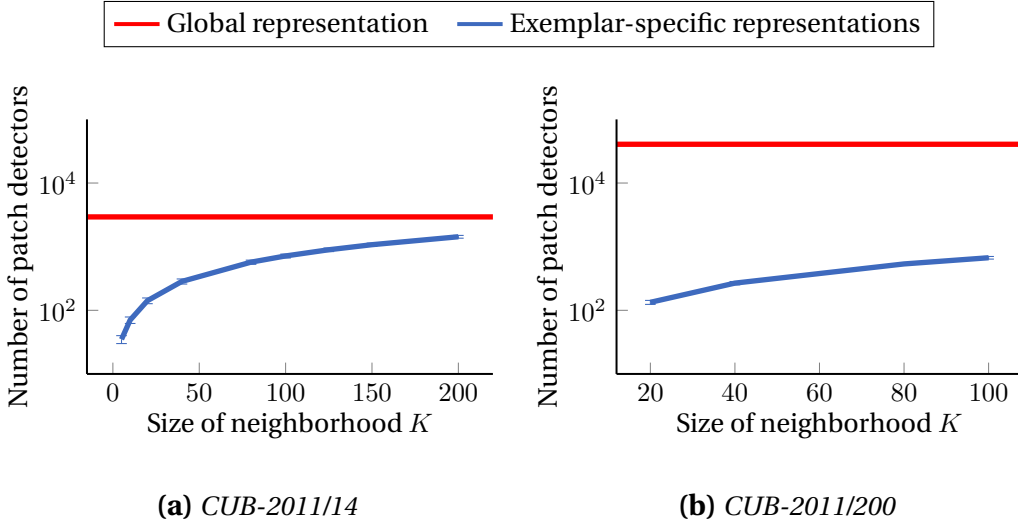


Figure 7.56: Number of discovered mid-level features for exemplar-specific representations as a function of the neighborhood size K . We plotted mean and standard deviation for different K . The dimensionality of fixed representations computed with all training examples serves as baseline (shown in red). Results are shown for the CUB-2011/14 subset (large) and the entire CUB-2011/200 dataset (right), respectively. Note that the y -axis is in log scale.

images. In consequence, the resulting dimensionality of learned representations increases linearly with the number of known training examples used during seeding. The previously analyzed local learning reduces the number of training examples to the K neighbors. Thus, we are interested in investigating the resulting compactness of learned representations.

Experimental Setup: The experimental setup is identical to the previous experiment. Instead of analyzing classification accuracies, we now investigate the number of discovered patch detectors individually for each test example. The number of detectors is finally averaged over all test instances. Results are shown in Fig. 7.56

Experimental Results: First of all, we note a clear decrease in the number of discovered patches when comparing global and local representations. As an example, the ratio between local and global representation is only 36.6% for the CUB-2011/14 dataset and $K = 150$. Thus, we reduced the dimensionality to one third and simultaneously increased the discrimination ability as shown in the previous experiment in Table 7.14.

On the full dataset, we noticed that local representations were not able to reach the classification accuracy of their global counterpart due to imperfect matching.

Nonetheless, the compactness of learned representations significantly increased. In fact, the required dimensionality with exemplar-specific features reduced to 1.7% for $K = 100$. Furthermore, we observe the aforementioned linear dependency between number of training examples and number of discovered patch detectors.

Summary: *In summary, exemplar-specific representations are clearly more compact than global representations.*

7.4.3.4 Activation Analysis for Varying Neighborhood Sizes

In addition to the quantitative results presented in the previous experiments, we are further interested in qualitative impressions.

Experimental Setup: We follow the activation analysis of Section 7.4.2.3 and visualize image parts where learned part detectors are most strongly activated. The entire setup for exemplar-specific discovery of mid-level features is identical to the previous experiments in this section. Activation maps for six test images are shown in Fig. 7.57.

Experimental Results: As we can see, the optimal neighborhood size varies across test examples. In some cases, already an extremely small number of $K = 5$ can be sufficient to capture relevant details. Two examples are given in the first and the last row. Also for increasing neighborhoods, the activation maps for these two images remain almost unchanged and clearly focus on the discriminative head or breast pattern. Note that in both cases, background structures do not negatively influence patch detectors since they are either uniform (top) or non-naturally regular (bottom).

In contrast to both examples, the bird shown in the second row requires an order of magnitude more neighbors to reliably discover relevant patterns. We attribute this effect to the bird's unusual pose which negatively influences the matching accuracy. Thus, the local neighborhood contains a larger variance of bird categories and discovered detectors are more diverse.

For the examples in rows three and four, we observe that the patches which are discovered for small neighborhoods correspond partly to intuitive regions. Nonetheless, their visual accuracy improves for larger neighborhood sizes.

Finally, the bird in row five can be best represented with detectors learned from a moderate number of neighbors. Here, larger neighborhoods lead to the discovery of background-specific detectors without an additional information gain.

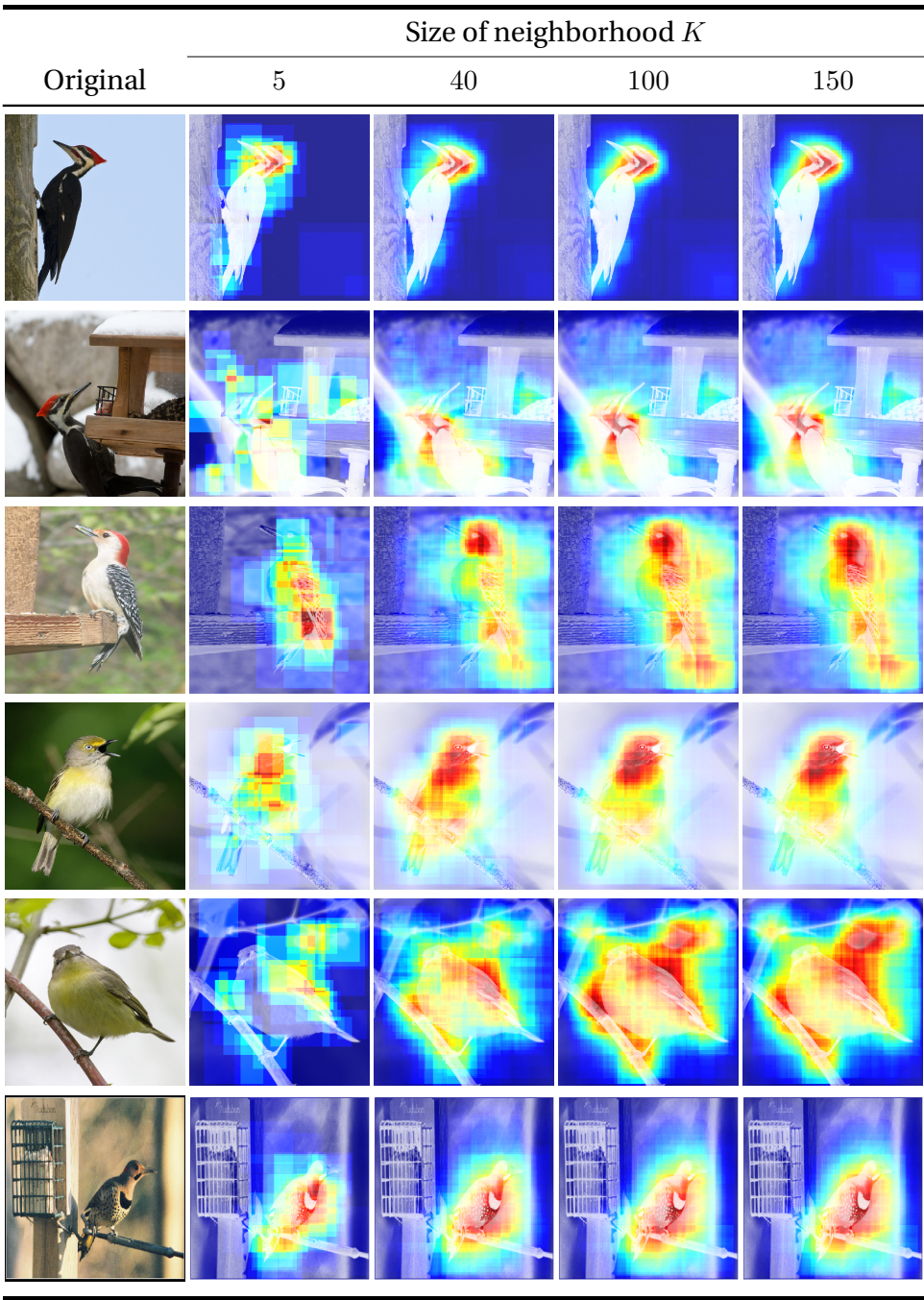


Figure 7.57: Detection responses of patch detectors for different numbers of neighbors. Activation maps are color coded using Matlab's *Jet colormap*. Warm colors indicate strong responses. Compare also against activations of mid-level features without local learning in Section 7.4.2.3, Fig. 7.53. Best viewed in color.

Summary: *In summary, exemplar-specific patch detectors often correspond to category-specific object parts. The optimal number of neighbors depends on view-point, pose, and background.*

7.4.4 Combining Mid-level Features and Semantic Parts

In the previous evaluations, we observed that mid-level features and exemplar-specific representations are well suited to capture relevant details in fine-grained recognition scenarios. However, we also observed several drawbacks, *e.g.*, the insufficient matching accuracy for large datasets. In consequence, we found that the final classification accuracy is inferior to discriminatively learned features or pipelines with localized feature extraction. Therefore, we are interested in combining our technique for the discovery of mid-level features with localized feature extraction based on human-annotated part positions as used in Section 7.4.1.

Experimental Setup: We combine both techniques using late fusion. In particular, we run both techniques independently to obtain two classification estimates. For the part annotation transfer, we use the identical setup as in the corresponding evaluation in Section 7.4.1. Regarding the discovery of mid-level features, we evaluate a global settings using all training images as well as the exemplar-specific variant with $K = 150$. All remaining settings are kept identical to the previous evaluations as described in Section 7.4.2 and Section 7.4.3, respectively. Classification scores obtained from the part annotation transfer pipeline and the mid-level features approach are individually normalized via soft-max to allow for comparable ranges (see (Duan et al., 2003)). Afterwards, we fuse both results by convex combination, which is also known as linear weighted fusion as reviewed in (Atrey et al., 2010). Note that the combination parameter $\lambda \in [0, 1]$ can be optimized based on leave-on-out estimation. Nonetheless, we show results for all possible values of λ to investigate the potential benefit of combining both approaches under optimal combination settings. The dependency between classification accuracy and combination parameter is shown in Fig. 7.58 for CUB-2011/14 and CUB-2011/200. Results of the optimal setting in comparison with competitive techniques are finally shown in Table 7.15.

Experimental Results: For the analysis regarding the combination parameter λ , we observe that all settings benefit from fusing both techniques with a suitable setting of λ . Furthermore, possible combination parameters which improve over classification accuracies of both baselines cover a wide range. Thus, from a practical point of view, computing an estimate for the optimal working point is likely possible. The only notable exception is the combination with exemplar-specific features on

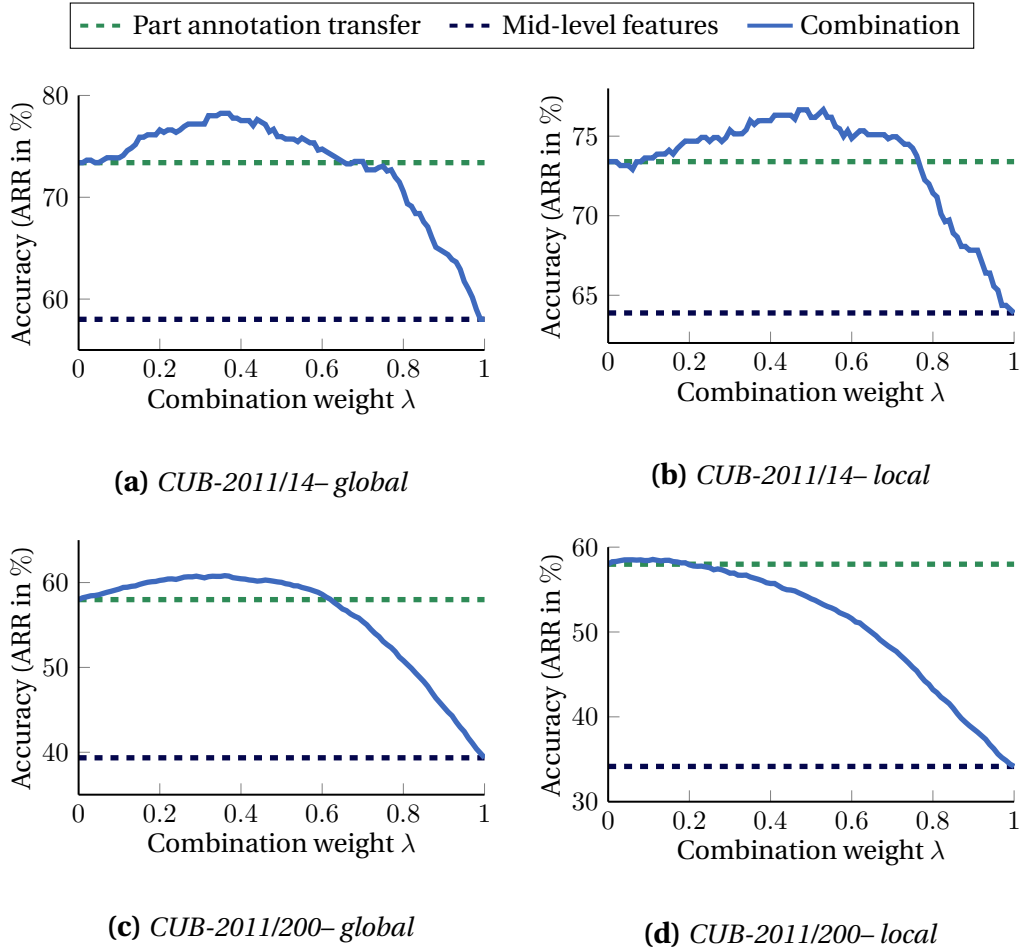


Figure 7.58: Analyzing the combination of mid-level features and part annotation transfer for fine-grained recognition on the CUB-2011 dataset. Results are shown for different values of the combination parameter λ . In four scenarios (global versus exemplar-specific and CUB-2011/200 versus CUB-2011/14), we investigate the dependency between combination parameter and classification accuracy. See Fig. 7.58 for further details.

Table 7.15: *Combining mid-level features and part annotation transfer for fine-grained recognition on the CUB-2011 dataset. Results are obtained for optimal combination parameter λ . See Fig. 7.58 for the dependency between combination parameter and classification accuracy for all scenarios.*

Approach	CUB-2011/14	CUB-2011/200
Dataset baseline (Wah et al., 2011b)	—	10.25%
Mid-level features (unsupervised), Section 7.4.2	54.01%	—
Mid-level features (supervised), Section 7.4.2	58.01%	39.35%
Local Mid-level features ($K = 150$), Section 7.4.3	63.95%	34.16%
Style-awareness Lee et al. (2013)	—	38.31%
PDL (Jia et al., 2013)	—	38.91%
Template learning (Yang et al., 2012)	—	43.67%
DPD (Zhang et al., 2013)	—	50.98%
POOF (Berg and Belhumeur, 2013)	70.10%	56.78%
Part annotation transfer, Section 7.4.1	73.40%	57.99%
Part annotation transfer, Section 7.4.1		
+ Mid-level features (global) Section 7.4.2	78.25%	60.81%
Part annotation transfer, Section 7.4.1		
+ Local Mid-level features ($K = 150$), Section 7.4.3	76.64%	58.55%
CNN + part localization (Zhang et al., 2014)	—	76.37%
CNN + part constellation (Simon and Rodner, 2015)	—	81.01%

the large CUB-2011/200 dataset (bottom right plot). Here, the accuracy of patch discovery is too low due to the inaccurate matching as analyzed in Section 7.4.3. Nonetheless, we still observe minor improvements by fusing scores.

In Table 7.15, we finally present all results obtained from the set of techniques which we evaluated in this section. As we can see, combining our discovery process of mid-level features with transfer results from human annotations is able to further improve over existing results. It even improves over the discriminatively learned POOF features by a significant margin. Nonetheless, latest results based on deep architectures are still several steps ahead. We thus conclude that transferring the presented techniques to latest feature extraction architectures is beneficial for future research.

Summary: *In summary, fusing mid-level features and semantic parts adds valuable information and is able to further improve classification accuracy in fine-grained recognition scenarios.*

8 | Conclusions

“Every end is a new beginning.”

In the previous chapters, we introduced several techniques for various aspects of lifelong learning and validated their practical applicability. We will now summarize the main contributions regarding each topic in Section 8.1. Furthermore, we will point to open questions and remaining problems in Section 8.2.

8.1 Summary and Thesis Contributions

In this thesis, we tackled the problem of lifelong learning for visual recognition systems. Our general aim was to build systems which continuously increase their knowledge by exploiting large labeled datasets, by interactively dealing with incoming data streams and experts-in-the-loop, and by handling and representing initially unknown data. The presented aspects are build on several peer-reviewed publications on international venues which are further extended in this thesis.

Introduction (Chapter 1): As an introduction to the topic of this thesis, we started with a motivation for image understanding in general. We highlighted drawbacks of the frequently used static, closed-world assumption and briefly introduced our lifelong learning cycle as a possible solution. Finally, we outlined the structure of this thesis which aims at solving several aspects that arise in the context of lifelong learning.

Basics of Machine Learning, Visual Recognition, and Lifelong Learning (Chapter 2): For a better understanding of present techniques in subsequent chapters, we introduced established concepts of machine learning and computer vision in Chapter 2. We started with a detailed overview on machine learning from frequentist’s and Bayesian’s perspectives in Section 2.1. From these general concepts, we derived a set of classification techniques and analyzed their theoretical properties.

For an intuitive understanding of involved assumptions and limitations, we presented visual showcases of each technique. We finally introduced Gaussian process models for regression and classification which built the foundation for the majority of subsequent chapters.

The presented algorithms for machine learning were formulated as generic problem solvers. For applying these algorithms to computer vision problems, we reviewed a variety of image representations in Section 2.2. The introduced techniques ranged from statistics of low-level features over activations from deep-learning architectures to semantically rich descriptions (*e.g.*, parts and attributes).

In Section 2.3, we finally presented existing approaches and research areas which are related to lifelong learning as covered by this thesis. We put these fields into perspective and highlighted missing aspects for successful realizations of the lifelong learning cycle.

Efficient Gaussian Process Models with Histogram Intersection Kernels (Chapter 3): Learning from large amounts of labeled data is one key challenge for lifelong learning systems. We therefore reviewed existing work on large-scale machine learning in Section 3.1. For the choice of Gaussian process label regression models, we presented techniques which substantially reduce computational limitations for all involved aspects of inference. Our techniques build on properties of generalized histogram intersection kernels and are denoted with GP/HIK.

In Section 3.2, we introduced the first set of techniques to allow for efficient model *training and classification*. Thereby, asymptotic computation times for training can be reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(N \log N)$ with respect to the number of training examples N . Furthermore, computation times for classification are reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(\log N)$. In addition, memory requirements scale only with $\mathcal{O}(N)$ instead of $\mathcal{O}(N^2)$. We can thereby overcome common complexity drawbacks of kernel-based methods. Finally, we showed that computation times can be further reduced to $\mathcal{O}(1)$ if test inputs are quantized in every dimension. We analyzed the resulting error, derived two bounds thereof, and proved their correctness.

An important property of GP models is the possibility to find useful parameter values for hyperparameters based on marginal likelihood maximization. We extended our previous techniques to also allow for efficient computations of the *marginal likelihood* in Section 3.3. Our techniques build on an efficiently computable upper bound for the determinant of the kernel matrix which we introduced and proved. In consequence, hyperparameter optimization scales only with $\mathcal{O}(N)$ instead of $\mathcal{O}(N^3)$.

An additional benefit of GP models is the availability of classification uncertainties in form of predictive variance estimates. In Section 3.4, we presented

techniques to efficiently compute or approximate the *predictive variance* using GP/HIK. Resulting computation times are reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, $\mathcal{O}(\log N)$, or even $\mathcal{O}(1)$. Which technique to use depends on the desired accuracy of the approximation.

We finally presented a simple framework for *semantic segmentation* in Section 3.5. This application scenario showcases the applicability of scalable GP models and the resulting benefit of replacing linear with non-linear classifiers in visual recognition tasks.

Incremental Learning (Chapter 4): Lifelong learning systems will inevitably face previously unseen data which needs to be incorporated. After a review of state of the art on incremental learning (Section 4.1), we presented techniques to incrementally train Gaussian process label regression models in their standard formulation (Section 4.2). We further showed how to extend our techniques for GP/HIK to incorporate new data efficiently (Section 4.3). Finally, we introduced and proved update techniques for the weight vector α as required for active learning (Section 4.2.3). All techniques resulted in closed-form solutions due to the inherent properties of GP models.

Active Learning (Chapter 5): Even with efficient methods for model training and updating, provided training data needs to be manually labeled. For increasing sets of observations and large-scale recorded data, this is obviously infeasible. To compensate this drawback, we introduced active learning in Section 5.1 which aims at selecting only the most informative examples for annotation. We gave a comprehensive review of the state of the art and showed connections to the science of human education.

Based on our efficient GP/HIK techniques, we showed in Section 5.3 how to *efficiently* evaluate existing active learning criteria for GP models. After analyzing drawbacks of these criteria, we introduced *estimated model changes* (EMC) as novel criterion for active learning in Section 5.4. For the choice of GP models, we derived efficient evaluations of this strategy. Furthermore, our analysis showed that the commonly used density-based re-weighting can be interpreted as an approximation of the EMC criterion. For simplicity, we mainly considered active learning criteria for binary classification models.

The EMC criterion still lacked a proper connection to the desired reduction of empirical risk. We therefore introduced a new general technique for active learning based on *estimated model output changes* (EMOC) in Section 5.5. As theoretical connection, we showed that EMOC is an upper bound for estimated risk minimization. Again, we derived efficient evaluations for GP models.

In Section 5.6, we investigated *multi-class scenarios* as required for lifelong

learning. We started by showing how to generalize established binary techniques to this scenario. Furthermore, we derived multi-class variants from our estimated model change and estimated model output change approaches. We highlighted that additional care needs to be taken regarding the estimation of class probabilities as well as suitable loss functions.

Finally, we introduced the problem of *unnameable instances* in active learning scenarios. A central observation is that annotators can refuse manual annotation. Exemplary reasons are that the requested example does not show a valid object (non-categorical rejection) or that the object is unknown to the annotator (categorical rejections). For both cases, we presented extensions of our EMOC framework to additionally handle the possibility of rejected annotations.

Mid-level Features (Chapter 6): In lifelong learning applications, object representations are required which allow for robust discrimination of object categories over time. We therefore reviewed established techniques for learning representations from data in Section 6.1. Furthermore, we introduced fine-grained recognition as application scenario for evaluating learned representations.

Driven by the insight that objects can be described by the presence or absence of distinctive parts, we presented a technique for discovering mid-level features in Section 6.2. Features can be discovered in a supervised or an unsupervised manner. Our approach groups re-occurring patterns by learning patch detectors and evaluating them using discrete convolutions.

In Section 6.3, we extended this approach to learn exemplar-specific representations. For every unseen object, the most similar known objects serve for feature learning and encoding. Thereby, learned representations are compact and specifically tailored to each new object.

Finally, we presented in Section 6.4 how part-annotations provided by human experts can be exploited to focus feature extraction on semantically important object regions. Since annotations are difficult to acquire, we presented a simple yet effective non-parametric method to transfer annotations from training images to test images.

Experimental Evaluation (Chapter 7): For all techniques introduced in the previous chapters, we presented evaluations in several application areas. Evaluations were conducted qualitatively and quantitatively to allow for intuitive and reliable conclusions. We made all developed source code publicly available to allow for reproducibility of results.

In the first block of experiments, we evaluated the resulting benefits of our GP/HIK techniques. For a wide range of application scenarios, we showed the applicability of GP/HIK to dataset sizes which are not possible to handle with GP

models in plain formulations. Experiments were conducted in binary and multi-class settings and covered image categorization, scene recognition, and semantic segmentation.

