EFFICIENT MULTI-CLASS INCREMENTAL LEARNING USING GAUSSIAN PROCESSES

Alexander Lütz, Erik Rodner and Joachim Denzler

Computer Vision Group, Friedrich Schiller University Jena
{alexander.freytag, erik.rodner, joachim.denzler}@uni-jena.de
`http://www.inf-cv.uni-jena.de`

One of the main assumptions in machine learning is that sufficient training data is available in advance and batch learning can be applied. However, because of the dynamics in a lot of applications, this assumption will break down in almost all cases over time. Therefore, classifiers have to be able to adapt themselves when new training data from existing or new classes becomes available, training data is changed or should be even removed. In this paper, we present a method allowing efficient incremental learning of a Gaussian process classifier. Experimental results show the benefits in terms of needed computation times compared to building the classifier from the scratch.

Introduction

In the last decade, research in visual object recognition has focused mostly on batch learning, *i.e.* a classifier is build using a pre-defined and fixed set of training examples. However, if we think of the main goal of closing the gap between human and computer vision, which still exists when focusing on the recognition performance in real-world scenarios, batch learning is not appropriate. Even by using the large number of training examples already present in current image databases [1], a fixed learning system would not be able to adapt to new tasks and object classes. Due to this reason, it is important to incrementally learn a classifier in an efficient manner, *i.e.* by utilizing previously calculated model parameters. Such a procedure is often referred to as *online or incremental learning*.

Incremental learning has been studied for various types of classifiers. The work of Cauwenberghs and Poggio [2] presents how to carefully update the set of support vectors and their weights when incrementally adding new training data to existing SVM classifiers. Building on the algorithmic ideas of [2], Tax and Laskov [3] focus on incrementally learning a support vector data description classifier, which is a one-class classification method. For online learning on a class level, *i.e.* when examples of new classes become available, the methods proposed by Yeh *et al.* [4] can be used

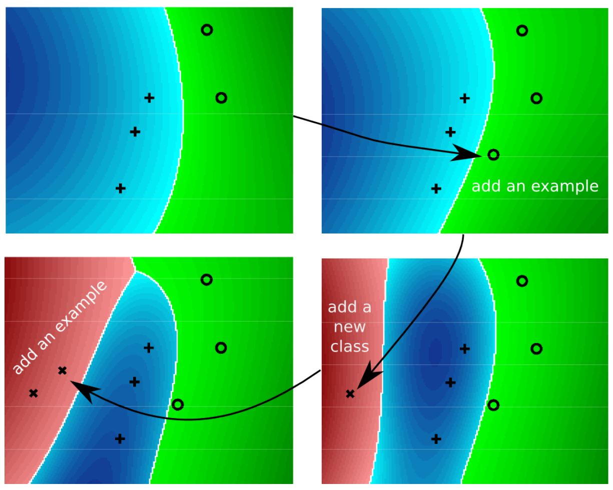

Fig. 1. Multi-class incremental learning: adding examples to already existing or new classes. Color brightness shows the classifier class score.

to perform online learning of one-vs-all SVMs. Csató and Opper [5] show how to apply the idea of Bayesian online learning to sparse Gaussian process (GP) models.

This paper focuses on exploring the advantages of a one-vs-all Gaussian process classifier in incremental learning scenarios. In contrast to previous work, we do not utilize a sparse GP representation or approximate inference techniques. We briefly review the GP framework and present the update formulas necessary to add new examples. The presented approach is applied to a 2d toy example in Fig. 1