The second block of experiments was devoted to active learning scenarios. We analyzed properties of a variety of techniques for binary and multi-class settings and investigated their applicability in artificial and real-world problems. Our analysis underlined the superiority of our presented techniques which are based on estimated model output changes.

In the third and final block of experiments, we analyzed mid-level features for fine-grained recognition scenarios. We observed that accurate part localization classification accuracy heavily benefits from accurately estimated part positions. Furthermore, our techniques for (un)supervised patch discovery are adequate for capturing discriminative object details.

The results of all three parts confirm the suitability of our introduced techniques for several aspects of lifelong learning systems. Additionally we highlighted limitations of current approaches. In summary, we obtained insights for the development of next generation algorithms for lifelong learning in visual recognition tasks.

8.2 Future Work and Open Questions

The area of lifelong learning is relatively new and offers a variety of challenging research questions. Since the current thesis could by far not solve all aspects, a multitude of work still remains to be done and questions to be answered. In addition, a variety of questions arises from this thesis which are likely worth being further investigated. In the following, we list some of these questions and provide possible ideas for future research.

Future Work on Lifelong Learning

Integrating all Developed Aspects for an Application-specific Prototype: This thesis focused on fundamental research questions for lifelong learning systems. Although we provided contributions to a variety of involved aspects, we did not combine all techniques into a single coherent system. This shortcoming was due to the orientation on fundamental research. In particular, there was no specific necessity for a coherent realization since cooperation partners in applied sciences were missing. However, latest cooperations with ecologists and biodiversity researchers (Beckstein et al., 2014; Freytag et al., 2015; Jaeger et al., 2015; Rodner et al., 2015b) open the door to a variety of application areas. Implementing lifelong learning systems in these research fields will be the next promising step with potentially huge insights and new challenges.

A Framework for Evaluating Lifelong Learning Systems: In lifelong learning scenarios, there is no single fixed set of categories. Instead, new categories become known over time and old categories might vanish. Not even the number of all possible categories is known or fixed. Consequently, standard metrics for evaluating object classification systems as introduced in Section 7.1.2.1 are not perfectly feasible. Furthermore, it is unclear if reliable classifications of few known categories are preferable over knowing more categories and being less accurate. Finally, the order of discovered and learned categories can differ between systems, which makes it even harder to compare them.

We thus require an evaluation framework which reflects these issues and unites them into a coherent evaluation metric. A starting point can be a weighted combination of multiple criteria, *e.g.*, the number of known categories, the classification accuracy relative to known categories, and the accuracy regarding an artificial super-set of categories as shown in our recent paper (Käding et al., 2016).

Build and Extend Class Hierarchies: For the simplicity of presentation, we ignored taxonomic relations between categories in this thesis. Instead, we assumed that class labels are exclusive and flat. However, organizing categories hierarchically by grouping similar classes to subgroups is intuitively useful, especially for increasing number of known categories. Existing techniques build on language-oriented taxonomies such as WordNet (Fellbaum, 1998). We believe that classification with fixed label hierarchies is a great first step, *e.g.*, see (Deng et al., 2012) or (Deng et al., 2014). Nonetheless, it ultimately remains to increase and adapt appropriate hierarchies over time if changes of the environment become known to a lifelong learning system.

Future Work on Large-scale Gaussian Process Models

Efficient Computations of the Likelihood's Derivatives: While our introduced bound for the negative log likelihood allowed for efficient evaluation, it is not applicable to the likelihood's derivatives. In consequence, optimizing large numbers of hyperparameters is still cumbersome even with our efficient techniques. It remains an open issue to find similar approximations to the likelihood's (partial) derivative. Successful realizations would allow for efficient determination of feature relevance in high-dimensional feature spaces.

Exploiting Sophisticated Algebraic Solvers: Our GP/HIK techniques build on a variety of algebraic operations, *e.g.*, for solving linear equations, computing eigenvalues, or optimizing functions without access to gradients. In consequence, increased stability of results and reduced computation times can be expected by

replacing generic algebraic solvers by dedicated pendants. One example would be the application of sophisticated preconditioning schemes for iterative linear solvers (*e.g.*, see (Golub and Loan, 1996, Sect. 10.3) or (Saad, 2003, Chapter 9)). Similar considerations hold for the computation of eigenvalues (replace the Arnoldi iteration with more sophisticated Lanczos-routines) and the gradient-free optimization routines (replace the Nelder-Mead method with modern variants).

Future Work on Active Learning

Active Learning and Big Data: We observed that active learning is a valuable tool for training better models with less labeled data by selecting informative examples for manual annotation. Thus, we can greatly reduce annotation costs, especially if only few labeled data is initially available. However, initially labeled data can also be large in certain application scenarios. It thus remains to investigate the potential benefit of active learning for huge models trained from big datasets. As intuitive example, it is unclear whether a clever selection of a single new image which is added to ImageNet can lead to considerable changes. Intuitively, models trained from big data might only be improved by also asking big questions. A point to start can be the selection of multiple images at once, which is known as batch mode active learning (see (Settles, 2009, Sect. 6.1) and (Wei et al., 2015)). Alternatively, larger labeling budgets could scale applicability of active learning to big data problems, *e.g.*, by using crowd-sourcing techniques as done to create the MS-COCO dataset (Lin et al., 2014).

Exploiting Semantically Rich Expert Feedback: Within this thesis, we have been interested in asking questions about the content of selected images to then incorporate the obtained class labels. However, humans can provide much richer feedback. We should consequently exploited additional types of feedback to unleash the full potential of experts in the loop. Possible feedback can include part locations (Branson et al., 2014a), attributes (Kovashka et al., 2015), or explanations for provided feedback (Donahue and Grauman, 2011). While existing techniques show already promising results, adapting these techniques towards lifelong learning systems seems highly promising. Furthermore, the necessary type of feedback is likely to change over time. While initial questions will likely cover simple class levels, sophisticated questions for subtle information can be valuable in later stages of learning.

Combining Active Learning and Domain Adaptation: In lifelong learning scenarios, shifts in data distributions need to be expected over time. Two intuitive examples are continuous shifts due to changes in class appearances or discrete

shifts if gray-valued cameras are replaced with color cameras. For both cases, domain adaptation techniques (Hoffman et al., 2014) can adapt previously collected data or models to new domains. Supervised domain adaptation uses few labeled examples from the new domain to constrain the estimation of required adaptations. From the point of active learning, it remains an interesting questions to actively select examples of the new domain for annotation. A starting point could be the selection of examples which are expected to result in better adaptation processes.

Active Learning with Deep Neural Networks: Deep neural networks gained recent popularity in a variety of application scenarios. Especially for fine-tuning pre-trained networks to new domains, applying active learning seems intuitively beneficial. As starting point, we note that neural networks are large parametric models and usually allow for computing corresponding gradients. In consequence, applying the estimated model change or estimated model output change criteria is possible and likely worth being investigated.

Future Work on Feature Learning

Improving Details Of Patch Discovery: To apply our presented technique for (un)supervised patch discovery to larger datasets, several aspects require further attention. First of all, the bootstrapping of patch detectors could heavily benefit from faster evaluations of detectors. This could be realized by replacing the discrete convolutions with hashing techniques as shown by Dean et al. (2013) to obtain fast approximate bootstrapping results. In addition, the nearest neighbor search for exemplar-specific representations and part-annotation transfer would heavily benefit from improved matching results. Latest insights from image retrieval are thus to be transferred, *e.g.*, using current approaches which rely on CNN activations Babenko and Lempitsky (2015).

Patch Discovery and Deep Convolutional Neural Networks: The replacement of hand-crafted features with learned representations lead to impressive results in a large variety of applications. Thus, it seems worth to replace the currently used HOG and ColorName cell arrays and instead perform the discovery process on activation maps obtained from CNN architectures. Similar attempts have been proposed by Li et al. (2015) and showed promising results. Furthermore, unsupervised training of large neural networks is still an open question. It is interesting to apply our unsupervised discovery of patch detectors to serve as initialization for incrementally adding new layers to the networks architecture.

Joint Learning of Representations for Multiple Output Types: Especially in biodiversity scenarios, a multitude of different information is of interest for application

researchers. In particular, not only a reliable classification of objects is important, but also the prediction of attributes such as age, gender, height, or number of visible parts. It is therefore appealing to combine all tasks into a single coherent framework, which allows for joint learning of underlying representations in an end-to-end manner.

A | Contributions Beyond the Scope of This Thesis

While our group worked on lifelong learning systems, we naturally also spent time on several projects which will not be covered by this thesis. In the following, we will give a brief overview on those topics and list references to work we published within the last years.

Novelty Detection: Reliably flagging objects which we have never seen before as “novel” is as hard for machines as it is easy for humans. Let us just remember a child’s ability to ask its parents about objects within its daily environment (the ‘Dad - what’s that?’ situation mentioned at the beginning of this thesis). However, spotting novel objects is one of the most important aspects of lifelong learning systems. In fact, a successful realization can allow for deciding about whether or not the current knowledge needs to be actively enlarged and updated.

By definition, a main challenge for *novelty detection* methods is the absence of training data for previously unknown categories (otherwise, those classes would not be novel at all). Thus, standard classification techniques can not directly be applied. *One-class classification (OCC)* techniques approach this problem by modeling known data as stemming from a single class. In consequence, any violation to the learned distribution is considered as novel. In (Bodesheim et al., 2012), we introduced a technique for one-class classification based on information theory and approximated changes of model outputs (see Section 5.5 for a related approach and a detailed discussion). An overview is visualized in Fig. A.1.

For the majority of visual recognition tasks, training data usually represents a multitude of classes. Consequently, modeling all data as a single “known” super-class might be an assumption too strong for real-world applications. In (Bodesheim et al., 2013c), we empirically validated that such a single model leads to results which are worse than the combination of class-specific models or – being more sophisticated – a proper multi-class formulation. Following this observation, we introduced a multi-class novelty detection approach based on projections into null

spaces which maximize the Fisher criterion. Thereby, all training examples of a single class are mapped to a unique point while different classes are mapped to different points. As visualized in Fig. A.2, a new example is considered as novel if its distance in the null space to the nearest class projection is still large.

While it is often hard to find a universal model for deciding upon novelty, reducing the problem to smaller sub-tasks can circumvent this problem. Intuitively, it should be sufficient to compare a new test example only with the most similar objects seen so far to estimate its novelty. In (Bodesheim et al., 2015), we followed this observation and showed how local novelty detection models can significantly outperform fixed universal models. Thereby, models are learned individually for

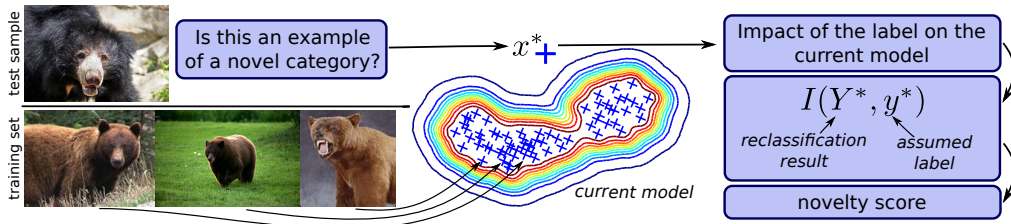


Figure A.1: Our proposed OCC approach presented in (Bodesheim et al., 2012): examples with largest estimated impact on the current model are considered as novel.

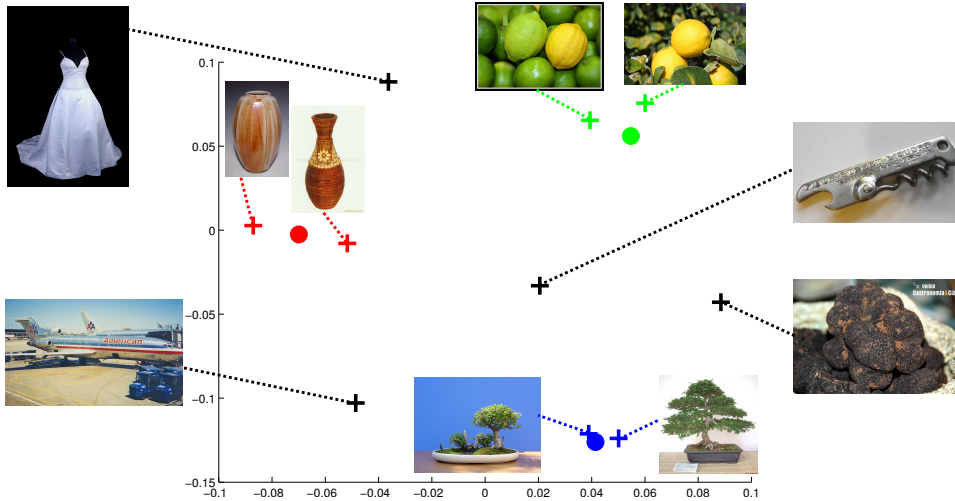


Figure A.2: Multi-class novelty detection using our approach introduced in (Bodesheim et al., 2013c). During training, a null space is computed such that all examples of a single class collapse to a unique point while different classes are mapped differently (colored dots are projections of three classes). At test time, examples of known classes are likely to be projected nearby (colored crosses), whereas novel objects reside far off in null space (black crosses).

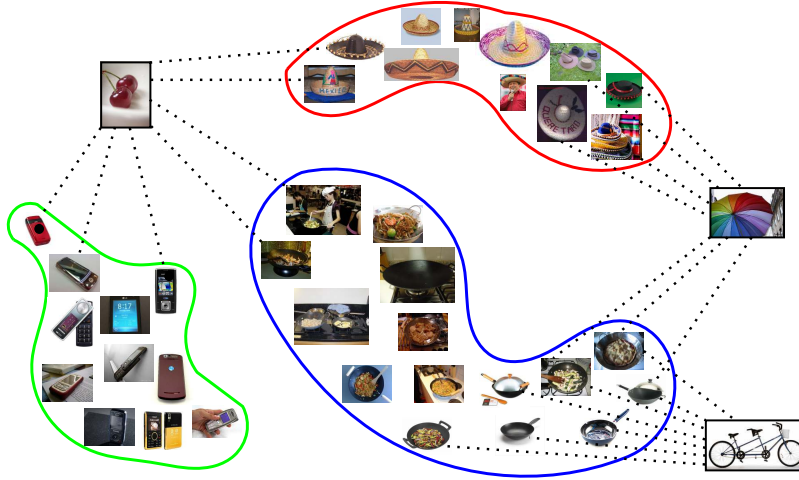


Figure A.3: Local novelty detection as presented in (Bodesheim et al., 2015). For each test input, only the most similar training examples contribute for training an exemplar-specific model to decide upon novelty.

each test input from a specifically determined set of highly similar training images. A visualization of this idea is given in Fig. A.3. Note the close connection to local learning and our exemplar-specific approaches as introduced in Section 6.3.

A thorough discussion on the importance of open-set recognition, *i.e.*, simultaneously performing object classification and novelty detection, is given in (Denzler et al., 2013). For a comprehensive review of related work in this area and a comparison of existing techniques, we refer the interested reader to (Bodesheim, 2016).

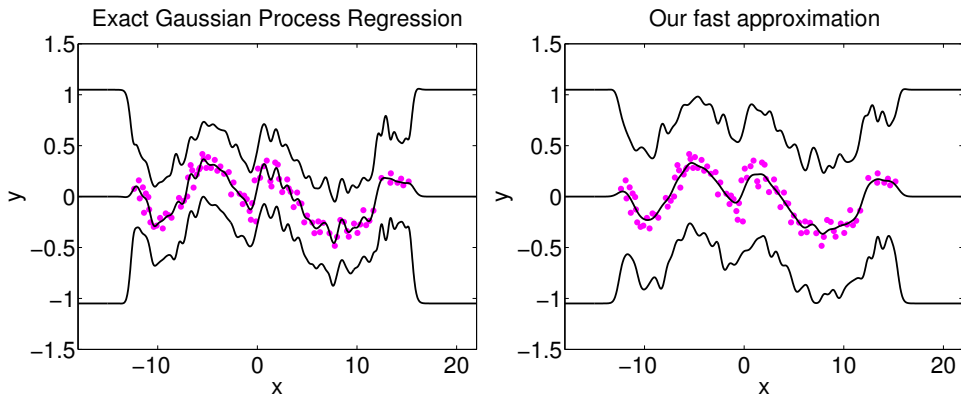


Figure A.4: Gaussian process regression based on the exact covariance matrix (left) and by using our proposed approximation (right) with a diagonal covariance matrix as presented in (Bodesheim et al., 2013b).

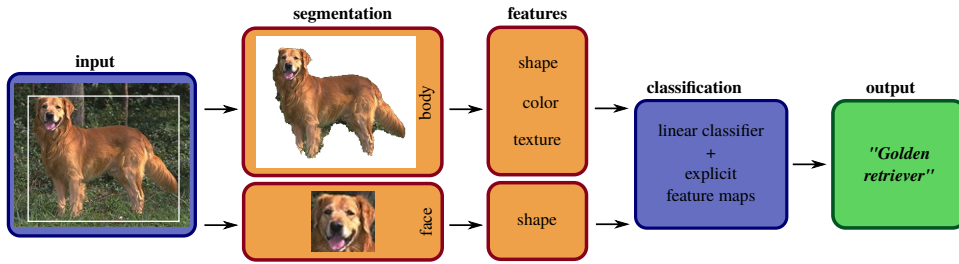


Figure A.5: Overview of our pipeline for fine-grained recognition as presented in (Göring et al., 2013).

Approximating Gaussian Process Regression: Learning examples over time is directly connected to the necessity of handling large amounts of data. Unfortunately, many of our prominent classification models are not *per se* designed to operate on thousands or even millions of known objects, due to both memory demand and computational complexity. Among others, this also holds for *Gaussian Process* regression models which are frequently used within this thesis. For these models, memory demand as well as training and evaluation time grow quadratic, cubic, and linear with the number of known examples, respectively. This effect is mainly due to the necessity of computing and inverting a large covariance matrix.

To circumvent this drawback, we showed in (Bodesheim et al., 2013b) how to find an approximating diagonal matrix with similar properties. While the resulting technique is significantly faster compared to the exact formulation, it leads to similar accuracy in applications such as novelty detection and active learning, as presented in (Bodesheim et al., 2013a). A visualization of the exact model and our approximation is given in Fig. A.4.

A Simple Pipeline for Fine-Grained Categorization: Standard image categorization tasks seek for coarse image labels specifying whether or not certain object classes are present within an image, *e.g.*, classes like car or person. In contrast to that, fine-grained recognition scenarios aim at solutions to reliably differentiate between semantic sub-categories (see also Section 6.1.4). Possible application scenarios are the discrimination among different dog breeds or bird species which usually are visually similar and only subtle details differ. In (Göring et al., 2013), we introduced a simple pipeline to tackle fine-grained recognition problems, which ranked third place (out of three participants) in the ImageNet Fine-grained Challenge 2012. An overview of our system is given in Fig. A.5.

Interactive Image Retrieval for Biodiversity Research: As for the approach of local novelty detection, it is often important to filter large image collection for instances which exhibit a strong visual similarity to new test inputs. This search

Select a relevant region!



Baseline image retrieval



After including expert-feedback

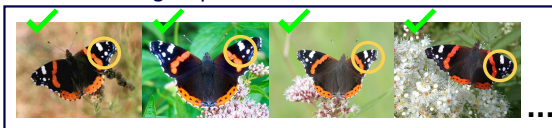


Figure A.6: Image retrieval aims at finding those images within a large collection which are visually most similar to new test inputs. A successful realization has a variety of possible application, e.g., in lifelong learning or biodiversity research (shown here): Collected datasets are filtered for individuals visually similar to an unseen object (left and top row). In (Freytag et al., 2015), we further present how an expert-in-the-loop can improve initial retrieval results. By selecting a must-have-region, e.g., the dashed rectangle, top-scored images are additionally verified to contain visually similar regions (indicated as circles in bottom row).

for visually similar and known instances is known as image retrieval. For more than twenty years, computer vision researcher aim at building smart, efficient, and powerful retrieval systems for a variety of application scenarios. Among others, a successful realization would be beneficial in our concept of lifelong learning systems.

Especially when a system classified an instance as novel, a retrieval system could present visually similar, known instances to the human-in-the-loop as possible sources for knowledge transfer. Furthermore, if the system is not certain about the example's novelty, even non-experts are likely able to verify the novelty by comparing the new instance against the visually most similar examples which are returned from a retrieval system.

In (Freytag et al., 2015), we presented a technique which further allows users to refine an initial retrieval result by specifying must-have-regions. Thereby, top-scored images are only selected if they contain parts which are visually similar to the specified query region. We specifically focused on biodiversity-like research scenarios, where experts collect and classify individuals to then evaluate current trends in ecological systems. Efficient suggestions of probable categories or similar individuals are highly beneficial in this context. A visualization is given in Fig. A.6. See also the corresponding video at <https://youtu.be/sQN8cNNxFjQ>.

Inverting Bag-of-visual-words Models: Within the last decade, our community often followed and adapted the bag-of-visual-words paradigm for representing images (see also Section 2.2.2). This paradigm suggests extracting a large number

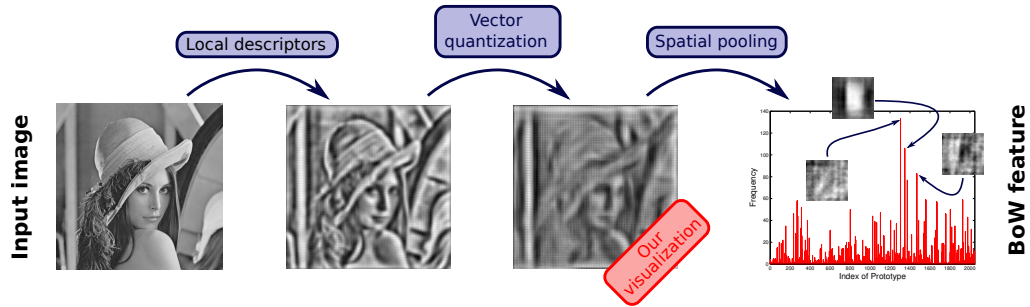


Figure A.7: Following the bag-of-visual-words paradigm, local features are vector quantized and statistics are formed by spatial pooling. Our technique presented in (Freytag et al., 2014d) visualizes the effects of quantization and thereby allows for visual inspection of codebooks used.

of features from small image regions which are then vector quantized into a pre-computed codebook. Finally, quantization statistics are formed by spatially pooling over the entire image or regions therein.

Since resulting image representations are usually hard to interpret, we analyzed in (Freytag et al., 2014d) how much visual information is actually discarded by quantization and pooling. Our method allows for visualizing the quality of a bag-of-words codebook. An overview is given in Fig. A.7.

B | Proofs

B.1 Quantization Error Analysis

Proof of Theorem 3.1. The quantization as introduced in Eq. (3.11) approximates a new example $\mathbf{x}_*$ with a quantized version $\tilde{\mathbf{x}}_*$. For that process, let q be the number of bins per dimension, *i.e.*, the number of different intervals each dimension is divided into. Furthermore, we assume permissible values of each dimension to be bounded $\mathbf{x}_*(d) \in [l_d, u_d]$. The maximum quantization error ε_q is then defined as the largest possible difference in any dimension between a permissible input and its quantized result:

$$\varepsilon_q = \max_{d \in \{1, \dots, D\}} \max_{\mathbf{x}_*(d) \in [l_d, u_d]} |\mathbf{x}_*(d) - \tilde{\mathbf{x}}_*(d)| . \quad (\text{B.1})$$

We are now interested in analyzing the relation between a given value for ε_q and the resulting difference between the exact predictive mean μ_* and its counterpart $\tilde{\mu}_*$ computed with quantized test inputs:

$$|\mu_* - \tilde{\mu}_*| = |\mathbf{k}_*^T \boldsymbol{\alpha} - \tilde{\mathbf{k}}_*^T \boldsymbol{\alpha}| \quad (\text{B.2})$$

$$= |(\mathbf{k}_* - \tilde{\mathbf{k}}_*)^T \boldsymbol{\alpha}| \quad (\text{B.3})$$

$$\leq \left| (\mathbf{k}_* - \tilde{\mathbf{k}}_*)^T \right| \cdot |\boldsymbol{\alpha}| , \quad (\text{B.4})$$

where $|v|$ returns the element-wise absolute value. For the ease of notation, we further define $\triangle \mathbf{k}_* = \mathbf{k}_* - \tilde{\mathbf{k}}_*$. Let us analyze this term in more detail under the

specific view of κ_{HIK} as kernel function:

$$|\Delta \mathbf{k}_*(i)| = |\kappa_{\text{HIK}}(\mathbf{x}_i, \mathbf{x}_*) - \kappa_{\text{HIK}}(\mathbf{x}_i, \tilde{\mathbf{x}}_*)| \quad (\text{B.5})$$

$$= \left| \sum_{d=1}^D \min(\mathbf{x}_i(d), \mathbf{x}_*(d)) - \min(\mathbf{x}_i(d), \tilde{\mathbf{x}}_*(d)) \right| \quad (\text{B.6})$$

$$\leq \sum_{d=1}^D |\min(\mathbf{x}_i(d), \mathbf{x}_*(d)) - \min(\mathbf{x}_i(d), \tilde{\mathbf{x}}_*(d))| \quad (\text{B.7})$$

$$= \sum_{d=1}^D |\Delta \min_*^i(d)|. \quad (\text{B.8})$$

Notably, the last step from Eq. (B.7) to Eq. (B.8) is severe and possibly loosens the bound since negative and positive terms could cancel in the sum of Eq. (B.7) but not in Eq. (B.8). Furthermore, the normalization of histogram features as inputs is also not taken into account. However, the step allows us to derive a very simple bound by analyzing a single term in the sum and the six cases, which occur with different orders of $\mathbf{x}_i(d)$, $\mathbf{x}_*(d)$, and $\tilde{\mathbf{x}}_*(d)$:

- (i) $\mathbf{x}_i(d) \leq \mathbf{x}_*(d) \leq \tilde{\mathbf{x}}_*(d) : |\Delta \min_*^i(d)| = 0$,
- (ii) $\mathbf{x}_*(d) \leq \mathbf{x}_i(d) \leq \tilde{\mathbf{x}}_*(d) : |\Delta \min_*^i(d)| = |\mathbf{x}_*(d) - \mathbf{x}_i(d)| \leq \varepsilon_q$,
- (iii) $\mathbf{x}_*(d) \leq \tilde{\mathbf{x}}_*(d) \leq \mathbf{x}_i(d) : |\Delta \min_*^i(d)| = |\mathbf{x}_*(d) - \tilde{\mathbf{x}}_*(d)| \leq \varepsilon_q$,
- (iv) $\tilde{\mathbf{x}}_*(d) \leq \mathbf{x}_*(d) \leq \mathbf{x}_i(d) : |\Delta \min_*^i(d)| = |\mathbf{x}_*(d) - \tilde{\mathbf{x}}_*(d)| \leq \varepsilon_q$,
- (v) $\tilde{\mathbf{x}}_*(d) \leq \mathbf{x}_i(d) \leq \mathbf{x}_*(d) : |\Delta \min_*^i(d)| = |\mathbf{x}_i(d) - \tilde{\mathbf{x}}_*(d)| \leq \varepsilon_q$,
- (vi) $\mathbf{x}_i(d) \leq \tilde{\mathbf{x}}_*(d) \leq \mathbf{x}_*(d) : |\Delta \min_*^i(d)| = 0$.

This directly leads to $|\Delta \mathbf{k}_*(i)| \leq D \cdot \varepsilon_q$ and combined with Eq. (B.4) to the final result of Theorem 3.1. $\square$

B.2 A Sharper Quantization Error Analysis

Proof of Theorem 3.2. In addition to the assumptions and definitions of the previous proof, we assume inputs $\mathbf{x}_*$ and their quantized pendants $\tilde{\mathbf{x}}_*$ to be normalized such that $\|\mathbf{x}_*\|_1 = \|\tilde{\mathbf{x}}_*\|_1 = 1$. Thus, we also assume $\mathbf{x}_*(d) \in [0, 1]$. Similar to Eq. (B.1), the maximum quantization error ε_q results to:

$$\varepsilon_q = \max_{d \in \{1, \dots, D\}} \max_{\mathbf{x}_*(d) \in [0, 1]} |\mathbf{x}_*(d) - \mathbf{x}_*(d)|. \quad (\text{B.9})$$

We again analyze the difference between the exact predictive mean μ_* and the one computed with the quantization trick $\tilde{\mu}_*$ as in Eq. (B.2):

$$|\mu_* - \tilde{\mu}_*| \leq \left| (\mathbf{k}_* - \tilde{\mathbf{k}}_*)^T \right| \cdot |\boldsymbol{\alpha}| . \quad (\text{B.10})$$

We now leave the idea of the previous proof and instead recall that the minimum operation can also be written as:

$$\min(a, b) = \frac{1}{2} \cdot (a + b - |a - b|) . \quad (\text{B.11})$$

Thereby, we can rewrite the first term in Eq. (B.10) as follows:

$$|\Delta \mathbf{k}_*(i)| = |\kappa_{\text{HIK}}(\mathbf{x}_i, \mathbf{x}_*) - \kappa_{\text{HIK}}(\mathbf{x}_i, \tilde{\mathbf{x}}_*)| \quad (\text{B.12})$$

$$= \left| \sum_{d=1}^D \min(\mathbf{x}_i(d), \mathbf{x}_*(d)) - \min(\mathbf{x}_i(d), \tilde{\mathbf{x}}_*(d)) \right| \quad (\text{B.13})$$

$$= \frac{1}{2} \left| \sum_{d=1}^D \mathbf{x}_i(d) + \mathbf{x}_*(d) - |\mathbf{x}_i(d) - \mathbf{x}_*(d)| \right. \\ \left. - \mathbf{x}_i(d) - \tilde{\mathbf{x}}_*(d) + |\mathbf{x}_i(d) - \tilde{\mathbf{x}}_*(d)| \right| \quad (\text{B.14})$$

$$= \frac{1}{2} \left| \sum_{d=1}^D \mathbf{x}_*(d) - \tilde{\mathbf{x}}_*(d) - |\mathbf{x}_i(d) - \mathbf{x}_*(d)| + |\mathbf{x}_i(d) - \tilde{\mathbf{x}}_*(d)| \right| . \quad (\text{B.15})$$

Let us now exploit the assumption that inputs and quantized counterparts are both normalized, thus, $\sum_{d=1}^D \mathbf{x}_*(d) - \tilde{\mathbf{x}}_*(d) = 0$. In consequence, we can shorten Eq. (B.15):

$$|\Delta \mathbf{k}_*(i)| = \frac{1}{2} \left| \sum_{d=1}^D |\mathbf{x}_i(d) - \tilde{\mathbf{x}}_*(d)| - |\mathbf{x}_i(d) - \mathbf{x}_*(d)| \right| \quad (\text{B.16})$$

$$= \frac{1}{2} \left| \|\mathbf{x}_i - \tilde{\mathbf{x}}_*\|_1 - \|\mathbf{x}_i - \mathbf{x}_*\|_1 \right| . \quad (\text{B.17})$$

Without loss of generality, let us now assume that $\|\mathbf{x}_i - \tilde{\mathbf{x}}_*\|_1 \geq \|\mathbf{x}_i - \mathbf{x}_*\|_1$ (the argumentation for the remaining case is symmetric). Thereby, we can apply the triangle inequality:

$$\|\mathbf{x}_i - \tilde{\mathbf{x}}_*\|_1 \leq \|\mathbf{x}_i - \mathbf{x}_*\|_1 + \|\mathbf{x}_* - \tilde{\mathbf{x}}_*\|_1 \quad (\text{B.18})$$

Combining Eq. (B.17) and Eq. (B.18), we finally obtain:

$$|\Delta \mathbf{k}_*(i)| \leq \frac{1}{2} (\|\mathbf{x}_i - \mathbf{x}_*\|_1 + \|\mathbf{x}_* - \tilde{\mathbf{x}}_*\|_1 - \|\mathbf{x}_i - \mathbf{x}_*\|_1) \quad (\text{B.19})$$

$$= \frac{1}{2} \|\mathbf{x}_* - \tilde{\mathbf{x}}_*\|_1 . \quad (\text{B.20})$$

This directly leads to $|\Delta \mathbf{k}_*(i)| \leq \frac{D \cdot \varepsilon_q}{2}$ and combined with Eq. (B.10) to the final result of Theorem 3.2. $\square$

B.3 An Upper Bound for a Matrix's Log-determinant

In Section 3.3.1, we approximated the Frobenius norm $|\mathbf{K}|_F^2$ with the sum of only the first M squared eigenvalues. Since we were interested in obtaining an upper bound of the log-determinant, it was not obvious that this bound was still valid when using a lower bound approximation for $|\mathbf{K}|_F^2$. To show the validity of the upper bound even with this approximation, we start with two small lemmas which will help to build the final proof. In addition, note that the bound is only well-defined for $\beta \neq \bar{t}$, since the 2×2 -matrix in Eq. (3.25) would be singular otherwise and its inversion not possible.

Lemma B.1 (Monotonicity of $\text{ub}(\beta, \mu_1, \mu_2)$ in Eq. (3.25) for $\beta = 1$). *Let $\mathbf{M} \in \mathbb{R}^{N \times N}$ be any positive definite matrix. Furthermore, let $\beta = 1$ be the largest eigenvalue of $\mathbf{M}$ and let denote $\mu_1 = \text{tr}(\mathbf{M})$ the trace of $\mathbf{M}$. Let finally $\tilde{\mu}_2$ with $0 < \tilde{\mu}_2 \leq \mu_2$ be a lower bound of the squared Frobenius norm $|\mathbf{K}|_F^2$ of $\mathbf{M}$, e.g., $\tilde{\mu}_2 = \sum_{i=1}^M \lambda_i^2$ with $M < N$. Then the following holds:*

$$\text{ub}(1, \mu_1, \tilde{\mu}_2) \geq \text{ub}(1, \mu_1, \mu_2) . \quad (\text{B.21})$$

Proof of Lemma B.1. Let us start by inspecting the bound given in Eq. (3.26) under the assumption of $\beta = 1$:

$$\text{ub}(1, \mu_1, \mu_2) = [\log 1, \log \bar{t}] \begin{bmatrix} 1 & \bar{t} \\ 1^2 & \bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.22})$$

$$= [0, \log \bar{t}] \begin{bmatrix} 1 & \bar{t} \\ 1 & \bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.23})$$

$$\boxed{\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}} = [0, \log \bar{t}] \frac{1}{\bar{t}^2 - \bar{t}} \begin{bmatrix} \bar{t}^2 & -\bar{t} \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.24})$$

$$= \log \bar{t} \frac{1}{\bar{t}(\bar{t} - 1)} (\mu_2 - \mu_1) \quad (\text{B.25})$$

$$\boxed{\text{definition of } \bar{t}} = \log \bar{t} \left(\frac{\mu_1 - \mu_2}{N - \mu_1} \left(\frac{\mu_1 - \mu_2}{N - \mu_1} - 1 \right) \right)^{-1} \cdot (\mu_2 - \mu_1) \quad (\text{B.26})$$

$$= \log \bar{t} \frac{(N - \mu_1)^2 (\mu_2 - \mu_1)}{(\mu_1 - \mu_2)(\mu_1 - \mu_2 - N + \mu_1)} \quad (\text{B.27})$$

$$= \log \bar{t} \frac{(N - \mu_1)^2}{N - 2\mu_1 + \mu_2} \quad (\text{B.28})$$

$$\boxed{\text{definition of } \bar{t}} = \log \left(\frac{\mu_1 - \mu_2}{N - \mu_1} \right) \frac{(N - \mu_1)^2}{N - 2\mu_1 + \mu_2} . \quad (\text{B.29})$$

According to the aim of Lemma B.1, let us now replace the squared Frobenius norm μ_2 in Eq. (B.29) with a lower bound $\tilde{\mu}_2$ such that $0 < \tilde{\mu}_2 \leq \mu_2$. Doing so, we notice that the first term in Eq. (B.29) increases while the denominator of the second term decreases:

$$\text{ub}(1, \mu_1, \mu_2) = (\text{B.29}) \quad (\text{B.30})$$

$$\leq \log \left(\frac{\mu_1 - \tilde{\mu}_2}{N - \mu_1} \right) \frac{(N - \mu_1)^2}{N - 2\mu_1 + \mu_2} \quad (\text{B.31})$$

$$\leq \log \left(\frac{\mu_1 - \tilde{\mu}_2}{N - \mu_1} \right) \frac{(N - \mu_1)^2}{N - 2\mu_1 + \tilde{\mu}_2} \quad (\text{B.32})$$

$$= \text{ub}(1, \mu_1, \tilde{\mu}_2) , \quad (\text{B.33})$$

which proves the lemma. $\square$

While the previous lemma for obtaining valid upper bounds by replacing the exact Frobenius norm with a smaller approximation was restricted to $\beta = 1$, we will now see that scaling the matrix M with $\gamma > 0$ (and therefore also its eigenvalues) leads to an additive constant in the bound which turns out to be independent of μ_1 and μ_2 .

Lemma B.2 (Multiplicative scaling of $\text{ub}(\beta, \mu_1, \mu_2)$ in Eq. (3.25)). *Let $M \in \mathbb{R}^{N \times N}$ be any positive definite matrix. Let further be β, μ_1 , and μ_2 suitable parameters as defined in the previous lemma. Then, for every positive factor $\gamma \in \mathbb{R}_+$, the following holds:*

$$\text{ub}(\gamma\beta, \gamma\mu_1, \gamma^2\mu_2) = \text{ub}(\beta, \mu_1, \mu_2) + N \log \gamma . \quad (\text{B.34})$$

As a side-note, we adhere that the additive constant is equivalent to the common constant for the log determinant of a $N \times N$ -matrix after scaling:

$$\det(\gamma \mathbf{A}) = \gamma^N \det(\mathbf{A}) \quad (\text{B.35})$$

$$\Leftrightarrow \log \det(\gamma \mathbf{A}) = N \log \gamma + \log \det(\mathbf{A}) . \quad (\text{B.36})$$

Thus, the quality of the bound is invariant with respect to γ .

Proof of Lemma B.2. For proving Lemma B.2, let us directly start with the definition of $\text{ub}(\beta, \mu_1, \mu_2)$ and insert $\gamma\beta, \gamma\mu_1, \gamma^2\mu_2$ instead. Note that μ_2 scales with γ^2 while β, μ_1 , and therefore also $\bar{t}$ are only affected by γ .

$$\text{ub}(\gamma\beta, \gamma\mu_1, \gamma^2\mu_2) = [\log \gamma\beta, \log \gamma\bar{t}] \begin{bmatrix} \gamma\beta & \gamma\bar{t} \\ \gamma^2\beta^2 & \gamma^2\bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \gamma\mu_1 \\ \gamma^2\mu_2 \end{bmatrix} \quad (\text{B.37})$$

$$= [\log \gamma\beta, \log \gamma\bar{t}] \cdot \left(\begin{bmatrix} \gamma & 0 \\ 0 & \gamma^2 \end{bmatrix} \begin{bmatrix} \beta & \bar{t} \\ \beta^2 & \bar{t}^2 \end{bmatrix} \right)^{-1} \begin{bmatrix} \gamma & 0 \\ 0 & \gamma^2 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.38})$$

$$= \left([\log \beta, \log \bar{t}] + [\log \gamma, \log \gamma] \right) \begin{bmatrix} \beta & \bar{t} \\ \beta^2 & \bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.39})$$

$$\boxed{\text{definition of ub}} = \text{ub}(\beta, \mu_1, \mu_2) + \underbrace{[\log \gamma, \log \gamma] \begin{bmatrix} \beta & \bar{t} \\ \beta^2 & \bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}}_{\doteq} \quad (\text{B.40})$$

$$= \text{ub}(\beta, \mu_1, \mu_2) + \overline{\text{ub}}_\gamma(\beta, \mu_1, \mu_2) . \quad (\text{B.41})$$

In the last step, we summarized the second part in an additional function $\overline{\text{ub}}_\gamma(\beta, \mu_1, \mu_2)$. It therefore remains to show that evaluating $\overline{\text{ub}}_\gamma(\beta, \mu_1, \mu_2)$ results in

$N \log \gamma$ to complete the proof:

$$\overline{\text{ub}}_\gamma(\beta, \mu_1, \mu_2) = [\log \gamma, \log \gamma] \begin{bmatrix} \beta & \bar{t} \\ \beta^2 & \bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.42})$$

$$= \log(\gamma) [1, 1] \begin{bmatrix} \beta & \bar{t} \\ \beta^2 & \bar{t}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.43})$$

$$\boxed{\text{see (B.24)}} = \log(\gamma) [1, 1] \frac{1}{\beta \bar{t}^2 - \beta^2 \bar{t}} \begin{bmatrix} \bar{t}^2 & -\bar{t} \\ -\beta^2 & \beta \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.44})$$

$$= \log(\gamma) \frac{1}{\beta \bar{t} (\bar{t} - \beta)} [\bar{t}^2 - \beta^2, \beta - \bar{t}] \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.45})$$

$$= \log(\gamma) \frac{1}{\beta \bar{t}} [\bar{t} + \beta, -1] \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (\text{B.46})$$

$$= \log(\gamma) \frac{1}{\beta \bar{t}} ((\bar{t} + \beta) \mu_1 - \mu_2) \quad (\text{B.47})$$

$$\boxed{\text{definition of } \bar{t}} = \log(\gamma) \frac{\beta N - \mu_1}{\beta^2 \mu_1 - \beta \mu_2} \left(\left(\frac{\beta \mu_1 - \mu_2 + \beta^2 N - \beta \mu_1}{\beta N - \mu_1} \right) \mu_1 - \mu_2 \right) \quad (\text{B.48})$$

$$\boxed{\text{cancel } (\beta N - \mu_1)} = \log(\gamma) \frac{\beta \mu_1^2 - \mu_2 \mu_1 + \beta^2 N \mu_1 - \beta \mu_1^2 - \mu_2 \beta N + \mu_2 \mu_1}{\beta^2 \mu_1 - \beta \mu_2} \quad (\text{B.49})$$

$$= \log(\gamma) \frac{\beta^2 N \mu_1 - \mu_2 \beta N}{\beta^2 \mu_1 - \beta \mu_2} \quad (\text{B.50})$$

$$= \log(\gamma) \frac{N (\beta^2 \mu_1 - \mu_2 \beta)}{\beta^2 \mu_1 - \beta \mu_2} \quad (\text{B.51})$$

$$= N \log \gamma . \quad (\text{B.52})$$

This proves the lemma. $\square$

We now have all building blocks at hand to obtain and proof the desired statement for Eq. (3.32) with only a few derivations.

Theorem B.3 (Upper bound with $\tilde{\mu}_2$). *Let $M \in \mathbb{R}^{N \times N}$ be any positive definite matrix. As in the previous lemmas, let μ_1 , μ_2 , and β denote the trace of M , the squared Frobenius norm of M , and an upper bound for the eigenvalues of M (e.g., the largest eigenvalue of M). If $\tilde{\mu}_2$ is a lower bound of μ_2 , the following holds:*

$$\log \det(M) \leq \text{ub}(\beta, \mu_1, \mu_2) \leq \text{ub}(\beta, \mu_1, \tilde{\mu}_2) . \quad (\text{B.53})$$

Proof of Theorem B.3. With the previously derived two lemmas, all we have to do for proving Theorem B.3 is to start with the original upper bound given by Bai and Golub (1997) in Lemma 3.3, and then apply Lemma B.2 with $\gamma = \frac{1}{\beta}$ followed by

Lemma B.1:

$$\log \det(\mathbf{M}) \leq \text{ub}(\beta, \mu_1, \mu_2) \quad (\text{B.54})$$

$$\boxed{\text{Lemma B.2}} = \text{ub}\left(1, \frac{1}{\beta}\mu_1, \frac{1}{\beta^2}\mu_2\right) - N \cdot \log\left(\frac{1}{\beta}\right) \quad (\text{B.55})$$

$$\boxed{\text{Lemma B.1}} \leq \text{ub}\left(1, \frac{1}{\beta}\mu_1, \frac{1}{\beta^2}\tilde{\mu}_2\right) - N \cdot \log\left(\frac{1}{\beta}\right) \quad (\text{B.56})$$

$$\boxed{\text{Lemma B.2}} = \text{ub}(\beta, \mu_1, \tilde{\mu}_2) \quad (\text{B.57})$$

□

B.4 Bounds on Quadratic Forms

Let $\mathbf{M} \in \mathbb{R}^{D \times D}$ be a real-valued, symmetric matrix. Thus, $\mathbf{M}$ can be transformed into $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ (Golub and Loan, 1996, Sect. 8.11) where $\mathbf{U} \in \mathbb{R}^{D \times D}$ is an orthonormal matrix (see also (Hermann, 2006, Sect. 2.5)). The matrix $\mathbf{D} \in \mathbb{R}^{D \times D}$ is a diagonal matrix containing all eigenvalues of $\mathbf{M}$, *i.e.*, $\mathbf{D}(i, j) = 0$ if $i \neq j$ and $\mathbf{D}(i, j) = \lambda_i$ if $i = j$. For any vector $\mathbf{x} \in \mathbb{R}^D$, we can then rewrite the quadratic form $\mathbf{x}^T \mathbf{M} \mathbf{x}$ as follows:

$$\mathbf{x}^T \mathbf{M} \mathbf{x} = \mathbf{x}^T \mathbf{U} \mathbf{D} \mathbf{U}^T \mathbf{x} = \tilde{\mathbf{x}}^T \mathbf{D} \tilde{\mathbf{x}} = \sum_{i=1}^D \lambda_i \tilde{\mathbf{x}}(i)^2 \quad (\text{B.58})$$

W.l.o.g., we can assume that the eigenvalues are ordered: $\lambda_1 \geq \dots \geq \lambda_D$. Note that $\tilde{\mathbf{x}}(i)$ is the projection of $\mathbf{x}$ onto the i^{th} eigenvector of $\mathbf{M}$. In consequence, we obtain the following upper bound approximation of the quadratic term:

$$\mathbf{x}^T \mathbf{M} \mathbf{x} = \sum_{i=1}^k \lambda_i \tilde{\mathbf{x}}(i)^2 + \sum_{i=k+1}^D \lambda_i \tilde{\mathbf{x}}(i)^2 \quad (\text{B.59})$$

$$\leq \sum_{i=1}^k \lambda_i \tilde{\mathbf{x}}(i)^2 + \lambda_{k+1} \sum_{j=k+1}^D \tilde{\mathbf{x}}(j)^2 \quad (\text{B.60})$$

Furthermore, $\mathbf{U}$ is orthonormal, thus, the length of vectors is invariant under $\mathbf{U}$'s transformation: $\|\mathbf{U}\mathbf{x}\| = \|\mathbf{x}\|$. We can use this aspect to further rewrite Eq. (B.60) and to obtain the final upper bound:

$$\mathbf{x}^T \mathbf{M} \mathbf{x} \leq \left(\sum_{i=1}^k \lambda_i \tilde{\mathbf{x}}(i)^2 \right) + \lambda_{k+1} \left(\|\mathbf{x}\|^2 - \sum_{j=1}^k \tilde{\mathbf{x}}(j)^2 \right) \quad (\text{B.61})$$

Note that a dual lower bound can be obtained by using the k smallest eigenvalues $\lambda_{D-k+1}, \dots, \lambda_D$ and replacing the remaining eigenvalues by λ_{D-k} :

$$\mathbf{x}^T \mathbf{M} \mathbf{x} \geq \left(\sum_{i=D-k+1}^D \lambda_i \tilde{\mathbf{x}}(i)^2 \right) + \lambda_{D-k} \left(\|\mathbf{x}\|^2 - \sum_{i=D-k+1}^D \tilde{\mathbf{x}}(j)^2 \right) . \quad (\text{B.62})$$

A noteworthy special case is $k = 0$, where we obtain the well-known bounds

$$\lambda_D \|\mathbf{x}\|^2 \leq \mathbf{x}^T \mathbf{M} \mathbf{x} \leq \lambda_1 \|\mathbf{x}\|^2 . \quad (\text{B.63})$$

which hold for arbitrary positive definite matrices $\mathbf{M} \in \mathbb{R}^{D \times D}$ and vectors $\mathbf{x} \in \mathbb{R}^D$.