Regression with Gaussian Processes

In the following, we briefly describe the basic principles of Gaussian process regression. We are given a set of input examples $\boldsymbol{X} = \big(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\big) \in \Omega^n$, where Ω is the space of images. For the regression case, let us further assume a vector $\boldsymbol{t}_L \in \mathbb{R}^n$ containing the corresponding function values is given and we are interested in estimating the relationship $f : \Omega \to \mathbb{R}$ between inputs and outputs. Because of the finite size of $\boldsymbol{X}$, we are not able to determine a single unique solution for f. Therefore, we model f to be drawn from a distribution over functions, *i.e.* a random process with index set $\mathbb{R}$. One possible choice for such an underlying random process is a Gaussian process $\mathcal{GP}(\mu, \kappa)$ [6]. The mean function $\mu : \Omega \to \mathbb{R}$ with

$$\mu(\boldsymbol{x}) = \mathbb{E}_f[\, f(\boldsymbol{x})\,]$$

is used to specify our prior assumptions about the mean value of the function at certain positions. The covariance function $\kappa : \Omega^2 \to \mathbb{R}$:

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_f \left[\big(f(\boldsymbol{x}) - \mu(\boldsymbol{x})\big)\big(f(\boldsymbol{x}') - \mu(\boldsymbol{x}')\big)\right]$$

models the expected covariance of the function values, which is directly related to the expected local variability of functions drawn from the process. Learning from output and input values and estimating the output t_* of a new example $\boldsymbol{x}_*$ can be done with Bayesian inference. In the following, we assume a zero mean function and that observed values $\boldsymbol{t}_L$ of f are corrupted with independent Gaussian noise with variance σ_n^2. In this special case, Bayesian inference leads to a closed-form solution for the posterior [6]:

$$p(t_* |\, \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{t}_L) = \mathcal{N}(t_* | \mu_*, \sigma_*^2) \ . \qquad (1)$$

This means that the posterior is Gaussian with the following mean and variance:

$$\mu_* = \boldsymbol{k}_*^T \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{t}_\mathrm{L} \qquad (2)$$

$$\sigma_*^2 = k_{**} - \boldsymbol{k}_*^T \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_* + \sigma_n^2 \ . \qquad (3)$$

With a slight abuse of notation, we use $k_{**} = \kappa(\boldsymbol{x}_*, \boldsymbol{x}_*)$, $\boldsymbol{k}_* = \kappa(\boldsymbol{X}, \boldsymbol{x}_*)$ and $\boldsymbol{K} = \kappa(\boldsymbol{X}, \boldsymbol{X})$ as abbreviations for the kernel values of the training set and the new example. It should be noted that the possibility to estimate the variance of the classifier output is an important benefit for advanced incremental learning, because it allows for active learning [7] and out-of-vocabulary detection [8].

GP Regression for Multi-class Classification

As we have seen in the previous section, the GP model and a Gaussian noise assumption leads to simple inference equations. If we consider non-Gaussian noise for classification problems, there are no closed-form solutions available and approximation methods are necessary [6]. However, it has been demonstrated in several papers [7, 8, 9] that applying GP regression directly to binary classification problems, without considering the discrete nature of the labels $t \in \{-1, 1\}$, gives comparable results. For multi-class classification problems with $t \in \{1, \ldots, M\}$, the one-vs-all technique can be applied, which leads to the following scores for each class $m \in \{1, \ldots, M\}$:

$$\mu_*^{(m)} = \boldsymbol{k}_*^T \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{t}_L^{(m)} \ , \qquad (4)$$

with $\boldsymbol{t}_L^{(m)}$ representing the vector of binary training labels for class m [7]. The final decision of the multi-class classifier is done by returning the class m with the highest score $\mu_*^{(m)}$.

Efficient Updates of the Kernel Matrix

As shown in equation (2) and (4), we only need $\boldsymbol{t}_\mathrm{L}, \boldsymbol{k}_*$ and $\left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1}$ for computing the predictive mean of a new input $\boldsymbol{x}_*$. Therefore, if training data is changed or new training data becomes available, we have to consider methods for updating the corresponding variables. In the following, we show how to update the inverse of the regularized kernel matrix $\left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)$ in an efficient manner. Another option is to update the Cholesky factorization, as described in [10, 11] and will be evaluated in a future work.