B.5 Closed Form Updates of Weight Vector α for GP Regression

Preconditions of Theorem 4.2: Let y_* be the class label of a new example $\mathbf{x}_*$. Let further $\mathbf{K} = \kappa(\mathcal{X}, \mathcal{X})$ be the kernel matrix of all training examples seen so far, $\mathbf{k}_* = \kappa(\mathcal{X}, \{\mathbf{x}_*\})$ is the vector of kernel values between the new example and all training examples, and $k_{**} = \kappa(\mathbf{x}_*, \mathbf{x}_*)$ is the self-similarity value of $\mathbf{x}_*$. In addition, we denote with $\mathbf{y}$ the vector of labels for all training samples, and we include the noise parameter σ_n^2 for model regularization within the covariance function as described in Section 2.1.5.7. As in Section 2.1.5, we follow a zero-mean assumption. Finally, let $\alpha = \mathbf{K}^{-1} \mathbf{y}$ be the weight vector for GP regression.

Proof of Theorem 4.2: For the sake of readability with respect to long formulas, we will set the page orientation to landscape for the entire proof. Thus, the proof starts on the next page.

Proof of Theorem 4.2.

$$\begin{aligned}
 \alpha' &= K'^{-1} \begin{bmatrix} y \\ y_* \end{bmatrix} \\
 &= \begin{bmatrix} K & k_* \\ k_*^T & k_{**} \end{bmatrix}^{-1} \begin{bmatrix} y \\ y_* \end{bmatrix} \\
 &\stackrel{\text{(Bernstein, 2009, Eq. (2.8.18), p. 117)}}{=} \begin{bmatrix} \left(K - \frac{1}{k_{**}} k_* k_*^T \right)^{-1} & - \left(K - \frac{1}{k_{**}} k_* k_*^T \right)^{-1} \frac{1}{k_{**}} k_* \\ - (k_{**} - k_*^T K^{-1} k_*)^{-1} k_*^T K^{-1} & (k_{**} - k_*^T K^{-1} k_*)^{-1} \end{bmatrix} \begin{bmatrix} y \\ y_* \end{bmatrix} \\
 &\stackrel{\text{Woodbury's formula, Eq. (4.4)}}{=} \begin{bmatrix} K^{-1} + K^{-1} k_* (k_{**} - k_*^T K^{-1} k_*)^{-1} k_*^T K^{-1} & - (K^{-1} + K^{-1} k_* (k_{**} - k_*^T K^{-1} k_*)^{-1} k_*^T K^{-1}) \frac{1}{k_{**}} k_* \\ - (k_{**} - k_*^T K^{-1} k_*)^{-1} k_*^T K^{-1} & (k_{**} - k_*^T K^{-1} k_*)^{-1} \end{bmatrix} \begin{bmatrix} y \\ y_* \end{bmatrix} \\
 &\stackrel{\text{Expand, use definition of } \alpha \text{ and GP variance}}{=} \begin{bmatrix} \alpha + K^{-1} k_* \frac{1}{\sigma_*^2} k_*^T \alpha - \left(K^{-1} + K^{-1} k_* \frac{1}{\sigma_*^2} k_*^T K^{-1} \right) \frac{y_*}{k_{**}} k_* \\ - \frac{1}{\sigma_*^2} k_*^T \alpha + \frac{y_*}{\sigma_*^2} \end{bmatrix} \\
 &\stackrel{\text{Detach old } \alpha}{=} \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} k_* \frac{1}{\sigma_*^2} k_*^T \alpha - \frac{y_*}{k_{**}} \left(K^{-1} + K^{-1} k_* \frac{1}{\sigma_*^2} k_*^T K^{-1} \right) k_* \\ - \frac{1}{\sigma_*^2} k_*^T \alpha + \frac{y_*}{\sigma_*^2} \end{bmatrix} \\
 &\stackrel{\text{Simplify 2nd entry}}{=} \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} k_* \frac{1}{\sigma_*^2} k_*^T \alpha - \frac{y_*}{k_{**}} \left(K^{-1} + K^{-1} k_* \frac{1}{\sigma_*^2} k_*^T K^{-1} \right) k_* \\ \frac{1}{\sigma_*^2} (y_* - k_*^T \alpha) \end{bmatrix} \\
 &\stackrel{\text{Expand enumerator with } k_*}{=} \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} k_* \frac{1}{\sigma_*^2} k_*^T \alpha - \frac{y_*}{k_{**}} \left(K^{-1} k_* + K^{-1} k_* \frac{1}{\sigma_*^2} k_*^T K^{-1} k_* \right) \\ \frac{1}{\sigma_*^2} (y_* - k_*^T \alpha) \end{bmatrix} \\
 &= (*)
 \end{aligned}$$

□

$$\begin{aligned}
 (*) &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} \mathbf{k}_* \frac{1}{\sigma_*^2} \mathbf{k}_*^T \alpha - \frac{y_*}{k_{**}} \left(K^{-1} \mathbf{k}_* + K^{-1} \mathbf{k}_* \frac{1}{\sigma_*^2} \mathbf{k}_*^T K^{-1} \mathbf{k}_* \right) \\ \frac{1}{\sigma_*^2} (y_* - \mathbf{k}_*^T \alpha) \end{bmatrix} \\
 \text{Factor out } K^{-1} \mathbf{k}_* \text{ in enumerator} &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} \mathbf{k}_* \left(\frac{1}{\sigma_*^2} \mathbf{k}_*^T \alpha - \frac{y_*}{k_{**}} \left(1 + \frac{1}{\sigma_*^2} \mathbf{k}_*^T K^{-1} \mathbf{k}_* \right) \right) \\ \frac{1}{\sigma_*^2} (y_* - \mathbf{k}_*^T \alpha) \end{bmatrix} \\
 \text{Definition of GP variance} &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} \mathbf{k}_* \left(\frac{1}{\sigma_*^2} \mathbf{k}_*^T \alpha - \frac{y_*}{k_{**}} \left(1 + \frac{k_{**} - \sigma_*^2}{\sigma_*^2} \right) \right) \\ \frac{1}{\sigma_*^2} (y_* - \mathbf{k}_*^T \alpha) \end{bmatrix} \\
 \text{Expand 1 in enumerator} &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} \mathbf{k}_* \left(\frac{1}{\sigma_*^2} \mathbf{k}_*^T \alpha - \frac{y_*}{k_{**}} \left(\frac{\sigma_*^2}{\sigma_*^2} + \frac{k_{**} - \sigma_*^2}{\sigma_*^2} \right) \right) \\ \frac{1}{\sigma_*^2} (y_* - \mathbf{k}_*^T \alpha) \end{bmatrix} \\
 \text{Merge denominators} &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} \mathbf{k}_* \left(\frac{1}{\sigma_*^2} \mathbf{k}_*^T \alpha - \frac{y_*}{k_{**}} \left(\frac{k_{**}}{\sigma_*^2} \right) \right) \\ \frac{1}{\sigma_*^2} (y_* - \mathbf{k}_*^T \alpha) \end{bmatrix} \\
 \text{Simplify enumerator} &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} K^{-1} \mathbf{k}_* \left(\frac{1}{\sigma_*^2} \mathbf{k}_*^T \alpha - \frac{y_*}{\sigma_*^2} \right) \\ \frac{1}{\sigma_*^2} (y_* - \mathbf{k}_*^T \alpha) \end{bmatrix} \\
 \text{Factor out } \frac{1}{\sigma_*^2} &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \frac{1}{\sigma_*^2} \begin{bmatrix} K^{-1} \mathbf{k}_* (\mathbf{k}_*^T \alpha - y_*) \\ (y_* - \mathbf{k}_*^T \alpha) \end{bmatrix} \\
 \text{Factor out } \mathbf{k}_*^T \alpha - y_* &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \frac{\mathbf{k}_*^T \alpha - y_*}{\sigma_*^2} \begin{bmatrix} K^{-1} \mathbf{k}_* \\ -1 \end{bmatrix} \\
 \text{Definition of GP mean} &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \frac{(\mu_* - y_*)}{\sigma_*^2} \begin{bmatrix} K^{-1} \mathbf{k}_* \\ -1 \end{bmatrix}
 \end{aligned}$$

C | Overview of Experimental Findings

C.1 Large-scale GP Models: Summary of Findings

GP Label Regression versus GP Classification (Section 7.2.1)

Binary Classification: GP label regression is on par or even superior to its theoretically sound competitors for binary classification tasks.

Multi-class Classification: One-vs-all GP label regression is well applicable as multi-class GP classification technique due to three reasons: it is easy to implement, it is fast to compute, and classification results are comparable to those of competing techniques.

Large-scale Classification with GP/HIK (Section 7.2.2)

Binary Classification Scenarios: GP/HIK is significantly faster than the baseline GP for training and classification in practical applications. Since the required memory is additionally reduced, it allows for learning on large-scale datasets which are otherwise intractable for exact GP inference.

Multi-class Classification Scenarios: GP/HIK provides a useful alternative to established SVM solvers in multi-class classification scenarios with large numbers of training examples and categories.

GP/HIK versus SVM on CNN activations: GP/HIK is well suited as classification model on top of CNN activations.

Iterative Linear Solvers for GP/HIK: The CG method can draw best advantage from our techniques for fast *matrix vector multiplications*.

Early Stopping of the Linear Solver: Early stopping of the iterative linear solver can significantly save computation time without negative implications on learned models.

Comparison with GP Sparsity Methods: Our techniques for exact GP inference outperform sparse GP approximation techniques.

Evaluation of the Quantization: The quantization of test inputs saves valuable classification time with an negligible overhead during training. Classification accuracy is unaffected for already 100 bins per dimension. Besides, the adaptive quantization is applicable to non-normalized features without significant extra costs.

Hyperparameter Optimization with GP/HIK (Section 7.2.3)

Verifying the Bound on the Negative Marginal Log-Likelihood: Our bound is a close approximation to the exact negative log-likelihood and has similar minima.

Different Generalized HIK for Multi-class Classification: Our optimization technique allows for tuning of hyperparameters if required by given data.

Different Generalized HIK for Binary Classification: Our introduced approximation of the negative log-likelihood is applicable to scenarios with tens of thousands of training examples.

Early Stopping of the Linear Solver When Hyperparameters Are Optimized: Early stopping is also applicable with activated hyperparameter optimization and saves valuable computation time during training.

Feature Relevance Determination: The probabilistic formulation of GP models is beneficial for feature relevance determination. However, an extension to thousands of hyperparameters for handling high-dimensional features is still an open issue.

Uncertainty Prediction with GP/HIK (Section 7.2.4)

Computation Time Analysis for Variance Approximations: Our introduced variance approximations come with different trade-offs regarding accuracy, computation time, and memory and can be chosen according to the current application constraints.

Tightness of Variance Approximations: FAPU leads to close approximations and should be used when approximation errors are undesirable. If only the relative order of variance estimates is required, CAPU is a reasonable choice.

Semantic Segmentation with GP/HIK (Section 7.2.5)

We can apply GP models to semantic segmentation tasks without the necessity of sparse approximation techniques. Furthermore, replacing classifiers in existing frameworks for semantic segmentation with GP models can pay off and result in potential performance gains.

C.2 Active Learning: Summary of Findings

Active Learning in Binary Classification Scenarios (Section 7.3.1)

Visual Inspection in a 2D Problem: We can intuitively observe advantages and limitations of different active learning criteria by visually inspecting them in a controlled 2D setup.

Quantitative Comparisons on ImageNet: We found that samples minimizing the estimated empirical risk or with large predictive variances are not feasible for active learning from real-world data. Remaining techniques lead to comparable results.

Quantitative Comparisons on Caltech-256: We can verify previous findings for active learning in binary classification experiments. An exception is $\mathcal{Q}_{\text{GP-ERM}}$ which leads to improved results. Best classification performance is again obtained using criteria based on estimated model output changes.

Qualitative Analysis of Selected Examples: Examples selected by $\mathcal{Q}_{\text{GP-}\Delta f\text{-approx}}$ are intuitively useful upon visual inspection. Queries are selected to improve the representation of positive data and to improve discrimination abilities with visually similar negative data.

EMOC with Perfect Label Estimates: EMOC can benefit from improved label estimates.

Importance of Density-based Re-weighting: The incorporation of data density into active sample selection is beneficial.

Active Learning in Multi-class Classification Scenarios (Section 7.3.2)

Recognition of Handwritten Digits on USPS: Our EMOC criterion is well suited for multi-class active learning and discovery of simple handwritten digits even without modelling rejections of unnameable instances.

Face Identification: Modelling unnameable instances is not required for person discovery in face identification scenarios. However, it substantially helps to query informative examples important for accurate person identification.

Active Discovery with Object Proposals: Modelling categorical and non-categorical rejections is crucial for real-world active learning tasks similar to biodiversity applications.

Improvement versus Selection Time: We conclude that EMOC requires an acceptable computational overhead given the superior classification and discovery results.

Qualitative Analysis: We argue that active learning techniques should mainly be compared quantitatively.

Detailed Analysis for EMOC (1): Loss Functions: A soft $\mathcal{L}_{| \cdot |}$ -loss for multi-class scenarios is preferable when evaluating EMOC.

Detailed Analysis for EMOC (2): Probability Estimates: Monte-carlo estimates for class probabilities are well suited to evaluate EMOC.

C.3 Mid-level Features: Summary of Findings

Part Annotation Transfer for Fine-grained Recognition (Section 7.4.1)

Localization Error Analysis (1) – Comparison with DPMs: Our part annotation transfer has a better part localization accuracy than the DPM baseline while being simple and easy to implement.

Fine-grained Recognition with Transferred Part Annotations: A simple part transfer and standard BoW features are comparable or even superior to sophisticated feature learning techniques of the same era. Furthermore, deep architectures again significantly boost performance. Hence, replacing BoW features with CNN representations is likely to further improve classification accuracy.

Localization Error Analysis (2) – Noisy Annotations: Accurate part annotations in training images are beneficial for fine-grained recognition.

Mid-level Features for Fine-grained Recognition (Section 7.4.2)

Bird Identification Using Mid-level Features: Unsupervised discovery of mid-level features is possible but is computationally demanding. Our supervised technique leads to classification results comparable to techniques for learning of mid-level features. However, it does not obtain state-of-the-art results on its own.

Analysis of Individual Steps: Our choices for each step fit well together. Thereby, no human part annotation nor a heuristic for part selection is required.

Activation Analysis of Learned Filters: Our discovered patch detectors are most strongly activated on intuitively useful object parts and generalize well from training to test images.

Exemplar-specific Representations via Local Learning (Section 7.4.3)

Matching Accuracy for Local Learning: The combination of texture, shape, and color is relevant to compute useful neighborhoods for local learning. However, the matching would benefit from further improvements presented in the image retrieval community.

Fine-grained Recognition with Local Learning: Exemplar-specific mid-level features can lead to improved discrimination abilities. However, the success of local learning heavily depends of the matching accuracy.

Dimensionality of Learned Representations: Exemplar-specific representations are clearly more compact than global representations.

Activation Analysis for Varying Neighborhood Sizes: Exemplar-specific patch detectors often correspond to category-specific object parts. The optimal number of neighbors depends on viewpoint, pose, and background.

Combining Mid-level Features and Semantic Parts (Section 7.4.4)

Fusing mid-level features and semantic parts adds valuable information and is able to further improve classification accuracy in fine-grained recognition scenarios.

Bibliography

- Ablavsky, V. and Sclaroff, S. (2011). Learning parameterized histogram kernels on the simplex manifold for image and action classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1473–1480. (cited on pages 98, 244, 345, 345, 245, and 246)
- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Susstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(11):2274–2282. (cited on page 123)
- Agarwal, S. and Roth, D. (2002). Learning a sparse representation for object detection. In *European Conference on Computer Vision (ECCV)*, pages 113–127. (cited on page 189)
- Angiulli, F. (2005). Fast condensed nearest neighbor rule. In *International Conference on Machine Learning (ICML)*, pages 25–32. (cited on page 32)
- Apple computer inc. (1977). *Simplicity is the ultimate sophistication: Introducing Apple II, the personal computer*. (cited on page 210)
- Arandjelovic, R. and Zisserman, A. (2012). Three things everyone should know to improve object retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2911–2918. (cited on page 205)
- Arandjelovic, R. and Zisserman, A. (2013). All about vlad. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1578–1585. (cited on page 77)
- Association of European Records and Rarities Committees (2003). Taxonomic recommendations. (cited on page 191)
- Atrey, P. K., Hossain, M. A., El Saddik, A., and Kankanhalli, M. S. (2010). Multimodal fusion for multimedia analysis: a survey. *Multimedia systems*, 16(6):345–379. (cited on page 307)

BIBLIOGRAPHY

- Azizpour, H. and Laptev, I. (2012). Object detection using strongly-supervised deformable part models. In *European Conference on Computer Vision (ECCV)*, pages 836–849. Springer. (cited on page 209)
- Babenko, A. and Lempitsky, V. (2015). Aggregating local deep features for image retrieval. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1269–1277. (cited on page 318)
- Babenko, A., Slesarev, A., Chigorin, A., and Lempitsky, V. (2014). Neural codes for image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 584–599. (cited on pages 81 and 205)
- Bache, K. and Lichman, M. (2013). UCI machine learning repository. (cited on pages 216, 270, 272, and 279)
- Bai, Z. and Golub, G. (1997). Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices. *Annals of Numerical Mathematics*, 4:29–38. (cited on pages 111, 333, and 389)
- Baram, Y., El-Yaniv, R., and Luz, K. (2004). Online choice of active learning algorithms. *Journal of Machine Learning Research (JMLR)*, 5:255–291. (cited on pages 153, 156, and 164)
- Barla, A., Franceschi, E., Odone, F., and Verri, A. (2002). Image kernels. In *Pattern Recognition with Support Vector Machines*, pages 83–96. Springer. (cited on pages 61 and 97)
- Bartlett, P. and Shawe-Taylor, J. (1999). Generalization performance of support vector machines and other pattern classifiers. In *Advances in Kernel Methods - Support Vector Learning*, pages 43–54. MIT Press, Cambridge, MA. (cited on pages 40 and 41)
- Bartlett, P. L. and Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *Journal of Machine Learning Research (JMLR)*, 9:1823–1840. (cited on page 87)
- Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417. Springer. (cited on pages 74 and 201)
- Beckstein, C., Böcker, S., Bogdan, M., H. Bruelheide, H. M. B., Denzler, J., Dittrich, P., Grosse, I., Hinneburg, A., König-Ries, B., Löffler, F., Marz, M., Müller-Hannemann, M., Winter, M., and Zimmermann, W. (2014). Explorative analysis of heterogeneous, unstructured, and uncertain data: A computer science perspective on

- biodiversity research. In *International Conference on Data Management Technologies and Applications (DATA)*, pages 251–257. (cited on page 315)
- Bendale, A. and Boulton, T. (2015a). Towards open set deep networks. *arXiv preprint arXiv:1511.06233*. (cited on page 87)
- Bendale, A. and Boulton, T. (2015b). Towards open world recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 87)
- Bendavid, S., Cesabianchi, N., Haussler, D., and Long, P. (1995). Characterizations of learnability for classes of $\{0, \dots, n\}$ -valued functions. *Journal of Computer and System Sciences*, 50(1):74 – 86. (cited on page 21)
- Berg, T. and Belhumeur, P. N. (2013). Poof: Part-based one-vs-one features for fine-grained categorization, face verification, and attribute estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 955–962. (cited on pages 189, 291, 292, 296, 297, 303, and 309)
- Bernstein, D. S. (2009). *Matrix Mathematics*. Princeton University Press, 2nd edition. (cited on pages 131 and 336)
- Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115. (cited on page 13)
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer. (cited on pages 27, 30, 33, 34, 36, 39, 45, 60, 68, 73, 77, 184, and 185)
- Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1987). Occam’s razor. *Information processing letters*, 24(6):377–380. (cited on page 22)
- Bo, L. and Sminchisescu, C. (2008). Greedy block coordinate descent for large scale gaussian process regression. In *Uncertainty in Artificial Intelligence (UAI)*. (cited on pages 95, 106, 345, 232, and 233)
- Bochner, S. (1955). *Harmonic Analysis and the Theory of Probability*. University of California Press, Berkeley and Los Angeles, CA, USA. (cited on page 60)
- Bodesheim, P. (2016). *Discovering unknown visual objects with novelty detection techniques*. PhD thesis, FSU Jena. preprint January 2016. (cited on pages 88, 95, 153, and 323)
- Bodesheim, P., Freytag, A., Rodner, E., and Denzler, J. (2013a). Approximations of gaussian process uncertainties for visual recognition problems. In *Scandinavian Conference on Image Analysis (SCIA)*, pages 182–194. (cited on pages 71, 95, 153, and 324)

BIBLIOGRAPHY

- Bodesheim, P., Freytag, A., Rodner, E., and Denzler, J. (2013b). An efficient approximation for gaussian process regression. Technical report, Computer Vision Group, Friedrich Schiller University Jena, Germany. Technical Report TR-FSU-INF-CV-2013-01. (cited on pages 95, 323, and 324)
- Bodesheim, P., Freytag, A., Rodner, E., and Denzler, J. (2015). Local novelty detection in multi-class recognition problems. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 813–820. (cited on pages 205, 322, and 323)
- Bodesheim, P., Freytag, A., Rodner, E., Kemmler, M., and Denzler, J. (2013c). Kernel null space methods for novelty detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3374–3381. (cited on pages 321 and 322)
- Bodesheim, P., Rodner, E., Freytag, A., and Denzler, J. (2012). Divergence-based one-class classification using gaussian processes. In *British Machine Vision Conference (BMVC)*, pages 50.1–50.11. (cited on pages 139, 171, 172, 321, and 322)
- Bonwell, C. C. and Eison, J. A. (1991). *Active learning: Creating excitement in the classroom*. School of Education and Human Development, George Washington University Washington, DC. (cited on page 152)
- Bottou, L. and Vapnik, V. (1992). Local learning algorithms. *Neural computation*, 4(6):888–900. (cited on page 204)
- Boughorbel, S., Tarel, J.-P., and Boujemaa, N. (2005). Generalized histogram intersection kernel for image recognition. In *IEEE International Conference on Image Processing (ICIP)*, pages III – 161–4. (cited on page 97)
- Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1365–1372. (cited on page 189)
- Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. (cited on pages 48 and 106)
- Branson, S., Horn, G. V., Wah, C., Perona, P., and Belongie, S. (2014a). The ignorant led by the blind: A hybrid human–machine vision system for fine-grained categorization. *International Journal of Computer Vision (IJCV)*, 108(1-2):3–29. (cited on page 317)
- Branson, S., Van Horn, G., Belongie, S., and Perona, P. (2014b). Improved bird species categorization using pose normalized deep convolutional nets. In *British Machine Vision Conference (BMVC)*. (cited on pages 81, 190, and 191)
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. (cited on page 129)

- Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. In *International Conference on Machine Learning (ICML)*, volume 3, pages 59–66. (cited on page 150)
- Bronstein, I. N. and Semendjajew, K. A. (1979). *Taschenbuch der Mathematik*. BSB B.G. Teubner Verlagsgesellschaft, Leipzig and Verlag Nauka, Moskau, 19 edition. (cited on page 222)
- Bruce, V., Green, P. R., and Georgeson, M. A. (2003). *Visual perception: Physiology, psychology, & ecology*. Psychology Press, 4 edition. (cited on page 1)
- Brust, C.-A., Sickert, S., Simon, M., Rodner, E., and Denzler, J. (2015). Convolutional patch networks with spatial prior for road detection and urban scene understanding. In *International Conference on Computer Vision Theory and Applications (VISAPP)*. (cited on pages 93 and 124)
- Budai, A., Bock, R., Maier, A., Hornegger, J., and Michelson, G. (2013). Robust vessel segmentation in fundus images. *International journal of biomedical imaging*, 2013. (cited on page 123)
- Cai, H., Yan, F., and Mikolajczyk, K. (2010). Learning weights for codebook in image classification and retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2320–2327. (cited on page 244)
- Cai, W., Zhang, Y., and Zhou, J. (2013). Maximizing expected model change for active learning in regression. In *International Conference on Data Mining (ICDM)*. (cited on page 155)
- Cai, W., Zhang, Y., Zhou, S., Wang, W., Ding, C. H. Q., and Gu, X. (2014). Active learning for support vector machines with maximum model change. In *European Conference Machine Learning and Knowledge Discovery in Databases (ECML, PKDD)*, pages 211–226. (cited on page 155)
- Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *European Conference on Computer Vision (ECCV)*, pages 778–792. Springer. (cited on pages 74 and 201)
- Campbell, C., Cristianini, N., and Smola, A. (2000). Query learning with large margin classifiers. In *International Conference on Machine Learning (ICML)*, pages 111–118. (cited on page 154)
- Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. R. H., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Conference on Artificial Intelligence (AAAI)*, volume 5, page 3. (cited on page 89)

BIBLIOGRAPHY

- Cauwenberghs, G. and Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 409–415. (cited on page 129)
- Cebron, N. and Berthold, M. (2009). Active learning for object classification: from exploration to exploitation. *Data Mining and Knowledge Discovery*, 18:283–299. (cited on pages 157 and 164)
- Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27. (cited on pages 50, 345, and 228)
- Chang, F., Guo, C.-Y., Lin, X.-R., and Lu, C.-J. (2010). Tree decomposition for large-scale svm problems. *Journal of Machine Learning Research (JMLR)*, 11:2935–2972. (cited on pages 93 and 95)
- Chao, Y.-W., Wang, Z., He, Y., Wang, J., and Deng, J. (2015). Hico: A benchmark for recognizing human-object interactions in images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1017–1025. (cited on page 2)
- Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014a). Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)*. (cited on page 81)
- Chatfield, K., Simonyan, K., and Zisserman, A. (2014b). Efficient on-the-fly category retrieval using convnets and gpus. In *Asian Conference on Computer Vision (ACCV)*, pages 129–145. (cited on page 81)
- Chen, X., Shrivastava, A., and Gupta, A. (2013). Neil: Extracting visual knowledge from web data. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1409–1416. (cited on page 89)
- Choi, J., Rastegari, M., Farhadi, A., and Davis, L. (2013). Adding unlabeled samples to categories by learned attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 875–882. (cited on page 188)
- Christakos, G. and Papanicolaou, V. (2000). Norm-dependent covariance permissibility of weakly homogeneous spatial random fields and its consequences in spatial statistics. *Stochastic Environmental Research and Risk Assessment*, 14(6):471–478. (cited on pages 61 and 73)
- Chum, O., Philbin, J., Sivic, J., Isard, M., and Zisserman, A. (2007). Total recall: Automatic query expansion with a generative feature model for object retrieval. In *IEEE International Conference on Computer Vision (ICCV)*. (cited on page 205)

- Coates, A., Lee, H., and Ng, A. Y. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 215–223. (cited on pages 76 and 185)
- Coates, A. and Ng, A. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning (ICML)*, pages 921–928. (cited on page 185)
- Cohn, D. A. (1996). Neural network exploration using optimal experiment design. *Neural networks*, 9(6):1071–1083. (cited on page 156)
- Comaniciu, D. and Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):603–619. (cited on page 123)
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*, volume 3. MIT press Cambridge. (cited on page 204)
- Cornell Lab of Ornithology (2015). All about birds. (cited on page 192)
- Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27. (cited on page 32)
- Cressie, N. (1990). The origins of kriging. *Mathematical geology*. (cited on page 73)
- Csató, L. and Oppor, M. (2001). Sparse representation for gaussian process models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 444–450. (cited on pages 94 and 129)
- Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision (ECCV-WS)*. (cited on pages 75 and 184)
- Csurka, G. and Perronnin, F. (2010). An efficient approach to semantic segmentation. *International Journal of Computer Vision (IJCV)*, pages 1–15. (cited on pages 123, 124, 185, 250, 251, and 252)
- Dalal, N. and Triggs, B. (2005). Histogram of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. (cited on pages 74, 75, and 201)
- Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *International Conference on Machine Learning (ICML)*, pages 233–240, New York, NY, USA. (cited on page 221)

BIBLIOGRAPHY

- de Freitas, N., Wang, Y., Mahdaviani, M., and Lang, D. (2006). Fast krylov methods for n-body learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 251–258. (cited on page 95)
- Dean, T., Ruzon, M. A., Segal, M., Shlens, J., Vijayanarasimhan, S., and Yagnik, J. (2013). Fast, accurate detection of 100,000 object classes on a single machine. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1814–1821. (cited on pages 202 and 318)
- Demir, B. and Bruzzone, L. (2014). A novel active learning method in relevance feedback for content-based remote sensing image retrieval. *IEEE Transactions on Geoscience and Remote Sensing (TGRS)*. (cited on page 154)
- Deng, J., Berg, A., Li, K., and Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision (ECCV)*, pages 71–84. (cited on pages 93 and 228)
- Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., and Adam, H. (2014). Large-scale object classification using label relation graphs. In *European Conference on Computer Vision (ECCV)*, pages 48–64. (cited on page 316)
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. (cited on pages 80, 92, 151, 204, 213, and 214)
- Deng, J., Krause, J., Berg, A., and Fei-Fei, L. (2012). Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3450–3457. (cited on pages 204 and 316)
- Deng, J., Krause, J., and Fei-Fei, L. (2013). Fine-grained crowdsourcing for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. (cited on page 184)
- Deng, L. and Yu, D. (2013). Deep learning for signal and information processing. *Microsoft Research Monograph*. (cited on page 80)
- Denzler, J., Rodner, E., Bodesheim, P., and Freytag, A. (2013). Beyond the closed-world assumption: The importance of novelty detection and open set recognition. In *GCPR Workshop on Unsolved Problems in Pattern Recognition (GCPR-WS)*. (cited on page 323)

- Doersch, C., Gupta, A., and Efros, A. A. (2013). Mid-level visual element discovery by discriminative mean-shift. In *Advances in Neural Information Processing Systems (NIPS)*, pages 494–502. (cited on pages 190 and 199)
- Domingos, P. (1999). The role of occam’s razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425. (cited on page 22)
- Donahue, J. and Grauman, K. (2011). Annotator rationales for visual recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1402. (cited on pages 188 and 317)
- Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning (ICML)*. (cited on pages 80, 81, and 230)
- Duan, K., Keerthi, S. S., Chu, W., Shevade, S. K., and Poo, A. N. (2003). Multi-category classification by soft-max combination of binary classifiers. In *Multiple Classifier Systems (MCS)*, pages 125–134. (cited on pages 178, 287, and 307)
- Duan, K., Parikh, D., Crandall, D., and Grauman, K. (2012). Discovering localized attributes for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3474–3481. (cited on pages 82, 151, 184, and 188)
- Dubout, C. and Fleuret, F. (2012). Exact acceleration of linear object detectors. In *European Conference on Computer Vision (ECCV)*, pages 301–311. (cited on pages 198 and 202)
- Duncan, W. J. (1944). Some devices for the solution of large sets of simultaneous linear equations: With an appendix on the reciprocation of partitioned matrices. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science : Series 7*, 35(249):660–670. (cited on page 134)
- Eaton, E. and Ruvolo, P. L. (2013). Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning (ICML)*, pages 507–515. (cited on page 85)
- Ebert, S., Fritz, M., and Schiele, B. (2012). Ralf: A reinforced active learning formulation for object class recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3626–3633. (cited on pages 154, 157, 164, and 268)
- Enzweiler, M. (2011). *Compound Models for Vision-Based Pedestrian Recognition*. PhD thesis, Ruprecht–Karls–University Heidelberg. (cited on page 2)

BIBLIOGRAPHY

- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338. (cited on pages 92 and 190)
- Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*, 9:1871–1874. (cited on pages 50, 93, 345, 228, 231, 291, and 296)
- Fang, M. and Zhu, X. (2012). I don't know the label: Active learning with blind knowledge. In *International Conference on Pattern Recognition (ICPR)*, pages 2238–2241. (cited on pages 151, 180, and 182)
- Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. (2009). Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1778–1785. (cited on page 188)
- Farquhar, J., Szedmak, S., Meng, H., and Shawe-Taylor, J. (2005). Improving "bag-of-keypoints" image categorisation: Generative models and pdf-kernels. Technical report, University of Southampton. (cited on page 185)
- Farrell, R., Oza, O., Zhang, N., Morariu, V. I., Darrell, T., and Davis, L. S. (2011). Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *IEEE International Conference on Computer Vision (ICCV)*, pages 161–168. (cited on pages 184, 191, and 217)
- Fawcett, T. (2004). Roc graphs: Notes and practical considerations for researchers. *Machine Learning*, 31:1–38. (cited on page 221)
- Fei-Fei, L. (2006). Knowledge transfer in learning to recognize visual objects classes. In *International Conference on Development and Learning*. (cited on page 85)
- Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision (CVPR-WS)*. (cited on page 92)
- Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 524 – 531 vol. 2. (cited on page 92)
- Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Blackwell Publishing Ltd. (cited on pages 204, 213, and 316)
- Felzenszwalb, P. and Huttenlocher, D. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59:167–181. (cited on pages 123, 193, 201, and 250)

- Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1627–1645. (cited on pages 74, 75, 81, 82, 83, 192, 195, 197, 208, 209, and 289)
- Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision (IJCV)*, 61(1):55–79. (cited on page 189)
- Fischler, M. A. and Elschlager, R. A. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers (TC)*, 22(1):67–92. (cited on page 189)
- Fong, D. C.-L. and Saunders, M. A. (2012). Cg versus minres: An empirical comparison. *SQU Journal for Science*, 17(1):44–62. (cited on page 232)
- Fowlkes, C., Belongie, S., Chung, F., and Malik, J. (2004). Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(2):214–225. (cited on page 61)
- Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139. (cited on page 188)
- Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168. (cited on pages 154 and 156)
- Freytag, A., Fröhlich, B., Rodner, E., and Denzler, J. (2012a). Efficient semantic segmentation with gaussian processes and histogram intersection kernels. In *International Conference on Pattern Recognition (ICPR)*, pages 3313–3316. (cited on pages 10, 92, and 122)
- Freytag, A., Rodner, E., Bodesheim, P., and Denzler, J. (2012b). Rapid uncertainty computation with gaussian processes and histogram intersection kernels. In *Asian Conference on Computer Vision (ACCV)*, pages 511–524. (cited on pages 9, 10, 11, 92, 116, 117, 128, 144, 146, 148, 153, and 157)
- Freytag, A., Rodner, E., Bodesheim, P., and Denzler, J. (2013). Labeling examples that matter: Relevance-based active learning with gaussian processes. In *German Conference on Pattern Recognition (GCPR)*, pages 282–291. (cited on pages 10, 11, 128, 140, 146, 148, 161, 164, and 166)

BIBLIOGRAPHY

- Freytag, A., Rodner, E., Darrell, T., and Denzler, J. (2014a). Exemplar-specific patch features for fine-grained recognition. In *German Conference on Pattern Recognition (GCPR)*, pages 144–156. (cited on pages 11, 191, and 204)
- Freytag, A., Rodner, E., and Denzler, J. (2014b). Birds of a feather flock together - local learning of mid-level representations for fine-grained recognition. In *ECCV Workshop on Parts and Attributes (ECCV-WS)*. (cited on pages 11 and 204)
- Freytag, A., Rodner, E., and Denzler, J. (2014c). Selecting influential examples: Active learning with expected model output changes. In *European Conference on Computer Vision (ECCV)*, pages 562–577. (cited on pages 10, 11, 140, 146, 148, 166, 171, 172, and 177)
- Freytag, A., Rühle, J., Bodesheim, P., Rodner, E., and Denzler, J. (2014d). Seeing through bag-of-visual-word glasses: towards understanding quantization effects in feature extraction methods. In *International Conference on Pattern Recognition (ICPR) - FEAST workshop*. (cited on pages 231 and 326)
- Freytag, A., Schadt, A., and Denzler, J. (2015). Interactive image retrieval for biodiversity research. In *German Conference on Pattern Recognition (GCPR)*, pages 129–141. (cited on pages 81, 301, 315, and 325)
- Fröhlich, B. (2013). *Semantic Segmentation with Efficient Tree-Based Methods*. PhD thesis, FSU Jena. (cited on pages 124 and 125)
- Fröhlich, B., Rodner, E., and Denzler, J. (2010). A fast approach for pixelwise labeling of facade images. In *International Conference on Pattern Recognition (ICPR)*, pages 3029–3032. (cited on pages 123, 124, 218, 219, 250, 251, 252, and 254)
- Fröhlich, B., Rodner, E., and Denzler, J. (2012). As time goes by — anytime semantic segmentation with iterative context forests. In *Symposium of the German Association for Pattern Recognition (DAGM)*, pages 1–10. (cited on page 124)
- Fröhlich, B., Rodner, E., and Denzler, J. (2012a). Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach. In *Asian Conference on Computer Vision (ACCV)*, pages 218–231. (cited on pages 124, 252, and 254)
- Fröhlich, B., Rodner, E., Kemmler, M., and Denzler, J. (2012b). Large-scale gaussian process classification using random decision forests. *Pattern Recognition and Image Analysis*, 22(1):113–120. (cited on page 95)
- Fröhlich, B., Rodner, E., Kemmler, M., and Denzler, J. (2013). Large-scale gaussian process multi-class classification for semantic segmentation and facade recogni-

- tion. *Machine Vision and Applications (MVA)*, 24(5):1043–1053. (cited on pages 179, 286, and 287)
- Frome, A., Singer, Y., and Malik, J. (2006). Image retrieval and classification using local distance functions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 417–424. (cited on page 204)
- Frome, A., Singer, Y., Sha, F., and Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8. (cited on page 205)
- Gauss, C. F. (1821). Theory of the combination of observations least subject to errors, part one, part two, supplement. In *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM). G. W. Stewart translated the original manuscript “*Theoria combinationis observationum erroribus minimis obnoxiae*” in 1995. (cited on pages 130 and 140)
- Gavves, E., Fernando, B., Snoek, C., Smeulders, A., and Tuytelaars, T. (2013). Fine-grained categorization by alignments. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1713–1720. (cited on page 212)
- Genton, M. (2002). Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research (JMLR)*, 2:299–312. (cited on pages 60 and 61)
- Girshick, R. (2015). Fast r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*. (cited on page 2)
- Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014a). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. (cited on page 93)
- Girshick, R. B., Felzenszwalb, P. F., and McAllester, D. (2014b). Discriminatively trained deformable part models, release 5. <http://people.cs.uchicago.edu/~rbg/latent-release5/>. (cited on pages 83 and 289)
- Golub, G. H. and Loan, C. F. V. (1996). *Matrix Computations*. Johns Hopkins University Press, 3 edition. (cited on pages 95, 105, 106, 107, 112, 317, and 334)
- Gong, Y., Wang, L., Guo, R., and Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision (ECCV)*, pages 392–407. (cited on page 81)
- Görling, C., Freytag, A., Rodner, E., and Denzler, J. (2013). Fine-grained categorization - short summary of our entry for the imagenet challenge 2012. *arXiv preprint arXiv:1310.4759*. (cited on page 324)

BIBLIOGRAPHY

- Görling, C., Rodner, E., Freytag, A., and Denzler, J. (2014). Nonparametric part transfer for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2489–2496. (cited on pages 12, 207, and 292)
- Grauman, K. and Darrell, T. (2007). The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research (JMLR)*, 8:725–760. (cited on page 98)
- Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology. (cited on pages 92, 214, and 215)
- Guillaumin, M., Verbeek, J., and Schmid, C. (2009). Is that you? metric learning approaches for face identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 498–505. (cited on page 273)
- Hager, W. W. (1989). Updating the inverse of a matrix. *Society for Industrial and Applied Mathematics (SIAM) Review*, 31(2):221–239. (cited on pages 131 and 389)
- Haines, T. and Xiang, T. (2011). Active learning using dirichlet processes for rare class discovery and classification. In *British Machine Vision Conference (BMVC)*. (cited on page 157)
- Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2014). Simultaneous detection and segmentation. In *European Conference on Computer Vision (ECCV)*, page Simultaneous Detection and Segmentation. (cited on page 93)
- Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 81)
- Hariharan, B., Malik, J., and Ramanan, D. (2012). Discriminative decorrelation for clustering and classification. In *European Conference on Computer Vision (ECCV)*, pages 459–472. (cited on pages 36, 190, 195, 196, 202, 289, 295, and 300)
- Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516. (cited on page 32)
- He, J. and Carbonell, J. G. (2007). Nearest-neighbor-based active learning for rare category detection. In *Advances in Neural Information Processing Systems (NIPS)*, pages 633–640. (cited on page 153)
- He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, pages 346–361. (cited on page 81)

- He, X., Zemel, R. S., and Carreira-Perpinan, M. A. (2004). Multiscale conditional random fields for image labeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II-695-II-702 Vol.2. (cited on page 124)
- Hermann, M. (2006). *Numerische Mathematik*. Oldenburg Verlag München Wien. (cited on page 334)
- Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436. (cited on pages 107, 232, and 233)
- Hoffman, J., Pathak, D., Darrell, T., and Saenko, K. (2015). Detector discovery in the wild: Joint multiple instance and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 80)
- Hoffman, J., Rodner, E., Donahue, J., Kulis, B., and Saenko, K. (2014). Asymmetric and category invariant feature transformations for domain adaptation. *International Journal of Computer Vision (IJCV)*, 109(1-2):28–41. (cited on page 318)
- Hospedales, T., Gong, S., and Xiang, T. (2012). A unifying theory of active discovery and learning. In *European Conference on Computer Vision (ECCV)*, pages 453–466. (cited on pages 88, 181, 271, and 279)
- Hospedales, T. M., Gong, S., and Xiang, T. (2013). Finding rare classes: Active learning with generative and discriminative models. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(2):374–386. (cited on pages 129, 157, and 164)
- Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear svm. In *International Conference on Machine Learning (ICML)*, pages 408–415. (cited on page 94)
- Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst. (cited on pages 216, 274, 275, and 279)
- Huang, R. and Riloff, E. (2012). Bootstrapped training of event extraction classifiers. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 286–295. (cited on pages 197 and 199)
- Ionescu, R., Popescu, M., and Grozea, C. (2013). Local learning to improve bag of visual words model for facial expression recognition. In *International Conference on Machine Learning - Workshop on Representation Learning (ICML-WS)*. (cited on page 204)

BIBLIOGRAPHY

- Jaeger, J., Simon, M., Denzler, J., Wolff, V., Fricke-Neuderth, K., and Kruschel, C. (2015). Croatian fish dataset: Fine-grained classification of fish species in their natural habitat. In *BMVC Workshop on Machine Vision of Animals and their Behaviour (BMVC-WS)*. (cited on page 315)
- Jain, L., Scheirer, W., and Boulton, T. (2014). Multi-class open set recognition using probability of inclusion. In *European Conference on Computer Vision (ECCV)*, pages 393–409. (cited on page 88)
- Jain, P. and Kapoor, A. (2009). Active learning for large multi-class problems. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 762–769. (cited on pages 154, 158, 271, and 279)
- Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3304–3311. (cited on page 77)
- Jia, Y. and Darrell, T. (2011). Heavy-tailed distances for gradient based image descriptors. In *Advances in Neural Information Processing Systems (NIPS)*, pages 397–405. (cited on page 206)
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, pages 675–678. (cited on pages 230, 240, and 276)
- Jia, Y., Vinyals, O., and Darrell, T. (2013). Pooling-invariant image feature learning. In *Arxiv*, volume abs/1302.5056. (cited on pages 78, 185, 186, 271, 291, 292, 296, 303, and 309)
- Joachims, T. (1998). *Text categorization with support vector machines: Learning with many relevant features*. Springer. (cited on page 75)
- John Winn, Antonio Criminisi, T. M. (2004). Microsoft research cambridge object recognition image database (msrc). <http://research.microsoft.com/vision/cambridge/recognition/>. (cited on page 92)
- Joshi, A., Porikli, F., and Papanikolopoulos, N. (2009). Multi-class active learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2372–2379. (cited on pages 154, 270, and 279)
- Juneja, M., Vedaldi, A., Jawahar, C., and Zisserman, A. (2013). Blocks that shout: Distinctive parts for scene classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 923–930. (cited on pages 190, 193, 195, 198, 199, 297, and 298)

- Jurie, F. and Triggs, B. (2005). Creating efficient codebooks for visual recognition. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 604–610. (cited on pages 75, 184, and 192)
- Käding, C., Freytag, A., Rodner, E., Bodesheim, P., and Denzler, J. (2015). Active learning and discovery of object categories in the presence of unnameable instances. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4343–4352. (cited on pages 11, 140, 146, 148, 171, 172, and 176)
- Käding, C., Freytag, A., Rodner, E., and Denzler, J. (2016). Watch, Ask, Learn, and Improve: A lifelong learning cycle for visual recognition. In *European Symposium on Artificial Neural Networks (ESANN)*. (submitted). (cited on page 316)
- Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. (2010). Gaussian processes for object categorization. *International Journal of Computer Vision (IJCV)*, 88:169–188. (cited on pages 69, 71, 153, 154, 157, 158, 159, 160, 164, 165, 176, 177, 258, 261, 263, 270, 279, 281, and 382)
- Karacali, B. and Krim, H. (2003). Fast minimization of structural risk by nearest neighbor rule. *IEEE Transactions on Neural Networks*, 14(1):127–137. (cited on page 32)
- Kemmler, M. (2013). *Machine Learning Methods for the Automatic Identification of Microorganisms from Raman Spectroscopic Data*. PhD thesis, FSU Jena. (cited on pages 30, 34, 73, and 96)
- Kemmler, M., Rodner, E., and Denzler, J. (2010). One-class classification with gaussian processes. In *Asian Conference on Computer Vision (ACCV)*, pages 489–500. (cited on page 141)
- King, R., Rowland, J., Oliver, S., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L., Sparkes, A., Whelan, K., and Clare, A. (2009). The automation of science. *Science*, 324(5923):85–89. (cited on page 150)
- Korc, F. and Förstner, W. (2009). etrimis image database for interpreting images of man-made scenes. Technical Report TRIGG-P-2009-01, Dept. of Photogrammetry, University of Bonn. (cited on pages 218, 219, and 254)
- Kovashka, A., Parikh, D., and Grauman, K. (2015). Whittlesearch: Interactive image search with relative attribute feedback. *International Journal of Computer Vision (IJCV)*, pages 1–26. (cited on page 317)
- Kovashka, A., Vijayanarasimhan, S., and Grauman, K. (2011). Actively selecting annotations among objects and attributes. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1403–1410. (cited on page 156)

BIBLIOGRAPHY

- Krähenbühl, P. and Koltun, V. (2014). Geodesic object proposals. In *European Conference on Computer Vision (ECCV)*, pages 725–739. (cited on pages 276, 283, 284, and 285)
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. (cited on pages 79, 80, 93, 103, 206, and 276)
- Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I. C., and Soares, J. V. (2012a). Leafsnap: A computer vision system for automatic plant species identification. In *European Conference on Computer Vision (ECCV)*, pages 502–516. (cited on page 191)
- Kumar, S., Mohri, M., and Talwalkar, A. (2012b). Sampling methods for the nyström method. *Journal of Machine Learning Research (JMLR)*, 13(1):981–1006. (cited on page 78)
- Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2009a). Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(12):2129–2142. (cited on page 202)
- Lampert, C. H., Nickisch, H., and Harmeling, S. (2009b). Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 951–958. (cited on pages 82 and 188)
- Lang, K. and Baum, E. (1992). Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, page 335–340. (cited on page 150)
- Larlus, D. and Jurie, F. (2009). Latent mixture vocabularies for object categorization and segmentation. *Image and Vision Computing*, 27:523–534. (cited on page 185)
- Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems (NIPS)*, pages 609–616. (cited on page 94)
- Lawrence, N. D., Platt, J. C., and Jordan, M. I. (2005). Extensions of the informative vector machine. In J. Winkler, N. D. L. and Niranjan, M., editors, *Deterministic and Statistical Methods in Machine Learning*, pages 56–87. Springer-Verlag, Berlin. (cited on page 71)