Let $\boldsymbol{K}$ be a $n \times n$-Matrix, $\boldsymbol{U}$ a $n \times p$-Matrix, $\boldsymbol{C}$ a $p \times p$-Matrix and $\boldsymbol{V}$ a $p \times n$-Matrix. Let further $\boldsymbol{K}$ and $\boldsymbol{C}$ be invertible. Woodbury's Formula [12] states that

$$\boldsymbol{K'}^{-1} = \left(\boldsymbol{K} + \boldsymbol{UCV}\right)^{-1} = \qquad (5)$$
$$\boldsymbol{K}^{-1} - \boldsymbol{K}^{-1}\boldsymbol{U}\Big(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{K}^{-1}\boldsymbol{U}\Big)^{-1}\boldsymbol{V}\boldsymbol{K}^{-1} \ .$$

If we already have precomputed $\boldsymbol{K}^{-1}$ and $\boldsymbol{K}$ is only changed by adding $\boldsymbol{UCV}$, we are able to compute $\boldsymbol{K'}^{-1}$ in a more efficient manner compared to standard approaches if $p \ll n$. Changes in only one specific training example $\boldsymbol{x}_i$ to $\boldsymbol{x}_i'$ can be modeled by

$$\boldsymbol{K'} = \left(\boldsymbol{K} + \boldsymbol{UV}\right) \qquad (6)$$

with $\boldsymbol{U} = \left[\boldsymbol{a}^{(i)}\ \boldsymbol{e}_i\right], \boldsymbol{V} = \left[\boldsymbol{e}_i^T\ \boldsymbol{a}^{(i)T}\right]^T$, $\boldsymbol{e}_i$ being the ith unity vector, and

$$\boldsymbol{a}_q^{(i)} = \begin{cases} \kappa\big(\boldsymbol{x}'_i, \boldsymbol{x}_q\big) - \boldsymbol{K}_{i,q} & \text{if } i \neq q \\ \frac{1}{2}\Big(\kappa\big(\boldsymbol{x}'_i, \boldsymbol{x}_q\big) - \boldsymbol{K}_{i,q}\Big) & \text{if } i = q \end{cases} \quad (7)$$

as the vector of differences between old and new kernel values for $\boldsymbol{x}_i$. Because of $\operatorname{rk}\big(\boldsymbol{UV}\big) = 2$ we call $\boldsymbol{K}'$ a rk-2-update of $\boldsymbol{K}$. If we now set $\boldsymbol{C}$ to the identity matrix, we can directly apply (5) for computing the new inverse of the kernel matrix efficiently.

If not one but a set $\mathcal{S} = \{\boldsymbol{x}_{s_1}, \ldots, \boldsymbol{x}_{s_k}\}$ of training examples changes, the update can be done either by iteratively changing one example or directly updating all examples simultaneously. For the latter case, the presented ideas for the rk-2-update can be generalized in the following way. Again we build the vector of differences between old and new kernel values

$$\boldsymbol{a}_q^{(i)} = \begin{cases} \kappa\big(\boldsymbol{x}'_i, \boldsymbol{x}_q\big) - \boldsymbol{K}_{i,q} & \text{if } \boldsymbol{x}_q \notin \mathcal{S} \\ \frac{1}{2}\Big(\kappa\big(\boldsymbol{x}'_i, \boldsymbol{x}_q\big) - \boldsymbol{K}_{i,q}\Big) & \text{if } \boldsymbol{x}_q \in \mathcal{S} \end{cases} \quad (8)$$

with $i \in \{s_1, \ldots, s_k\}$ and set $\boldsymbol{C}$ to the unity matrix. Furthermore, we set the two matrices $\boldsymbol{U} = \left[\boldsymbol{a}^{(s_1)} \ldots \boldsymbol{a}^{(s_k)}\ \boldsymbol{e}_{s_1} \ldots \boldsymbol{e}_{s_k}\right]$ and $\boldsymbol{V} = \left[\boldsymbol{e}_{s_1}^T \ldots \boldsymbol{e}_{s_k}^T \boldsymbol{a}^{(s_1)T} \ldots \boldsymbol{a}^{(s_k)T}\right]^T$ such that $\boldsymbol{UV}$ transforms $\boldsymbol{K}$ to $\boldsymbol{K}'$, which means that $\boldsymbol{K}'$ is a rk-$2k$-Update of $\boldsymbol{K}$. Thereby, we again can apply (5) for efficiently computing the modified inverse $\boldsymbol{K}'^{-1}$.