- Lazebnik, S. and Raginsky, M. (2009). Supervised learning of quantizer codebooks by information loss minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(7):1294–1309. (cited on page 186)
- Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178. (cited on pages 77, 92, 200, 215, and 291)
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. (cited on page 187)
- Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2006). Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 801–808. (cited on page 185)
- Lee, S., Zhang, H., and Crandall, D. J. (2015). Predicting geo-informative attributes in large-scale image collections using convolutional neural networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 550–557. (cited on page 2)
- Lee, Y. J., Efros, A. A., and Hebert, M. (2013). Style-aware mid-level representation for discovering visual connections in space and time. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1857–1864. (cited on pages 190, 291, 292, 296, 303, and 309)
- Léon Bottou, Olivier Chapelle, D. D. J. W., editor (2007). *Large-scale kernel machines*. MIT Press, Cambridge, Massachusetts, London, England. (cited on pages 46, 48, and 109)
- Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Conference on Research and Development in Information Retrieval*, pages 3–12. Springer-Verlag New York, Inc. (cited on page 154)
- Li, X. and Guo, Y. (2014). Multi-level adaptive active learning for scene classification. In *European Conference on Computer Vision (ECCV)*, pages 234–249. (cited on page 156)
- Li, Y., Liu, L., Shen, C., and van den Hengel, A. (2015). Mid-level deep pattern mining. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 318)
- Lin, H.-T., Lin, C.-J., and Weng, R. (2007). A note on platt’s probabilistic outputs for support vector machines. *Machine learning*, 68(3):267–276. (cited on page 51)

BIBLIOGRAPHY

- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. (cited on pages 92, 217, 276, 277, 278, 279, and 317)
- Liu, J., Kanazawa, A., Jacobs, D. W., and Belhumeur, P. N. (2012). Dog breed classification using part localization. In *European Conference on Computer Vision (ECCV)*, pages 172–185. (cited on page 191)
- Liu, Y. (2004). Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of Chemical Information and Computer Sciences*, 44:1936–1941. (cited on page 150)
- Long, C., Hua, G., and Kapoor, A. (2013). Active visual recognition with expertise estimation in crowdsourcing. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3000–3007. (cited on pages 151 and 180)
- Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 2, 81, 93, 124, and 250)
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110. (cited on pages 74, 75, and 201)
- Loy, C. C., Hospedales, T. M., Xiang, T., and Gong, S. (2012). Stream-based joint exploration-exploitation active learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1560–1567. (cited on page 150)
- Loy, C. C., Xiang, T., and Gong, S. (2010). Stream-based active unusual event detection. In *Asian Conference on Computer Vision (ACCV)*, pages 161–175. (cited on page 150)
- Lütz, A., Rodner, E., and Denzler, J. (2011). Efficient multi-class incremental learning using gaussian processes. In *Open German-Russian Workshop on Pattern Recognition and Image Understanding (OGRW)*, pages 182–185. (cited on pages 10 and 128)
- Lütz, A., Rodner, E., and Denzler, J. (2013). I want to know more - efficient multi-class incremental learning using gaussian processes. *Pattern Recognition and Image Analysis (PRIA)*, 23(3):402–407. (cited on pages 10 and 128)
- Mac Aodha, O., Campbell, N., Kautz, J., and Brostow, G. (2014). Hierarchical subquery evaluation for active learning on a graph. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 156)

- MacKay, D. J. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press. (cited on pages 77 and 78)
- Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 187)
- Mairal, J., Bach, F., and Ponce, J. (2012). Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(4):791–804. (cited on page 186)
- Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In *International Conference on Machine Learning (ICML)*, pages 689–696. (cited on pages 185 and 186)
- Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2008). Discriminative learned dictionaries for local image analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on page 186)
- Maji, S. and Berg, A. (2009). Max-margin additive classifiers for detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 40–47. (cited on pages 93 and 97)
- Maji, S., Berg, A., and Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on pages 94, 97, 99, and 101)
- Maji, S., Berg, A. C., and Malik, J. (2012). Efficient classification for additive kernel svms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(1):66–77. (cited on page 94)
- Maji, S., Kannala, J., Rahtu, E., Blaschko, M., and Vedaldi, A. (2013). Fine-grained visual classification of aircraft. Technical report. (cited on page 191)
- Malisiewicz, T., Gupta, A., and Efros, A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *IEEE International Conference on Computer Vision (ICCV)*, pages 89–96. (cited on pages 195 and 211)
- Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2013). Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(11):2624–2637. (cited on pages 92 and 129)
- Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *The Royal Society*, 209:415–446. (cited on page 62)

BIBLIOGRAPHY

- Milgram, J., Cheriet, M., and Sabourin, R. (2005). Estimating accurate multi-class probabilities with support vector machines. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1906–1911. (cited on page 178)
- Minka, T. P. (2001). *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology. (cited on pages 69, 223, 224, and 225)
- Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2015). Never-ending learning. In *Conference on Artificial Intelligence (AAAI)*. (cited on page 89)
- Moosmann, F., Triggs, B., and Jurie, F. (2006). Randomized clustering forests for building fast and discriminative visual vocabularies. In *Advances in Neural Information Processing Systems (NIPS)*. (cited on page 185)
- Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press, London, England. (cited on pages 30, 36, 48, 55, 73, and 162)
- Neal, R. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto. (cited on page 67)
- Neal, R. M. (1998). Regression and classification using gaussian process priors. *Bayesian statistics*, 6. (cited on page 69)
- Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7:308–313. (cited on pages 68 and 114)
- Neubert, S. (2003). Some perspectives of interactive constructivism on the theory of education. Technical report, University of Cologne. (cited on page 152)
- Nguyen-Tuong, D., Seeger, M. W., and Peters, J. (2009). Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034. (cited on pages 130 and 139)
- Nickisch, H. and Rasmussen, C. E. (2008). Approximations for binary gaussian process classification. *Journal of Machine Learning Research (JMLR)*, 9:2035–2078. (cited on pages 69, 71, 223, 224, and 225)
- Niemann, H. (1983). *Klassifikation von Mustern*, volume 1. Springer Berlin. (cited on page 24)
- Niemann, H. (1989). *Pattern Analysis and Understanding*. Springer, second edition. (cited on pages 4, 15, and 203)

- Nowak, E., Jurie, F., and Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision (ECCV)*, pages 490–503. (cited on pages 75 and 78)
- Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325. (cited on page 185)
- Osugi, T., Kim, D., and Scott, S. (2005). Balancing exploration and exploitation: a new algorithm for active machine learning. In *International Conference on Data Mining (ICDM)*, pages 330–337. (cited on pages 157 and 164)
- Oxford Dictionaries (2015). "histogram". oxford dictionaries. In *Oxford Dictionaries*. Oxford University Press. (accessed March 24, 2015). (cited on page 96)
- Parikh, D. and Grauman, K. (2011). Interactively building a discriminative vocabulary of nameable attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1681–1688. (cited on pages 151 and 188)
- Parkhi, O., Vedaldi, A., Jawahar, C. V., and Zisserman, A. (2011). The truth about cats and dogs. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1427–1435. (cited on page 184)
- Pelleg, D. and Moore, A. W. (2004). Active learning for anomaly and rare-category detection. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1073–1080. (cited on page 153)
- Pentina, A. and Lampert, C. (2014). A pac-bayesian bound for lifelong learning. In *International Conference on Machine Learning (ICML)*. (cited on page 85)
- Pentina, A., Sharmanska, V., and Lampert, C. H. (2015). Curriculum learning of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 85)
- Perronnin, F. (2008). Universal and adapted vocabularies for generic visual categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(7):1243–1256. (cited on page 186)
- Perronnin, F., Akata, Z., Harchaoui, Z., and Schmid, C. (2012). Towards good practice in large-scale learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 93, 107, 233, and 235)
- Perronnin, F. and Dance, C. (2006). Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on page 77)

BIBLIOGRAPHY

- Perronnin, F., Dance, C., Csurka, G., and Bressan, M. (2006). Adapted vocabularies for generic visual categorization. In *European Conference on Computer Vision (ECCV)*, pages 464–475. (cited on page 185)
- Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, pages 143–156. Springer. (cited on page 77)
- Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on page 295)
- Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on page 205)
- Plackett, R. L. (1950). Some theorems in least squares. *Biometrika*, 37(1/2):pp. 149–157. (cited on pages 130 and 140)
- Platt, J. (2000). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10:61–74. (cited on pages 51 and 198)
- Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press. (cited on pages 48 and 129)
- Press, W., Vetterling, W., Flannery, B., and Teukolsky, S. (1999). *Numerical recipes in C: the art of scientific programming*. Cambridge University Press, 2 edition. (cited on pages 114, 138, and 139)
- Quattoni, A. and Torralba, A. (2009). Recognizing indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 413–420. (cited on pages 92, 215, and 231)
- Quiñonero Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research (JMLR)*, 6:1939–1959. (cited on pages 71, 94, 235, and 236)
- Quiñonero Candela, J., Rasmussen, C. E., and Williams, C. K. (2007). Approximation methods for gaussian process regression. In Bottou, Léon, O. C. D. D. J. W., editor, *Large-scale kernel machines*, pages 203–223. Cambridge, MA: MIT Press. (cited on pages 96 and 108)

- Ramanan, D. (2006). Learning to parse images of articulated bodies. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1129–1136. (cited on page 210)
- Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA. (cited on pages 53, 59, 61, 63, 64, 65, 67, 68, 69, 72, 94, 96, 164, and 287)
- Rasmussen, C. E. and Nickisch, H. (2010). Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research (JMLR)*, 11:3011–3015. (cited on pages 223 and 235)
- Rastegari, M., Farhadi, A., and Forsyth, D. (2012). Attribute discovery via predictable discriminative binary codes. In *European Conference on Computer Vision (ECCV)*, pages 876–889. (cited on page 188)
- Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR-WS)*, pages 512–519. (cited on page 81)
- Rifkin, R. and Klautau, A. (2004). In defense of one-vs-all classification. *Journal of Machine Learning Research (JMLR)*, 5:101–141. (cited on pages 69 and 223)
- Riloff, E., Jones, R., et al. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In *National Conference on Artificial Intelligence*, pages 474–479. (cited on pages 197 and 198)
- Ristin, M., Guillaumin, M., Gall, J., and Gool, L. V. (2014). Incremental learning of ncm forests for large-scale image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3654–3661. (cited on pages 93 and 129)
- Rodner, E. (2011). *Learning from Few Examples for Visual Recognition Problems*. PhD thesis, FSU Jena. (cited on pages 39, 68, 74, 85, 86, 105, and 164)
- Rodner, E., Freytag, A., Bodesheim, P., and Denzler, J. (2012). Large-scale gaussian process classification with flexible adaptive histogram kernels. In *European Conference on Computer Vision (ECCV)*, pages 85–98. (cited on pages 9, 92, 96, 97, and 110)
- Rodner, E., Freytag, A., Bodesheim, P., Fröhlich, B., and Denzler, J. (2015a). Large-scale gaussian process inference with generalized histogram intersection kernels for visual recognition tasks. *International Journal of Computer Vision (IJCV)*. (submitted). (cited on pages 10, 92, 96, 110, 116, 117, 122, and 128)

BIBLIOGRAPHY

- Rodner, E., Simon, M., Brehm, G., Pietsch, S., Wägele, J. W., and Denzler, J. (2015b). Fine-grained recognition datasets for biodiversity analysis. In *CVPR Workshop on Fine-grained Visual Classification (CVPR-WS)*. (cited on page 315)
- Roig, G., Boix, X. and de Nijs, R., Ramos, S., Kühnlenz, K., and Van Gool, L. (2013). Active map inference in crfs for efficient semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2312–2319. (cited on page 171)
- Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443. (cited on page 75)
- Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 309–314. (cited on pages 123, 192, 194, 291, and 295)
- Rowley, H. A., Baluja, S., Kanade, T., et al. (1995). Human face detection in visual scenes. Technical report, Carnegie Mellon University Pittsburgh (CMU). (cited on pages 197 and 206)
- Roy, N. and McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning (ICML)*, pages 441–448. (cited on pages 156, 256, 270, 279, and 281)
- Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: an efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2572. (cited on pages 74 and 201)
- Rühle, J., Rodner, E., and Denzler, J. (2015). Beyond thinking in common categories: Predicting obstacle vulnerability using large random codebooks. In *Machine Vision Applications (MVA)*, pages 198–201. (cited on page 78)
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical Report ICS Report 8506, Institute for cognitive sciences, University of California, San Diego. (cited on page 187)
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, pages 323–533. (cited on page 187)
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42. (cited on pages 80, 92, 220, and 289)

- Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*, 77(1-3):157–173. (cited on page 218)
- Ryan, M. and Martens, G. (1989). *Planning a College Course: A Guidebook for the Graduate Teaching Assistant*. ERIC, Ann Arbor, Mich. (cited on page 152)
- Saad, Y. (2003). *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics (SIAM). (cited on pages 107 and 317)
- Saar-Tsechansky, M. and Provost, F. (2007). Handling missing values when applying classification models. *Journal of Machine Learning Research (JMLR)*, 8:1625–1657. (cited on pages 212 and 291)
- Sadeghi, F., Divvala, S. K., and Farhadi, A. (2015). Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1456–1464. (cited on page 89)
- Sadeghi, M. and Forsyth, D. (2014). 30hz object detection with dpm v5. In *European Conference on Computer Vision (ECCV)*, pages 65–79. (cited on page 206)
- Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5(2):197–227. (cited on page 188)
- Scheirer, W., Rocha, A., Sapkota, A., and Boult, T. (2013). Towards open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36. (cited on page 87)
- Schohn, G. and Cohn, D. (2000). Less is more: Active learning with support vector machines. In *International Conference on Machine Learning (ICML)*, pages 839–846. (cited on pages 154 and 156)
- Schölkopf, B. and Smola, A. (2002). *Learning with kernels: Support Vector Machines, Regularization, Optimization, and beyond*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA. (cited on pages 16, 18, 20, 22, 39, 40, 41, 42, 47, 48, 49, 52, 62, 63, and 105)
- Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse gaussian process regression. In *Workshop on AI and Statistics*, volume 9, page 2003. (cited on page 94)
- Seeger, M. W. (2007). Low rank updates for the cholesky decomposition. Technical report, University of California at Berkeley. (cited on pages 130 and 139)

BIBLIOGRAPHY

- Seo, S., Wallat, M., Graepel, T., and Obermayer, K. (2000). Gaussian process regression: Active data selection and test point rejection. In *Symposium of the German Association for Pattern Recognition (DAGM)*, pages 241–246. (cited on pages 153 and 156)
- Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*. (cited on page 93)
- Settles, B. (2008). *Curious machines: active learning with structured instances*. PhD thesis. (cited on page 268)
- Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. (cited on pages 155, 156, and 317)
- Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics. (cited on page 176)
- Settles, B., Craven, M., and Ray, S. (2008). Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1289–1296. MIT Press. (cited on page 155)
- Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30. (cited on page 94)
- Sharma, P. and Nevatia, R. (2013). Efficient detector adaptation for object detection in a video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3261. (cited on page 198)
- Shawe-Taylor, J., Anthony, M., and Biggs, N. (1993). Bounding sample size with the vapnik-chervonenkis dimension. *Discrete Applied Mathematics*, 42(1):65–73. (cited on page 21)
- Shen, Y., Ng, A., and Seeger, M. (2006). Fast gaussian process regression using kd-trees. In *Advances in Neural Information Processing Systems (NIPS)*. (cited on pages 95 and 108)
- Shotton, J., Winn, J. M., Rother, C., and Criminisi, A. (2009). Textonboost for image understanding - multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision (IJCV)*, 81:2–23. (cited on page 124)

- Sickert, S. (2012). Strategien des aktiven lernens. Diploma thesis, Computer Vision Group, Friedrich Schiller University Jena, Germany. (cited on page 157)
- Sillito, R. R. and Fisher, R. B. (2007). Incremental one-class learning with bounded computational complexity. In *Artificial Neural Networks (ICANN)*, pages 58–67. Springer. (cited on page 129)
- Simon, M. and Rodner, E. (2015). Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*. (cited on pages 189, 292, and 309)
- Simon, M., Rodner, E., and Denzler, J. (2014). Part detector discovery in deep convolutional neural networks. In *Asian Conference on Computer Vision (ACCV)*, pages 162–177. (cited on page 80)
- Singh, S., Gupta, A., and Efros, A. (2012). Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision (ECCV)*, pages 73–86. (cited on pages 189, 190, 195, 199, 297, and 298)
- Sivic, J., Russell, B. C., Zisserman, A., Freeman, W. T., and Efros, A. A. (2008). Unsupervised discovery of visual object class hierarchies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on page 204)
- Smola, A. J. and Bartlett, P. (2001). Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems (NIPS)*. (cited on page 94)
- Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264. (cited on page 94)
- Sontag, E. (1998). Vc dimension of neural networks. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 69–96. Springer-Verlag, Berlin. (cited on page 20)
- Sorenson, H. W. (1970). Least-squares estimation: from gauss to kalman. *IEEE Spectrum*, 7(7):63–68. (cited on page 73)
- Sreekanth, V., Vedaldi, A., Jawahar, C. V., and Zisserman, A. (2010). Generalized RBF feature maps for efficient detection. In *British Machine Vision Conference (BMVC)*. (cited on page 210)
- Su, Y., Allan, M., and Jurie, F. (2010). Improving object classification using semantic attributes. In *British Machine Vision Conference (BMVC)*, volume 10, pages 26–1. (cited on page 188)

BIBLIOGRAPHY

- Sun, J. and Ponce, J. (2013). Learning discriminative part detectors for image classification and cosegmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3400–3407. (cited on pages 230 and 231)
- Sung, K. K. and Poggio, T. (1994). Example based learning for view-based human face detection. Technical report, Massachusetts Institute of Technology (MIT). (cited on pages 197 and 206)
- Sydorov, V., Sakurada, M., and Lampert, C. H. (2014). Deep fisher kernels—end to end learning of the fisher kernel gmm parameters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1402–1409. (cited on page 79)
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*. (cited on page 93)
- Tax, D. and Laskov, P. (2003). Online svm learning: from classification to data description and back. In *Workshop on Neural Networks for Signal Processing (NNSP)*, pages 499–508. (cited on page 129)
- Thrun, S. (1995). Lifelong learning: A case study. Technical report, DTIC Document. (cited on pages 85 and 86)
- Thrun, S. (1996). Is learning the n -th thing any easier than learning the first? In *Advances in Neural Information Processing Systems (NIPS)*, pages 640–646. (cited on page 85)
- Tommasi, T. and Caputo, B. (2009). The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In *British Machine Vision Conference (BMVC)*. (cited on page 85)
- Tong, S. and Koller, D. (2002). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research (JMLR)*, 2:45–66. (cited on pages 154, 157, 158, and 258)
- Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(11):1958–1970. (cited on pages 32 and 92)
- Torresani, L., Szummer, M., and Fitzgibbon, A. (2010). Efficient object category recognition using classemes. In *European Conference on Computer Vision (ECCV)*, pages 776–789. (cited on page 188)

- Tu, Z. and Bai, X. (2010). Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(10):1744–1757. (cited on page 123)
- Tuytelaars, T., Lampert, C. H., Blaschko, M. B., and Buntine, W. (2010). Unsupervised object discovery: A comparison. *International Journal of Computer Vision (IJCV)*, 88(2):284–302. (cited on page 261)
- Tzeng, E., Hoffman, J., Darrell, T., and Saenko, K. (2015). Simultaneous deep transfer across domains and tasks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4068–4076. (cited on page 80)
- Urtasun, R. and Darrell, T. (2008). Sparse probabilistic regression for activity-independent human pose inference. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on pages 95 and 204)
- van de Sande, K. E. A., Gevers, T., and Snoek, C. G. (2010). Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1582–1596. (cited on page 74)
- Van De Weijer, J., Schmid, C., Verbeek, J., and Larlus, D. (2009). Learning color names for real-world applications. *IEEE Transactions on Image Processing (TIP)*, 18(7):1512–1523. (cited on pages 74, 75, 291, and 300)
- Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 9(2579-2605):85. (cited on pages 270, 274, and 277)
- van Gemert, J. C., Veenman, C. J., Smeulders, A. W., and Geusebroek, J.-M. (2010). Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(7):1271–1283. (cited on page 76)
- Vapnik, V. (1998). *Statistical learning theory*, volume 2. Wiley, New York. (cited on pages 22, 39, 49, 50, and 63)
- Vapnik, V. (2000). *The nature of statistical learning theory*. Springer, 2 edition. (cited on page 203)
- Vapnik, V. and Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16(2):264–280. (cited on page 20)
- Vavasis, S. A. (1990). Quadratic programming is in NP. *Information Processing Letters*, 36(2):73–77. (cited on page 48)

BIBLIOGRAPHY

- Vavasis, S. A. (2009). Complexity theory: Quadratic programming. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization, Second Edition*, pages 451–454. Springer. (cited on page 48)
- Vedaldi, A. and Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms. <http://www.vlfeat.org>. (cited on pages 75 and 291)
- Vedaldi, A., Gulshan, V., Varma, M., and Zisserman, A. (2009). Multiple kernels for object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 606–613. (cited on page 210)
- Vedaldi, A. and Zisserman, A. (2010). Efficient additive kernels via explicit feature maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3539 – 3546. (cited on page 94)
- Vedaldi, A. and Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(3):480–493. (cited on pages 60, 94, 210, 291, and 296)
- Vedaldi, A. and Zisserman, A. (2015). Vgg convolutional neural networks practical (release 2015a). (cited on page 187)
- Vezhnevets, A., Buhmann, J. M., and Ferrari, V. (2012). Active learning for semantic segmentation with expected change. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3162–3169. (cited on pages 170 and 171)
- Vidal-Naquet, M. and Ullman, S. (2003). Object recognition with informative features and linear classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 281–288. (cited on page 189)
- Vijayanarasimhan, S. and Grauman, K. (2009). What’s it going to cost you? : Predicting effort vs. informativeness for multi-label image annotations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2262–2269. (cited on page 151)
- Vijayanarasimhan, S. and Grauman, K. (2011a). Cost-sensitive active visual category learning. *International Journal of Computer Vision*, 91:24–44. (cited on pages 151 and 156)
- Vijayanarasimhan, S. and Grauman, K. (2011b). Large-scale live active learning: Training object detectors with crawled data and crowds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 97–114. (cited on page 151)
- Vijayanarasimhan, S., Jain, P., and Grauman, K. (2014). Hashing hyperplane queries to near points with applications to large-scale active learning. *IEEE Transactions*

- on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(2):276–288. (cited on pages 150 and 154)
- Viola, P. and Jones, M. (2004). Robust real-time object detection. *International Journal of Computer Vision (IJCV)*, 57(2):137–154. (cited on page 2)
- von Glasersfeld, E. (1989). Cognition, construction of knowledge, and teaching. *Synthese*, 80(1):121–140. (cited on page 152)
- von Goethe, J. W. (2015). *Goethe: Faust Acts I and II Complete*. Poetry In Translation. (<http://www.poetryintranslation.com/Downloadcat/Pdf/Faustpdf.zip>, accessed at 2015-11-04). (cited on page 213)
- Vondrick, C., Khosla, A., Malisiewicz, T., and Torralba, A. (2013). Hog-gles: Visualizing object detection features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8. (cited on page 78)
- Wah, C. and Belongie, S. (2013). Attribute-based detection of unfamiliar classes with humans in the loop. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–786. (cited on pages 88 and 188)
- Wah, C., Branson, S., Perona, P., and Belongie, S. (2011a). Multiclass recognition and part localization with humans in the loop. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2524–2531. (cited on pages 191 and 208)
- Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011b). The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology. (cited on pages 184, 190, 191, 209, 217, 218, 290, 292, 295, 296, 303, and 309)
- Wang, G., Hoiem, D., and Forsyth, D. (2012). Learning image similarity from flickr groups using fast kernel machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(11):2177–2188. (cited on page 94)
- Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3360–3367. (cited on page 76)
- Wei, K., Iyer, R., and Bilmes, J. (2015). Submodularity in data subset selection and active learning. In *International Conference on Machine Learning (ICML)*. (cited on page 317)
- Weston, J. and Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In Verleysen, M., editor, *European Symposium on Artificial Neural Networks (ESANN)*, Brussels. (cited on page 49)

BIBLIOGRAPHY

- Williams, C. and Rasmussen, C. (1996). Gaussian processes for regression. In *Advances in Neural Information Processing Systems (NIPS)*. MIT. (cited on page 67)
- Williams, C. K. and Barber, D. (1998). Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(12):1342–1351. (cited on pages 69, 70, 223, 224, and 225)
- Williams, C. K. I. and Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. In *International Conference on Machine Learning (ICML)*, pages 1159–1166. (cited on page 111)
- Winn, J., Criminisi, A., and Minka, T. (2005). Object categorization by learned universal visual dictionary. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1800–1807. (cited on page 186)
- Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390. (cited on page 22)
- Wu, J. (2010). A fast dual method for hik svm learning. In *European Conference on Computer Vision (ECCV)*, pages 552–565. (cited on pages 94, 97, 345, 232, and 233)
- Wu, J. (2012). Efficient hik svm learning for image classification. *IEEE Transactions on Image Processing*, pages 4442–4453. (cited on page 94)
- Yagnik, J., Strelow, D., Ross, D. A., and Lin, R.-s. (2011). The power of comparative reasoning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2431–2438. (cited on page 206)
- Yang, C., Duraiswami, R., and Davis, L. S. (2005). Efficient kernel machines using the improved fast gauss transform. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1561–1568. (cited on page 95)
- Yang, L., Jin, R., Pantofaru, C., and Sukthankar, R. (2007). Discriminative cluster refinement: Improving object category recognition given limited training data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on page 186)
- Yang, L., Jin, R., Sukthankar, R., and Jurie, F. (2008). Unifying discriminative visual codebook generation with classifier training for object category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on page 186)
- Yang, M. Y. and Förstner, W. (2011a). A hierarchical conditional random field model for labeling and classifying images of man-made scenes. In *Workshop Computer Vision (ICCV-WS)*, pages 196–203. (cited on pages 252 and 254)

- Yang, M. Y. and Förstner, W. (2011b). Regionwise classification of building facade images. In *Photogrammetric Image Analysis*, volume 6952 of *Lecture Notes in Computer Science*, pages 209–220. Springer. (cited on pages 251, 252, and 254)
- Yang, S., Bo, L., Wang, J., and Shapiro, L. (2012). Unsupervised template learning for fine-grained object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3122–3130. (cited on pages 291, 292, 296, 303, and 309)
- Yeh, T. and Darrell, T. (2008). Dynamic visual category learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. (cited on page 129)
- Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. (2015). Understanding neural networks through deep visualization. In *Deep Learning Workshop at the International Conference on Machine Learning (ICML-WS)*. (cited on page 187)
- YouTube (2014). Youtube statistics. (cited on page 145)
- Yu, A. and Grauman, K. (2014). Fine-grained visual comparisons with local learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 192–199. (cited on pages 191 and 204)
- Yu, F., Cao, L., Feris, R., Smith, J., and Chang, S.-F. (2013). Designing category-level attributes for discriminative visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 771–778. (cited on page 188)
- Yuster, R. (2008). Matrix sparsification for rank and determinant computations via nested dissection. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 137–145. (cited on page 111)
- Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. (cited on pages 81 and 187)
- Zhang, H., Berg, A. C., Maire, M., and Malik, J. (2006). Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2126–2136. (cited on page 204)
- Zhang, J., Marszałek, M., Lazebnik, S., and Schmid, C. (2007). Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision (IJCV)*, 73(2):213–238. (cited on page 75)
- Zhang, N., Donahue, J., Girshick, R., and Darrell, T. (2014). Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision (ECCV)*. (cited on pages 93, 189, 190, 191, 292, and 309)

BIBLIOGRAPHY

- Zhang, N., Farrell, R., Iandola, F., and Darrell, T. (2013). Deformable part descriptors for fine-grained recognition and attribute prediction. In *IEEE International Conference on Computer Vision (ICCV)*, pages 729–736. (cited on pages 184, 189, 291, 292, 296, 303, and 309)
- Zheng, W.-S., Li, X., Xiang, T., Liao, S., Lai, J., and Gong, S. (2015). Partial person re-identification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4678–4686. (cited on page 2)
- Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems (NIPS)*, pages 487–495. (cited on page 2)

List of Figures

1.1	Given an image, which story would you tell with a single word, a whole sentence, or even within a paragraph?	2
1.2	Steps towards a story-telling machine	3
1.3	The lifelong learning cycle for visual recognition	7
1.4	Large-scale learning for visual recognition	10
1.5	Incremental learning for visual recognition	10
1.6	Active learning for visual recognition	11
1.7	Feature learning for visual recognition	12
2.1	VC-dimension of linear models in $D = 2$	21
2.2	Drawbacks in empirical risk minimization and visualization of the regularized risk principle.	23
2.3	Likelihood function, prior belief, and a-posteriori function	28
2.4	Classification with KNN classifiers	32
2.5	Normal distribution classifier and LDA model	34
2.6	Linear separations and SVM solution	37
2.7	Geometric interpretation of SVM and margins	38
2.8	Examples for kernelized soft margin classifiers	47
2.9	Functions sampled from a Gaussian process prior	54
2.10	Functions sampled from a Gaussian process after observing data	56
2.11	Predictive distributions of a Gaussian process after observing noisy data	58
2.12	Famous local feature descriptors	75
2.13	Learned prototypes in the bag-of-visual-words approach are hardly human understandable	76
2.14	Feature hierarchies with CNN models	80
2.15	Detecting objects and parts with deformable part models	83
2.16	Attributes as semantically rich descriptions	84
2.17	Motivation for transfer learning	86
2.18	Illustration of closed-world and open-space recognition.	87

LIST OF FIGURES

2.19	Commonsense knowledge expresses relations between categories	88
3.1	Predictive mean estimates are piecewise linear for GP regression models equipped with κ_{HIK}	100
3.2	Quantization of inputs with equidistant placement of interval borders	102
3.3	Predictive mean estimates $\tilde{\mu}_*$ with quantized test inputs	103
3.4	Interplay of key ingredients for training and optimization of hyperparameters with FMKs	114
3.5	Different approximations of predictive variance estimates using GP regression models equipped with κ_{HIK}	120
3.6	A general pipeline for semantic segmentation	124
3.7	Semantic segmentation with scalable Gaussian process models	125
3.8	Semantic segmentation with GP – a qualitative example	126
4.1	Visualizing incremental learning with plain GP models: update and increment	134
4.2	Visualizing incremental learning with plain GP models (cont.): multiple increments and decrements	137
5.1	Active learning with strategies based on Gaussian processes as introduced by Kapoor et al. (2010).	159
5.2	Deriving simple query techniques from a Gaussian process’ predictive distribution is likely prone to outliers.	159
5.3	A motivation for active learning with estimated model changes (EMC)	163
5.4	Active learning with estimated model changes (EMC)	165
5.5	A visualization of active learning with estimated model output changes (EMOC)	169
5.6	Active learning with estimated model output changes (EMOC)	174
5.7	Visualizing problems in active learning with unnameable instances	180
6.1	Visualization of high intra-class variance and small inter-class variance in fine-grained recognition scenarios.	190
6.2	Answers for questions in Fig. 6.1	192
6.3	Overview of our seeding step for unsupervised discovery of mid-level features	193
6.4	Background masking using GrabCut or human annotations.	194
6.5	Exemplary seeding results for unsupervised patch discovery	194
6.6	Visualization of learned patch detectors from single positive examples.	196
6.7	Results of bootstrapping patch detectors.	199
6.8	Bootstrapping patch detectors in supervised and unsupervised settings	200

6.9	Encoding images with responses of learned patch detectors	201
6.10	Do we need to know how to differentiate between 20 sparrows, when we currently spot a woodpecker?	203
6.11	Local learning for exemplar-specific representations and models . .	205
6.12	Selecting objects with similar shape and overall appearance	206
6.13	Analysis of relative part offsets in the CUB-2011/200 dataset	209
6.14	Transferring part annotations from known to unknown objects based on similar pose	211
6.15	Non-parametric transfer of part annotations	212
7.1	The ImageNet dataset	214
7.2	The Caltech-256 dataset	215
7.3	The 15Scenes dataset	215
7.4	The USPS dataset	216
7.5	The Caltech-UCSD Birds-200-2011 dataset	217
7.6	The Caltech-UCSD Birds-200-2011 dataset with part annotations .	218
7.7	Datasets for semantic segmentation: eTRIMS and LabelMeFacade .	219
7.8	Comparing GP label regression with techniques for approximate inference in binary classification scenarios	224
7.9	Comparing GP label regression with techniques for approximate inference in multi-class classification scenarios	225
7.10	Multi-class classification using all 1,000 categories of ImageNet. . .	229
7.11	Performance of GP/HIK on 15Scenes with CNN features extracted from different <code>relu</code> layers and a varying number of training examples per class.	230
7.12	Evaluation of the computation time and convergence of linear solvers	233
7.13	Evaluation of model regularization by early stopping of the GP/HIK conjugate gradient method.	234
7.14	Comparison of GP/HIK with the FITC method on the 15Scenes dataset with CNN features.	236
7.15	Effect of different quantization levels on classification accuracy, required computation times during training, and for inference. . . .	237
7.16	Evaluating the upper bound of the negative log-likelihood	239
7.17	Evaluation of model regularization by early stopping and with hyperparameter optimization	243
7.18	Distributions of synthesized features for evaluating feature relevance determination abilities	244
7.19	Learned relative weights for each dimension for $\kappa_{\text{HIK-ARD}}$ in a synthetic experiment	245
7.20	Accuracy as a function of the training set size when dimension-specific weights are optimized for $\kappa_{\text{HIK-ARD}}$	246

LIST OF FIGURES

7.21 Tightness of GP/HIK variance approximations on the 15Scenes dataset with relu7 features	249
7.22 Normalized variance approximations on the 15Scenes dataset with relu7 features	250
7.23 Qualitative results for semantic segmentation with GP/HIK	253
7.24 Visual analysis of active learning criteria in a 2D scenario	257
7.25 Evaluating active learning criteria in binary classification tasks derived from ImageNet	260
7.26 Evaluating active learning criteria in binary classification tasks derived from Caltech-256	262
7.27 Evaluating active learning with $Q_{GP-\Delta f}$ in one-vs-all binary classification tasks derived from Caltech-256	263
7.28 Visual analysis of selected queries in one-vs-all experiments on Caltech-256	264
7.29 Qualitative analysis of a single one-vs-all experiments on Caltech-256	265
7.30 Benefits of accessible ground truth labels when computing active learning scores	267
7.31 Quantitative benefits of density-based re-weighting for active learning criteria	268
7.32 Visualization of USPS handwritten digits with unnameable instances	270
7.33 Results of active learning and class discovery for multi-class digit recognition	272
7.34 Video: active query selection in the presence of unnameable instances	273
7.35 Visualization of human faces from LFW with unnameable instances	274
7.36 Results of active learning and class discovery for multi-class face identification	275
7.37 Object proposals extracted from MS-COCO	277
7.38 Visualization of object proposals computed from MS-COCO with unnameable instances	277
7.39 Results of active learning and class discovery for multi-class animal categorization	278
7.40 Qualitative results of active learning and class discovery for multi-class animal categorization	280
7.41 Active learning and class discovery versus total time spent on USPS	281
7.42 Active learning and class discovery versus total time spent on LFW	282
7.43 Active learning and class discovery versus total time spent on MS-COCO	283
7.44 Qualitative analysis of individual runs in multi-class active learning experiments	284

7.45	Qualitative analysis of best split within multi-class active learning experiments	285
7.46	Evaluating different loss functions within EMOC for multi-class scenarios	286
7.47	Evaluating different techniques to estimate multi-class probabilities within EMOC for multi-class scenarios	287
7.48	Normalized localization error for part localization with DPMs or part annotation transfer	290
7.49	Localization error for part localization with DPMs or part annotation transfer	290
7.50	Qualitative comparison of part detection with DPM and our part annotation transfer.	293
7.51	Analyzing the influence of part annotation errors on the final classification decisions	294
7.52	Visualizing patch detector responses on training images	299
7.53	Visualizing patch detector responses on unseen test images	299
7.54	Analyzing the matching accuracy for local learning of patch detectors	301
7.55	Dependency of classification accuracy and neighborhood sizes for exemplar-specific mid-level features	302
7.56	Number of discovered mid-level features for exemplar-specific representations	304
7.57	Detection responses of patch detectors for different numbers of neighbors.	306
7.58	Analyzing the combination of mid-level features and part annotation transfer for fine-grained recognition on the CUB-2011 dataset . . .	308
A.1	One-class classification based on estimated model impact	322
A.2	Multi-class novelty detection based on null space projections	322
A.3	Local novelty detection	323
A.4	Approximating GP regression using a diagonal covariance matrix .	323
A.5	A simple pipeline for fine-grained recognition	324
A.6	Interactive image retrieval for biodiversity research	325
A.7	Visualizing quantization effects in bag-of-visual-words models. . .	326

LIST OF FIGURES

List of Tables

3.1	Overview of asymptotic computation times and memory demand for GP label regression in its plain formulation, with (G)HIKs, and when quantizing test inputs	115
3.2	Compact overview of asymptotic computation times and memory demand with focus on number of training examples only.	115
3.3	Summary of different techniques for computing or approximating the predictive variance	122
7.1	Evaluation GP/HIK on 200 binary classification tasks derived from the ImageNet database.	227
7.2	Evaluation of GP/HIK on the 15Scenes dataset with standard BoW features and CNN features.	231
7.3	Evaluating the effect of optimizing hyperparameters for $\kappa_{\text{HIK-POLY}}$ and $\kappa_{\text{HIK-EXP}}$ on the resulting classification accuracy using the 15Scenes dataset	241
7.4	Evaluating the effect of optimizing hyperparameters for $\kappa_{\text{HIK-POLY}}$ and $\kappa_{\text{HIK-EXP}}$ on the resulting classification accuracy using the ImageNet dataset	242
7.5	Computation times needed for computing the predictive variance and our approximations thereof	247
7.6	Quantitative results of semantic segmentation on two different datasets	254
7.7	Summary of visually analyzing active learning criteria on 2D data	259
7.8	Experimental results for active learning in binary classification scenarios	261
7.9	Experimental results for active learning and class discovery in multi-class scenarios	279
7.10	Mean accuracy results on CUB-2011.	292
7.11	Analyzing the impact of individuals steps of the fine-grained recognition pipeline	293
		387

LIST OF TABLES

7.12 Mid-level features for fine-grained recognition on the CUB-2011 dataset	296
7.13 Influence of individual steps during patch discovery on the classification accuracy	298
7.14 Exemplar-specific mid-level features for fine-grained recognition on the CUB-2011 dataset	303
7.15 Combining mid-level features and part annotation transfer for fine-grained recognition on the CUB-2011 dataset	309

List of Definitions, Theorems, and Lemmas

2.1	Theorem (A lower bound on sufficient sample size)	21
2.2	Theorem (A generalization bound for SVM and linear separable data without margin errors)	40
2.3	Theorem (A generalization bound for SVM for linear separable data with margin errors)	41
2.4	Definition (Stochastic process)	52
2.5	Definition (Gaussian process)	53
2.6	Definition ((Positive definite) kernel)	59
2.7	Theorem (Mercer's theorem)	62
2.8	Corollary (Kernel trick)	62
2.9	Theorem (Representer theorem)	63
3.1	Theorem (Worst-case analysis for the error in the predictive mean with quantized inputs)	104
3.2	Theorem (A sharper worst-case analysis for the error in the predictive mean with quantized inputs)	104
3.3	Lemma (An upper bound for a matrix's determinant by Bai and Golub (1997))	111
4.1	Lemma (Recursive Matrix Inversion with Woodbury's Formula (Hager, 1989))	131
4.2	Theorem (Closed Form Update of GP Regression Weights α)	140
B.1	Lemma (Monotonicity of $\text{ub}(\beta, \mu_1, \mu_2)$ in Eq. (3.25) for $\beta = 1$)	330
B.2	Lemma (Multiplicative scaling of $\text{ub}(\beta, \mu_1, \mu_2)$ in Eq. (3.25))	331
B.3	Theorem (Upper bound with $\tilde{\mu}_2$)	333

LIST OF DEFINITIONS, THEOREMS, AND LEMMAS

Notations

$\mathbb{N} = \{0, 1, 2, \dots\}$	Set of all natural numbers
$\mathbb{R}$	Set of all real numbers
$\mathbb{N}_+, \mathbb{R}_+$	Sets restricted to strictly positive numbers, <i>i.e.</i> , without zero
$\mathbb{N}_*, \mathbb{R}_*$	Sets restricted to positive numbers with zero
$a, b, c, \dots$	Scalars: lower case and not bold
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$	Vectors: lower case and bold, notation assumes column vectors, <i>i.e.</i> , $D \times 1$ matrices
$a(d)$	Entry of vector $\mathbf{a}$ in dimension d
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \dots$	Matrices: upper case and bold
$A(i, j)$	Entry of matrix $\mathbf{A}$ in row i and column j
Ω	Input feature space
D	Dimensionality of input feature space Ω
$\mathbf{x} \in \Omega$	A single input, also referred to as sample, observation, example, or pattern.
$\mathcal{Y}$	Space of possible target values, <i>e.g.</i> , $\mathcal{Y} = \mathbb{R}$ for regression, or $\mathcal{Y} = \{-1, 1\}$ for binary classification
$y \in \mathcal{Y}$	Target value for input $\mathbf{x}$. Called label in classification scenarios.
N	Number of labeled training examples
M	Number of known classes in classification scenarios
$\mathcal{L}$	A collected and labeled dataset $\mathcal{L} \subset \Omega \times \mathcal{Y}$, $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$
$\mathbf{x}_* \in \Omega$	Previously unseen test example
y_*	Label for previously unseen test example
$\langle \cdot, \cdot \rangle$	Inner product of two vectors
$d(\cdot, \cdot)$	Distance function for vectors

Notations

$p(\cdot)$	probability, <i>i.e.</i> , probability density or probability mass (for continuous and discrete inputs, respectively). For the sake of brevity, the random variable in $p(\mathbb{X} = x)$ is usually omitted and shortened to $p(x)$
$\delta(\cdot)$	Dirac delta function
$\delta_{\cdot, \cdot}$	Kronecker delta
$\text{sgn}(\cdot)$	Sign function
$\mathcal{O}(\cdot)$	Big O notation for analyzing asymptotic growth of functions regarding increasing input sizes, special case of Landau notations
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$
$\det(\mathbf{A})$	Determinant of matrix $\mathbf{A}$
$\text{rk}(\mathbf{A})$	Rank of matrix $\mathbf{A}$
$\text{tr}(\mathbf{A})$	Trace of matrix $\mathbf{A}$
$\mathbb{H}$	Space of possible mappings from Ω to $\mathcal{Y}$
$h(\cdot)$	A possible mapping $h \in \mathbb{H}$
$g(\cdot)$	Unknown ground truth relation
$h_*(\cdot)$	Best learnable mapping from infinite data
$\hat{h}(\cdot)$	Model, a specific mapping learned from finite data $\mathcal{L}$
$f(\cdot)$	Intermediate or latent function for evaluating h
$\mathcal{L}_f(\cdot, \cdot)$	Loss function measuring similarity of mappings
$\mathbb{F}$	A family $\mathbb{F} \subset \mathbb{H}$ of possible mappings, <i>e.g.</i> , all linear mappings
$\mathcal{R}(h)$	Risk of hypothesis h (see Section 2.1.2.1)
$\mathcal{R}_{\text{emp}}(h, \mathcal{L})$	Empirical risk evaluated on $\mathcal{L}$ (see Section 2.1.2.2)
$\mathcal{R}_{\text{reg}}(h, \mathcal{L})$	Regularized risk (see Section 2.1.2.3)
$\mathcal{C}(h)$	Complexity term for computing $\mathcal{R}_{\text{reg}}$
ν	Trade-off parameter for regularization (error vs. complexity)
p_ϵ	Probability for future classification errors
θ	Collection of parameters of a parametric model
θ_*	Parameters after optimization, <i>e.g.</i> , θ_*^{ML} or θ_*^{MAP}
$\mathcal{N}(\mu, \Sigma)$	Multi-variate Gaussian distribution specified by mean vector μ and covariance matrix Σ

$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Probability density of $\mathbf{x}$ with respect to a multi-variate Gaussian distribution
$\mathbf{w}$	Weight vector of linear models (see Eq. (2.46))
$\boldsymbol{\mu}_{-1}, \boldsymbol{\mu}_1$	Mean vector of training samples with negative or positive label
$\boldsymbol{\Sigma}_0$	Covariance matrix computed from all training examples
$f_{\mathbf{w},b}$	Linear separation with normal vector $\mathbf{w}$ and offset b
$\text{mg}_{\mathcal{L}}(f_{\mathbf{w},b})$	Margin of the SVM solution $f_{\mathbf{w},b}$ with respect to $\mathcal{L}$
C	Trade-off parameter for soft-margin SVMs
$\boldsymbol{\alpha}$	Dual solution of weight vector $\mathbf{w}$
ξ_i	Slack variable for $\mathbf{x}_i$ in soft-margin SVMs
$\mathcal{H}$	Hilbert space of possibly higher dimension than Ω .
$\Phi : \Omega \rightarrow \mathcal{H}$	Embedding of inputs in high dimensional space
$\kappa : \Omega \times \Omega \rightarrow \mathbb{R}$	Kernel function (see Theorem 2.7) with associated reproducing kernel Hilbert space $\mathcal{H}_{\kappa}$
σ	Bandwidth parameter of an RBF kernel
$\mathbf{F}$	Stochastic process (see Definition 2.4)
$m(\cdot)$	Mean function of a Gaussian process (see Definition 2.5)
$\kappa(\cdot, \cdot)$	Covariance function of a Gaussian process (see Definition 2.5)
$\mathbf{K}$	Kernel matrix or GP covariance matrix computed from all training examples
$\mathbf{k}_*$	Vector of kernel values between training inputs and test input $\mathbf{x}_*$
k_{**}	Self-similarity $\kappa(\mathbf{x}_*, \mathbf{x}_*)$
ε	Noise process on labels
σ_n^2	Variance of additive White Gaussian noise
$\tilde{\mathbf{K}}$	Regularized kernel matrix $(\mathbf{K} + \sigma_n^2 \mathbf{I})$ for noisy labels
μ_*	Predictive GP mean for test sample $\mathbf{x}_*$ (see Eq. (2.99))
σ_*^2	Predictive GP variance for test sample $\mathbf{x}_*$ (see Eq. (2.100))

Notations

Acronyms

A

ARD automatic relevance determination

ARR average recognition rate

AUC area under an ROC curve

B

BoW bag-of-visual-words

C

CAPU coarse approximation of the predictive uncertainty

CG conjugate gradients

CN color name

CNN convolutional neural network

COLT computational learning theory

D

DPM deformable parts model

F

FAPU fine approximation of the predictive uncertainty

FMK fast multiplication kernel

G

GMM Gaussian mixture model

GP Gaussian process

H

HIK histogram intersection kernel

Acronyms

HOG histogram of oriented gradients

I

i.i.d. independent and identically distributed

ILS iterative linear solver

K

KNN K-nearest neighbor

L

LDA linear discriminant analysis

LUT look-up table

M

MAP maximum a posteriori

MKL multiple kernel learning

ML maximum likelihood

MVM matrix vector multiplication

O

OCC One-class classification

ORR overall recognition rate

P

PAC probably approximately correct

psd positive semi-definite

PUP precise uncertainty prediction

Q

q-CAPU coarse approximation of the predictive uncertainty with quantized inputs

QP quadratic program

R

RBF radial basis function

RKHS reproducing kernel Hilbert space

ROC receiver operating characteristic

S

SIFT scale-invariant feature transformation

SMO sequential minimal optimization

SVDD support vector data description

SVM support vector machine

W

w.l.o.g. without loss of generality

Acronyms

Index

A

active learning 6, 145, 147, 312, 313
 exploitative 153
 explorative 153
 myopic 149, 150
 pool-based 150
 query synthesis 150
 stream-based 150
ARD *see* automatic relevance determination
area under an ROC curve 221, 224, 259, 395
Arnoldi iteration 112
ARR *see* average recognition rate
attribute 81
AUC *see* area under an ROC curve
automatic relevance determination 67, 395
average recognition rate 219, 225, 230, 252, 270, 395