The previous explanations focused on efficiently updating $\boldsymbol{K}$ when existing examples are modified. If new training data becomes available we can directly make use of these ideas. For adding one example (rk-2-update), we increase the size of $\boldsymbol{K}$ by one, whereas for k new examples (rk-$2k$-update) we attach k new rows and columns to $\boldsymbol{K}$ with value one on the main diagonal and zero elsewhere. After preparing $\boldsymbol{K}$ in this way, we can apply the presented update approach and end up with an GP classifier incrementally trained in an efficient way.

The presented approach is independent of the number of classes taken into account. As presented in eq. (4) every one-vs-all classifier uses the same matrix $\boldsymbol{K}$ and differs merely in the corresponding binary label vector $\boldsymbol{t}_{\mathrm{L}}{}^{(m)}$. Therefore, only a single update of $\boldsymbol{K}'^{-1}$ is required. Even for new classes a corresponding classifier can be trained with minimal effort by simply computing the new binary label vector and using the efficiently updated $\boldsymbol{K}'^{-1}$.

Asymptotic Runtimes

Training a GP classifier with $n + k$ training examples and ignoring any further knowledge needs $\mathcal{O}\big((n+k)^3\big)$ time. This term is dominated by the time spent for computing the inverse of $\boldsymbol{K}$. Even with methods such as Cholesky decomposition, which takes advantage of the positive definiteness of $\boldsymbol{K}$, the main complexity stays cubically in the number of examples. Using the presented iterative IL approach the asymptotic bound decreases to $\mathcal{O}\big(k(n + k)^2\big)$. Finally, when adding k examples directly we can perform the computation in $\mathcal{O}\big(k^3 + (n + k)^2\big)$ time. Unfortunately, the involved constants are relatively high, which is caused by the fact that $\Big(\boldsymbol{C}^{-1}+\boldsymbol{V}\boldsymbol{K}^{-1}\boldsymbol{U}\Big)$ (eq. (5)) is not symmetric and can therefore not be inverted using Cholesky decomposition. If $n \in \mathcal{O}(k)$, the iterative update should be preferred, whereas for large datasets with only small changes ($k \ll n$) the direct approach should be the method of choice. In all cases the update of the corresponding $\boldsymbol{t}_{\mathrm{L}}$ can be done in constant time. If an example of a new class becomes available the new label vector has to be computed requiring $\mathcal{O}(n+k)$ and not affecting the complexity at all.

Experiments

Our update rules do not use any approximations, therefore, the differences between the results of batch and incremental learning are marginal and do not affect the recognition rates in our experiments. Due to this reason, we concentrate on *evaluating the runtime* only.

Data We follow the experimental setup of [4] and use the well-known Caltech-101 dataset [13]. For each step, we randomly sample n images and train a GP regression classifier based on these examples. Afterwards, we additionally sample k examples to add them incrementally or for batch training. Using suitable features and multi-class GP classification allows average recognition rates of up to 74% with 15 training examples on this dataset [7].