B

back-propagation 187
bag-of-visual-words 75–79, 82, 96, 185, 214, 395
Bayes classifier 24
Bayesian perspective 15, 24
bias-variance trade-off 22
binary classification 14, 69, 161

BOW *see* bag-of-visual-words
Brownian motion 61

C

canonical hyperplane 39
CAPU *see* coarse approximation of the predictive uncertainty
category 13
CG *see* conjugate gradients
Cholesky decomposition 138
closed-world assumption 4, 51, 87
CN *see* color name
CNN *see* convolutional neural network
coarse approximation of the predictive uncertainty 121, 122, 143, 161, 247–250, 341, 395
coarse approximation of the predictive uncertainty with quantized inputs 121, 122, 143, 161, 247, 249, 396
color name 74, 195, 395
COLT *see* computational learning theory
commonsense knowledge 88
computational learning theory 20, 395
conjugate gradients 107, 108, 232, 395
constructivism 152
convex 42, 48
convolutional neural network 79, 395
coordinate descent 107
cost function 146

cross-validation 23, 49, 221

D

data update 127, *see* incremental learning

deep learning 79, 186, 187

deformable parts model 81, 82, 96, 208, 289, 395

density

empirical density 167

discovery 5, 269

discriminative approach 25

domain adaptation 6

downhill-simplex algorithm 68, 114

DPM *see* deformable parts model

E

early stopping 107, 233

empirical Bayes 66

empirical distribution 18

estimated model change 161, 255

estimated model output change 167, 255, 284, 286, 321

Expectation-Propagation 69, 223–225

expected cost 16, *see* risk

F

FAPU *see* fine approximation of the predictive uncertainty

fast multiplication kernel 98, 101, 110, 111, 114, 395

feature learning 7, 295, 300, 307

fine approximation of the predictive uncertainty 119, 122, 143, 161, 247–250, 341, 395

fine-grained recognition 190, 191, 288, 314, 324

fine-tuning 187, 231

Fisher's linear discriminant 36

flat-5 error 220

FMK *see* fast multiplication kernel

frequentist perspective 15, 16

Frobenius norm 111

G

Gaussian mixture model 34, 185, 395

Gaussian Process 8, 23, 53, 324, 395

generative approach 25

geometrical margin 39

GMM *see* Gaussian mixture model

GP *see* Gaussian Process

gradient descent 106, 187

grid search 49

ground truth function 15

H

HIK *see* histogram intersection kernel

histogram 96

histogram intersection kernel 61, 93, 395

histogram of oriented gradients 74, 75, 96, 195, 396

HOG *see* histogram of oriented gradients

homogeneous noise *see* noise

hyperparameter 66

hyperprior 66

hypothesis 15

I

i.i.d. *see* independent and identically distributed

ILS *see* iterative linear solver

independent and identically distributed 27, 28, 85, 294, 396

inducing variable 94

informativeness 146, 149

iterative linear solver 95, 106, 233, 247, 317, 396

K

kernel 47, 59
 additive kernel 60
 homogeneous kernel 60
 isotropic kernel 60
 positive definite kernel 59, 396
 stationary kernel 60

kernel trick 62, 389

K-means 77, 184, 185

KNN *see* K -nearest neighbor

kriging 73

L

label regression 69, 223

Lagrange multiplier 42

Laplace approximation 69, 223–225

LDA *see* linear discriminant analysis

learning 4, 13, 15

 decremental learning 127

 incremental learning 5, 127, 313

 large-scale learning 5, 91

 lifelong learning 4, 5, 7, 15, 85, 86

 online learning 129

 supervised learning 5, 15

likelihood 26, 30

linear discriminant analysis 35, 37, 396

local feature 73

local learning 204, 323

look-up table 102, 109, 396

loss function 16, 167

 log-loss 156, 178

L_1 -loss 17

L_2 -loss 17, 18, 156

 0/1-loss 16, 18, 156, 219

 Hinge loss 45

LU decomposition 138

LUT *see* look-up table

M

MAP *see* maximum a posteriori

margin 37

marginal likelihood 29

matrix vector multiplication 95, 233, 339, 396

maximum a posteriori 28, 396

maximum likelihood 26, 28, 396

mean squared error 18

Mercer's theorem 62, 389

misclassification rate 18

MKL *see* multiple kernel learning

ML *see* maximum likelihood

model 15, 19

model update 127, *see* incremental learning

 direct update 133

 iterative update 133

multi-class classification 14, 49, 70, 176

multiple kernel learning 67, 68, 396

MVM *see* matrix vector multiplication

N

K -nearest neighbor 19, 31, 32, 51, 396

Nelder-Mead algorithm *see* downhill-simplex algorithm

noise 56

 homogeneous noise 57

 round-off noise 69

 white Gaussian noise 57, 64

non-parametric 30

novelty detection 5, 153, 321

O

object retrieval 205, 324

OCC *see* One-class classification

Occam's razor 22

One-class classification 321, 396

one-vs-all 50, 70, 125, 136, 178, 224, 225, 247, 248

one-vs-one 50, 70

open space recognition 4, 86

ORR *see* overall recognition rate

overall recognition rate 219, 252, 396
 overfitting 22

P

PAC *see* probably approximately correct
 paired-sample t-test 222
 part 81
 patch 189
 patch detector 189
 plug-in principle 26, 29, 33, 51
 positive definite 59
 positive semi-definite 59–61, 396
 posterior 28
 precise uncertainty prediction 117, 122,
 142, 160, 247–249, 396
 preconditioning 107
 prior 28, 30
 probably approximately correct 20, 40,
 396
 psd *see* positive semi-definite
 PUP *see* precise uncertainty prediction

Q

q-CAPU *see* coarse approximation of
 the predictive uncertainty with
 quantized inputs
 QP *see* Quadratic program
 Quadratic program 48, 396
 query function 147

R

radial basis function 47, 396
 RBF *see* radial basis function
 receiver operating characteristic 221,
 396
 regression 14
 representer theorem 63, 389
 reproducing kernel Hilbert space 63, 396
 residual 106

risk

empirical risk 18, 148
 estimated empirical risk 155
 expected risk 16, 148
 regularized risk 23
 structured risk 23

RKHS *see* reproducing kernel Hilbert
 space

ROC *see* receiver operating characteris-
 tic

S

scale-invariant feature transformation
 74, 75, 397

Schur complement 132, *see* Woodbury's
 formula

segmentation

semantic segmentation 123
 unsupervised segmentation 123

sequential minimal optimization 48,
 129, 397

shallow learning 79

SIFT *see* scale-invariant feature transfor-
 mation

sign operator 35

slack variable 44

SMO *see* sequential minimal optimiza-
 tion

squashing function 50

stochastic process 52

story-telling machine 3

support vector 44

support vector data description 129, 397

support vector machine 19, 23, 36, 37,
 129, 397

SVDD *see* support vector data descrip-
 tion

SVM *see* support vector machine

T

- training from scratch *see* incremental learning
- transfer
 - annotation transfer 6, 207, 210, 288, 307
 - knowledge transfer 6, 85, 129
- transfer function 69
- two-sample t-test 222
- type-2 maximum likelihood 66
- V**
 - VC-dimension 20, 21, 46, 381
- visual perception 1
- W**
 - warm start 142, 143
 - white Gaussian noise *see* noise
 - Wiener process 61
 - Wilcoxon signed rank test 222
 - Woodbury's formula 131
- Z**
 - zero-mean assumption 53, 62, 160

Index

Curriculum Vitae

Personal

Name:	Dipl.-Inf. Alexander Freytag
Date of birth:	November 20, 1988
Place of birth:	Erfurt, Germany
Name at birth:	Alexander Lütz
Marital status:	Married
Nationality	German

Education and Qualifications

09/2016 (defended)	Ph.D. in computer vision with honors (Dr. rer. nat., summa cum laude), Friedrich Schiller University Jena, Germany
03/2011	Diploma degree in computer science (Dipl.-Inf.) with honors (best achievable grade in all subjects), Friedrich Schiller University Jena,, Germany
05/2006	German abitur (High School diploma equivalent), grade: 1.1 (A+ equivalent), Albert-Schweitzer-Gymnasium Erfurt (specialized for mathematics and natural sciences), Germany

CURRICULUM VITAE

Scientific Short CV

since 10/2016	Research Scientist for Machine Learning and Computer Vision at Carl-Zeiss AG, Corporate Research and Technology
10/2014 – 09/2016	Research coordinator at the Michael Stifel Center Jena (MSCJ) for Data-driven and Simulation Science
06/2013 – 08/2013	Visiting researcher in the vision group of the International Computer Science Institute (ICSI), Berkeley, California, USA, which is headed by Trevor Darrell
03/2011 – 09/2016	Research associate and Ph.D. student in the Computer Vision Group, Friedrich Schiller University Jena, Germany, which is headed by Joachim Denzler
05/2009 – 03/2010	Student research assistant in the Computer Vision Group, Friedrich Schiller University Jena, Germany

Honors and Awards

2014	Best Poster Award on ICPR FEAST 2014
2014	<i>Magister Honestus</i> awarded by Albert-Schweitzer-Gymnasium in Erfurt
2012	Best Paper Honorable Mention Award on ACCV 2012
2012	Exam award for best diploma of the year 2011/2012
2011	Diploma with distinction and the highest grade in all subjects
2009	Award for best intermediate diploma of the year 2008

Travel Grants and Scholarships

2016	CVF grant to attend the Doctorial Consortium at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
------	---

2015	DAAD Travel Grant to attend the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
2015	Volkswagen Stiftung Travel Grant to attend the Herrenhäuser Conference on Big Data in a Transdisciplinary Perspective
2013	DAAD scholarship for a research-visit of the International Computer Science Institute (ICSI), Berkeley, USA
2009 – 2011	Scholarship holder of the German National Academic Foundation

Organized Workshops

07/2017	<i>Workshop on “Continuous and Open-Set Learning”</i> (IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, Hawaii)
09/2016	<i>Autumn School on “Dynamics of natural (eco)systems: theory and applications”</i> (Michael Stifel Center for Data-driven and Simulation Science, Jena, Germany)
02/2016	<i>Retreat on “Cross-disciplinary exchange for data-driven and simulation science”</i> (Michael Stifel Center for Data-driven and Simulation Science, Großliebrungen, Germany)
06/2015	<i>Workshop on “Tag der Forschung – Wissen aus Daten, Daten aus Wissen”</i> (Friedrich Schiller University Jena, Jena, Germany)

Oral Presentations and Invited Talks

10/2016	<i>“Tutorial on Deep Learning”</i> (Michael Stifel Center for Data-driven and Simulation Science, Jena, Germany)
---------	---

CURRICULUM VITAE

- 09/2016 *“Chimpanzee Faces in the Wild: Log-Euclidean CNNs for Predicting Identities and Attributes of Primates”*
(German Conference on Pattern Recognition (GCPR), Hannover, Germany)
- 10/2015 *“Interactive Image Retrieval for Biodiversity Research”*
(German Conference on Pattern Recognition (GCPR), Aachen, Germany)
- 03/2015 *“Lifelong Learning using Big Data for Visual Recognition Systems”*
(Herrenhausen Conference: “Big Data in a Transdisciplinary Perspective”, Hannover, Germany)
- 01/2015 *“Active Learning (for Lifelong Learning in Visual Recognition Systems)”*
(invited talk at the 5th IST Symposium on Computer Vision and Machine Learning, Vienna, Austria)
- 10/2014 *“Fine-grained Recognition and Patch Discovery”*
(invited talk at the Workshop on Learning, Modelling, and Inference in Visual Data, Dresden, Germany)
- 10/2014 *“Active Learning (for Visual Object Recognition)”*
(invited talk at the Workshop on Learning, Modelling, and Inference in Visual Data, Dresden, Germany)
- 09/2014 *“Exemplar-specific Patch Features for Fine-grained Recognition”*
(German Conference on Pattern Recognition (GCPR), Münster, Germany)
- 04/2014 *“Intelligente Bildverarbeitung – Wie man Robotern das Sehen beibringt”*
(invited talk at the Tage der Wissenschaften und Kultur of Albert-Schweitzer-Gymnasium Erfurt, Spezialschulenteil, Erfurt, Germany)
- 03/2014 *“Computer Vision Research in Jena”*
(invited talk at the STEINBEIS SpectroNet Collaboration Forum, Ilmenau, Germany)
- 01/2014 *“Adaptive and Active Lifelong Learning for Visual Recognition Systems”*
(invited talk at the 1st DAAD FITweltweit Alumni Veranstaltung, Saarbrücken, Germany)

- 01/2014 *“Where to look first – Autonomous Experimentation”*
(Cosre Anwender-Workshop, Jena, Germany)
- 09/2013 *“Labeling examples that matter: Relevance-Based Active Learning with Gaussian Processes”*
(German Conference on Pattern Recognition (GCPR), Saarbrücken, Germany)
- 07/2013 *“Computer, die lebenslang lernen”*
(invited talk at the Albert-Schweitzer-Gymnasiums Erfurt, Spezialschuleteil, Erfurt, Germany)
- 11/2012 *“Rapid Uncertainty Computation with Gaussian Processes and Histogram Intersection Kernels”*
(Asian Conference on Computer Vision (ACCV), Daejeon, South Korea)

Poster Presentations

- 09/2016 *“Lifelong Learning for Visual Recognition Systems”*
(Annual Assembly of the Michael Stifel Center for Data-driven and Simulation Science, Jena, Germany)
- 07/2016 *“Lifelong Learning for Visual Recognition Systems”*
(Doctorial Consortium at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, Nevada, USA)
- 03/2016 *“Image and Signal Analysis for Data-driven Science – Knowledge from Data”*
(Annual Assembly of the Leibniz Institute on Aging, Jena, Germany)
- 10/2015 *“Active Learning and Discovery of Object Categories in the Presence of Unnameable Instances”*
(Nectar Track Session at German Conference on Pattern Recognition (GCPR), Aachen, Germany)
- 10/2015 *“Selecting Influential Examples: Active Learning with Expected Model Output Changes”*
(Nectar Track Session at German Conference on Pattern Recognition (GCPR), Aachen, Germany)

CURRICULUM VITAE

- 06/2015 *“Active Learning and Discovery of Object Categories in the Presence of Unnameable Instances”*
(IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, Massachusetts, USA)
- 06/2015 *“Image and Signal Analysis for Data-driven Science - Knowledge from Data”*
(Tag der Forschung, Friedrich Schiller University, Jena, Germany)
- 03/2015 *“Lifelong Learning using Big Data for Visual Recognition Systems”*
(Herrenhausen Conference: “Big Data in a Transdisciplinary Perspective”, Hannover, Germany)
- 05/2014 *“Digitale Bildverarbeitung”*
(Annual Assembly of the Leibniz Institute on Aging, Jena, Germany)
- 09/2014 *“Selecting Influential Examples: Active Learning with Expected Model Output Changes”*
(European Conference on Computer Vision (ECCV), Zürich, Switzerland)
- 09/2014 *“Birds of a Feather Flock Together - Local Learning of Mid-level Representations for Fine-grained Recognition”*
(ECCV Workshop on Parts and Attributes (ECCV-WS), Zürich, Switzerland)
- 06/2014 *“Nonparametric Part Transfer for Fine-grained Recognition”*
(IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, Ohio, USA)
- 09/2013 *“Open-Set Lifelong Learning for Visual Recognition Problems”*
(Nectar Track Session at German Conference on Pattern Recognition (GCPR), Saarbrücken, Germany)
- 06/2013 *“Kernel Null Space Methods for Novelty Detection”*
(IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, Oregon, USA)
- 11/2012 *“Large-Scale Gaussian Process Classification with Flexible Adaptive Histogram Kernels”*
(Asian Conference on Computer Vision (ACCV), Daejeon, South Korea)

10/2012

“Large-Scale Gaussian Process Classification with Flexible Adaptive Histogram Kernels”

(European Conference on Computer Vision (ECCV), Florenz, Italien)

CURRICULUM VITAE

List of Own Publications

- Bodesheim, P., **Freytag, A.**, Rodner, E., and Denzler, J. (2013a). Approximations of gaussian process uncertainties for visual recognition problems. In *Scandinavian Conference on Image Analysis (SCIA)*, pages 182–194.
- Bodesheim, P., **Freytag, A.**, Rodner, E., and Denzler, J. (2013b). An efficient approximation for gaussian process regression. Technical report, Computer Vision Group, Friedrich Schiller University Jena, Germany. Technical Report TR-FSU-INF-CV-2013-01.
- Bodesheim, P., **Freytag, A.**, Rodner, E., and Denzler, J. (2015). Local novelty detection in multi-class recognition problems. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 813–820.
- Bodesheim, P., **Freytag, A.**, Rodner, E., Kemmler, M., and Denzler, J. (2013c). Kernel null space methods for novelty detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3374–3381.
- Bodesheim, P., Rodner, E., **Freytag, A.**, and Denzler, J. (2012). Divergence-based one-class classification using gaussian processes. In *British Machine Vision Conference (BMVC)*, pages 50.1–50.11.
- Denzler, J., Rodner, E., Bodesheim, P., and **Freytag, A.** (2013). Beyond the closed-world assumption: The importance of novelty detection and open set recognition. In *GCPR Workshop on Unsolved Problems in Pattern Recognition (GCPR-WS)*.
- Freytag, A.**, Fröhlich, B., Rodner, E., and Denzler, J. (2012a). Efficient semantic segmentation with gaussian processes and histogram intersection kernels. In *International Conference on Pattern Recognition (ICPR)*, pages 3313–3316.
- Freytag, A.**, Rodner, E., Bodesheim, P., and Denzler, J. (2012b). Beyond classification - large-scale gaussian process inference and uncertainty prediction. In *Big Data Meets Computer Vision: First International Workshop on Large Scale Visual Recognition and Retrieval (NIPS-WS)*.

LIST OF OWN PUBLICATIONS

- Freytag, A.**, Rodner, E., Bodesheim, P., and Denzler, J. (2012c). Rapid uncertainty computation with gaussian processes and histogram intersection kernels. In *Asian Conference on Computer Vision (ACCV)*, pages 511–524.
- Freytag, A.**, Rodner, E., Bodesheim, P., and Denzler, J. (2013). Labeling examples that matter: Relevance-based active learning with gaussian processes. In *German Conference on Pattern Recognition (GCPR)*, pages 282–291.
- Freytag, A.**, Rodner, E., Darrell, T., and Denzler, J. (2014a). Exemplar-specific patch features for fine-grained recognition. In *German Conference on Pattern Recognition (GCPR)*, pages 144–156.
- Freytag, A.**, Rodner, E., and Denzler, J. (2014b). Birds of a feather flock together - local learning of mid-level representations for fine-grained recognition. In *ECCV Workshop on Parts and Attributes (ECCV-WS)*.
- Freytag, A.**, Rodner, E., and Denzler, J. (2014c). Selecting influential examples: Active learning with expected model output changes. In *European Conference on Computer Vision (ECCV)*, pages 562–577.
- Freytag, A.**, Rodner, E., Simon, M., Loos, A., Köhl, H., and Denzler, J. (2016). Chimpanzee faces in the wild: Log-euclidean cnns for predicting identities and attributes of primates. In *German Conference on Pattern Recognition (GCPR)*, pages 51–63.
- Freytag, A.**, Rühle, J., Bodesheim, P., Rodner, E., and Denzler, J. (2014d). Seeing through bag-of-visual-word glasses: towards understanding quantization effects in feature extraction methods. In *International Conference on Pattern Recognition (ICPR) - FEAST workshop*.
- Freytag, A.**, Schadt, A., and Denzler, J. (2015). Interactive image retrieval for biodiversity research. In *German Conference on Pattern Recognition (GCPR)*, pages 129–141.
- Göring, C., **Freytag, A.**, Rodner, E., and Denzler, J. (2013). Fine-grained categorization - short summary of our entry for the imagenet challenge 2012. *arXiv:1310.4759*.
- Göring, C., Rodner, E., **Freytag, A.**, and Denzler, J. (2014). Nonparametric part transfer for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2489–2496.
- Käding, C., **Freytag, A.**, Rodner, E., Bodesheim, P., and Denzler, J. (2015). Active learning and discovery of object categories in the presence of unnameable in-

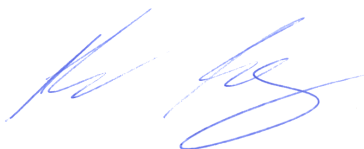
- stances. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4343–4352.
- Käding, C., **Freytag, A.**, Rodner, E., and Denzler, J. (2016). Watch, Ask, Learn, and Improve: A lifelong learning cycle for visual recognition. In *European Symposium on Artificial Neural Networks (ESANN)*.
- Käding, C., **Freytag, A.**, Rodner, E., Perino, A., and Denzler, J. (2016a). Large-scale active learning with approximated expected model output changes. In *German Conference on Pattern Recognition (GCPR)*, pages 179–191.
- Käding, C., Rodner, E., **Freytag, A.**, and Denzler, J. (2016b). Active and continuous exploration with deep neural networks and expected model output changes. In *NIPS Workshop on Continual Learning and Deep Networks (NIPS-WS)*.
- Käding, C., Rodner, E., **Freytag, A.**, and Denzler, J. (2016c). Fine-tuning deep neural networks in continuous learning scenarios. In *ACCV Workshop on Interpretation and Visualization of Deep Neural Nets (ACCV-WS)*.
- Lütz, A.** (2011). Robust classification and semi-supervised object localization with gaussian processes. In Mester, R. and Felsberg, M., editors, *Annual Symposium of the German Association for Pattern Recognition (DAGM)*, volume 6835 of *Lecture Notes in Computer Science*, pages 456–461. Springer, Heidelberg.
- Lütz, A.**, Rodner, E., and Denzler, J. (2011). Efficient multi-class incremental learning using gaussian processes. In *Open German-Russian Workshop on Pattern Recognition and Image Understanding (OGRW)*, pages 182–185.
- Lütz, A.**, Rodner, E., and Denzler, J. (2013). I want to know more - efficient multi-class incremental learning using gaussian processes. *Pattern Recognition and Image Analysis (PRIA)*, 23(3):402–407.
- Rodner, E., **Freytag, A.**, Bodesheim, P., and Denzler, J. (2012). Large-scale gaussian process classification with flexible adaptive histogram kernels. In *European Conference on Computer Vision (ECCV)*, pages 85–98.
- Rodner, E., **Freytag, A.**, Bodesheim, P., Fröhlich, B., and Denzler, J. (2016). Large-scale gaussian process inference with generalized histogram intersection kernels for visual recognition tasks. *International Journal of Computer Vision (IJCV)*, pages 1–28.
- Wojek, C., Ranipa, K., Rawat, A., Milde, T., and **Freytag, A.** (2017). Image quality assessment of fundus images using deep convolutional neural networks with extremely few parameters. In *ARVO Annual Meeting, Global connections in vision research*. (accepted for publication).

LIST OF OWN PUBLICATIONS

Ehrenwörtliche Erklärung

Hiermit erkläre ich ehrenwörtlich, dass mir die am heutigen Tage geltende Promotionsordnung der Fakultät für Mathematik und Informatik an der Friedrich-Schiller-Universität Jena bekannt ist. Ich habe die Dissertation selbstständig angefertigt, habe keine Textabschnitte oder Ergebnisse eines Dritten oder eigenen Prüfungsarbeiten ohne Kennzeichnung übernommen und habe alle von mir benutzten Hilfsmittel, persönliche Mitteilungen und Quellen in meiner Arbeit angegeben. Die Auswahl und Auswertung sämtlichen Materials geschah eigenständig. Insbesondere habe ich keine Hilfe eines Promotionsberaters in Anspruch genommen. Auch haben Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Des Weiteren erkläre ich, dass ich die vorliegende Arbeit noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung, sowie keine identische, eine in wesentlichen Teilen ähnliche oder eine andere Abhandlung bei einer anderen Hochschule als Dissertation eingereicht habe.



Jena, den 21. Januar 2016

Alexander Freytag