Features For realistic evaluations, we again follow the setup of [4] and choose feature vectors corresponding to the spatial pyramid match

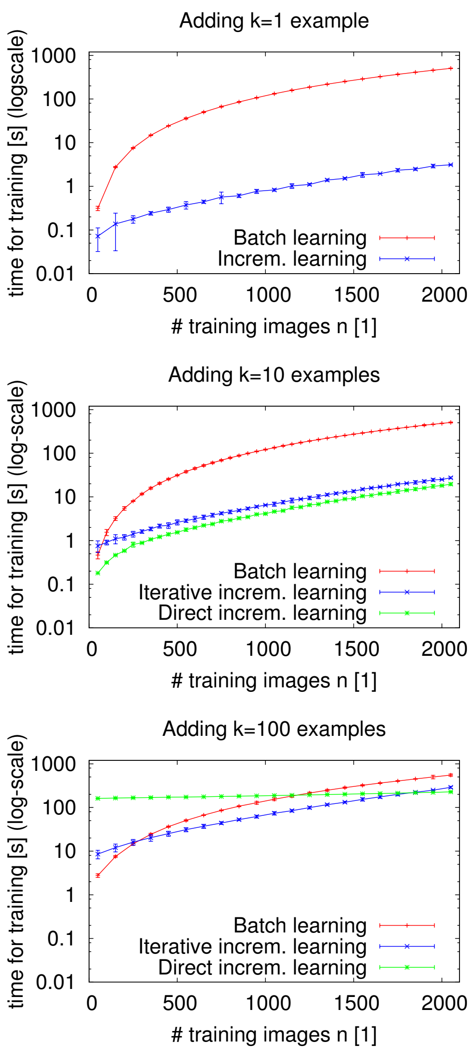

Fig. 2. Time measurements for incremental compared to batch learning with GP regression. Times are given in seconds and displayed in logarithmic scale.

kernel [14] as a representation for each image. Given computation times include the time needed for feature calculation.

Evaluation The experimental results confirm a significant benefit arising from incrementally training the classifier compared to batch training. As expected the resulting gain depends on the ratio of n and k. In our experiments, the speed-up was up to a factor of 100 (Fig. 2). Additionally, the theoretical result that a direct IL approach is most useful when k is large but $\frac{k}{n}$ is small, can be verified in Fig. 2.

Conclusions and Further Work

The aim of this paper was to show that Gaussian Processes, which are a powerful machine learning tool, can be applied to incremental learning scenarios using efficient retraining methods.We presented in-depth how to utilize the well-known Woodbury formula to incrementally update a GP-based classifier and performed experiments for the tasks of object recognition. Due to the lack of space, we did not consider efficient Cholesky updates [11] and numerical comparisons, which will be the topic of future work. Furthermore, the aim of this paper was to show the large number of advantages of the Gaussian Process framework for incremental learning.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248 – 255.

[2] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *NIPS*, 2001, vol. 13, pp. 409–415.

[3] D. Tax and P. Laskov, "Online svm learning: from classification to data description and back," in *Workshop on Neural Networks for Signal Processing (NNSP'03)*, 2003, pp. 499–508.

[4] T. Yeh and T. Darrell, "Dynamic visual category learning," in *CVPR*, 2008, pp. 1–8.

[5] L. Csató and M. Opper, "Sparse representation for gaussian process models," in *NIPS*, 2000, pp. 444–450.

[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 01 2006.

[7] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Gaussian processes for object categorization," *IJCV*, vol. 88, pp. 169–188, 2010.

[8] M. Kemmler, E. Rodner, and J. Denzler, "One-class classification with gaussian processes," in *ACCV*, 2010, pp. 489–500.

[9] E. Rodner, D. Hegazy, and J. Denzler, "Multiple kernel gaussian process classification for generic 3d object recognition from time-of-flight images," in *Proceedings of the IVCNZ'10*, 2010.

[10] D. Nguyen-Tuong, M. W. Seeger, and J. Peters, "Model learning with local gaussian process regression." *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[11] M. W. Seeger, "Low rank updates for the cholesky decomposition," University of California at Berkeley, Tech. Rep., 2004.

[12] W. W. Hager, "Updating the inverse of a matrix," *Society for Industrial and Applied Mathematics (SIAM) Review*, vol. 31, no. 2, pp. 221–239, 1989.

[13] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *CVPR '04: Workshop on Generative-Model Based Vision*, 2005.

[14] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006, pp. 2169–2178.