Approximations of Gaussian Process Uncertainties for Visual Recognition Problems

Paul Bodesheim[1], Alexander Freytag[1], Erik Rodner[1,2], and Joachim Denzler[1]

[1] Computer Vision Group, Friedrich Schiller University Jena, Germany
http://www.inf-cv.uni-jena.de
[2] UC Berkeley EECS, International Computer Science Institute, United States

Abstract. Gaussian processes offer the advantage of calculating the classification uncertainty in terms of predictive variance associated with the classification result. This is especially useful to select informative samples in active learning and to spot samples of previously unseen classes known as novelty detection. However, the Gaussian process framework suffers from high computational complexity leading to computation times too large for practical applications. Hence, we propose an approximation of the Gaussian process predictive variance leading to rigorous speedups. The complexity of both learning and testing the classification model regarding computational time and memory demand decreases by one order with respect to the number of training samples involved. The benefits of our approximations are verified in experimental evaluations for novelty detection and active learning of visual object categories on the datasets C-Pascal of Pascal VOC 2008, Caltech-256, and ImageNet.

1 Introduction

The Gaussian process framework is a powerful tool for solving regression and classification problems [15], especially for complex recognition tasks in the visual domain [9,10,16,7]. Using a Gaussian process classification model learned with training data, statistical inference can be done in a Bayesian manner providing the estimated class label of a test sample together with the uncertainty of this estimation. While the class label is obtained from the predictive mean, the classification uncertainty is related to the predictive variance of the Gaussian process. However, learning such a model turns out to be costly, since a matrix inversion is involved, whose calculation has a complexity cubic in the number of training samples. In the test step, computing the predictive mean is then possible in linear time, but the complexity of calculating the predictive variance is still quadratic. Although Gaussian processes achieve great performance in many computer vision tasks such as novelty detection [10,7], object categorization [9,16], and active learning [9,7], training and testing the model is expensive.

To overcome this drawback of high computation costs, we present an approximation of the variance leading to quadratic runtime during learning and only linear time in the test step. Besides storing all training samples, the memory demand of our approximation is only linear, since we do not need to keep the

J.-K. Kämäräinen and M. Koskela (Eds.): SCIA 2013, LNCS 7944, pp. 182–194, 2013.

whole kernel matrix in memory. Our proposed approximation is proved to be a true upper bound for the exact variance. In addition, we apply our basic idea of the approximation to the predictive mean allowing for faster learning of the whole Gaussian process model.

Recently, fast exact Gaussian process inference was proposed in [16] for large-scale recognition and extended in [7] for fast variance computations applied to novelty detection and active learning. However, the speed up is achieved by exploiting properties of the histogram intersection kernel. In contrast, our approximation goes one step further and is independent of the kernel function, which allows for using generalized rbf kernels [20] or any other application-specific kernel function. Experimental results show superior recognition performances of those kernel functions compared to histogram intersection kernels (see Sect. 7).

As applications of our approximation, we consider novelty detection and active learning. For novelty detection, the performance is evaluated in one-class classification scenarios [10,7], where only samples of a single class are known during learning. In the test step, samples of other classes serve as novelties, which need to be detected. The assumption is that novelties are dissimilar to already observed samples and are thus far away from the training data in the feature space. Furthermore, active learning experiments [9,7] for binary classification tasks are conducted. Starting with a small set of labeled samples, active learning methods automatically select samples from a pool of unlabeled data. These samples can then be labeled and incorporated in the learning step in order to improve the classification model. The goal is to select the most informative samples first, *e.g.*, samples close to the decision boundary to refine discrimination or samples far away from already labeled data to explore the feature space.

The paper is organized as follows. In Sect. 2, we review related work on various Gaussian process approximations as well as novelty detection and active learning using Gaussian process regression. The key idea of our approximation is presented in Sect. 3 together with an optimization problem to minimize the approximation error. How to solve this optimization problem is described in Sect. 4. Furthermore, to avoid costly optimizations, we propose a fast variance approximation in Sect. 5. In Sect. 6, we show that similar approximation techniques can also be used to allow for efficient predictive mean calculations. We present results of our experiments in Sect. 7. Conclusions can be found in the last section.

2 Related Work

In this section, we briefly review the Gaussian process regression framework and related work on approximating Gaussian processes as well as the application of Gaussian process regression for novelty detection and active learning.

2.1 Gaussian Process Regression

The key idea of Gaussian process regression is to model the dependency of continuous outputs $y(\mathbf{x}) = \mathrm{f}(\mathbf{x}) + \varepsilon$ on a latent function f and a noise term ε. Typically,

the noise term is assumed to be normally distributed with zero mean and noise variance σ_{n}^2, *i.e.*, $\varepsilon \sim \mathcal{N}(0, \sigma_{\mathsf{n}}^2)$. Latent functions f are supposed to be drawn from a Gaussian process prior with mean function $\mu(\cdot)$ and covariance function $\kappa(\cdot,\cdot)$. Without any prior knowledge, a zero mean assumption is a common choice. As a result of these requirements, output values y^* of unknown samples $\mathbf{x}^*$ can be predicted in a Bayesian manner by marginalizing over latent functions f. Given training samples $\mathbf{X}$ with labels $\mathbf{y}$, the predictive distribution of y^* is Gaussian: $y^* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}^* \sim \mathcal{N}(\mu_*, \sigma_*^2)$ and moments can be calculated in closed form:

$$\mu_* = \mathbf{k}_*^{\mathsf{T}} \left(\mathbf{K} + \sigma_{\mathsf{n}}^2 \mathbf{I}\right)^{-1} \mathbf{y} \quad \text{and} \tag{1}$$

$$\sigma_*^2 = k_{**} - \mathbf{k}_*^{\mathsf{T}} \left(\mathbf{K} + \sigma_{\mathsf{n}}^2 \mathbf{I}\right)^{-1} \mathbf{k}_* + \sigma_{\mathsf{n}}^2 \ , \tag{2}$$

where $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$, $\mathbf{k}_* = \kappa(\mathbf{X}, \mathbf{x}^*)$, $k_{**} = \kappa(\mathbf{x}^*, \mathbf{x}^*)$. For more details about Gaussian process regression and classification, we refer the reader to [15].

2.2 Approximations of Gaussian Processes

An overview of various sparse approximations for Gaussian process regression is provided by [13]. They propose a unifying scheme based on latent inducing variables that includes several previous methods, *e.g.*, subset of regressors (SoR), deterministic training conditional (DTC), fully independent training conditional (FITC), and partially independent training conditional (PITC) approximations. The simplest possible approximation of only using a subset of data (SoD) is also mentioned in [13] but does not fall inside their scheme. For comparisons with our approximations, we select the FITC approximation originally proposed by [18] as state-of-the-art and baseline technique. Furthermore, the work of [11] gives an extensive overview of approximating Gaussian processes in binary classification tasks with noise models different from the Gaussian noise model.

2.3 Novelty Detection with Gaussian Processes

Novelty detection with Gaussian processes in one-class classification scenarios can be done by label regression as shown in [10]. The authors propose using a zero mean assumption as well as training labels $\mathbf{y} = \mathbf{1} = (1, 1, \ldots, 1)^{\mathsf{T}}$ to compute predictions according to (1) and (2). They recommend using either the predictive mean μ_* (GP-Mean) or the predictive variance σ_*^2 (GP-Var) as novelty scores such that novelties are detected via a small mean value or a large variance. In their experiments, the simple Gaussian noise model mentioned in Sect. 2.1 leads to better performance compared to more complex models (sigmoid or probit) using Laplace approximation or expectation propagation [15].

2.4 Active Learning with Gaussian Processes

Active learning in binary classification tasks using Gaussian process regression was proposed in [9]. Informative samples are selected via a small absolute mean

value or a large variance similar to novelty detection but with binary labels $y_i \in \{-1, 1\}$. Since the predictive variance is independent of the label vector $\mathbf{y}$, we end up in a novelty detection scenario where informative samples for active learning are supposed to be far away from known samples in the feature space. The authors of [9] also use the simple Gaussian noise model stated in Sect. 2.1.

3 Diagonal Approximation of the Kernel Matrix

The advantage of the Gaussian process framework is the possibility of computing the classification uncertainty in terms of predictive variance, which can directly be used for novelty detection or active learning as reviewed in the previous section. However, computing the predictive variance is costly. With respect to the number of training samples, learning time is cubic since the inverse of the regularized kernel matrix is required and testing time is quadratic. Therefore, we are interested in an approximation of the predictive variance that only needs linear time for a new test sample and meets the following requirements:

1. the approximation is an upper bound for the exact variance: $\tilde{\sigma}_*^2 \geq \sigma_*^2$, to ensure that only samples similar to the training data obtain small uncertainties, and
2. the approximation error is minimal: $\tilde{\sigma}_*^2 - \sigma_*^2 \rightarrow \min$.

Recalling the calculation of Gaussian process predictive variance, we observe that the high computation cost arises from calculating the quadratic form $q = \mathbf{k}_*^\mathsf{T} \tilde{\mathbf{K}}^{-1} \mathbf{k}_*$ using $\tilde{\mathbf{K}} = \mathbf{K} + \sigma_\mathsf{n}^2 \mathbf{I}$ to denote the regularized kernel matrix. Therefore, our fast uncertainty approximation aims at approximating q in an efficient manner. Obviously, computing q would only take linear time if $\tilde{\mathbf{K}}^{-1}$ is a diagonal matrix[1]. Replacing $\tilde{\mathbf{K}}^{-1}$ by the inverse of a diagonal matrix $\mathbf{D}$:

$$\tilde{q} = \mathbf{k}_*^\mathsf{T} \mathbf{D}^{-1} \mathbf{k}_* \tag{3}$$

is an approximation of the quadratic form q and computable in linear time. We directly get an approximation of the Gaussian process predictive variance σ_*^2:

$$\tilde{\sigma}_*^2 = k_{**} - \mathbf{k}_*^\mathsf{T} \mathbf{D}^{-1} \mathbf{k}_* + \sigma_\mathsf{n}^2 \tag{4}$$

using $\tilde{q}$ instead of q. If $\mathbf{D}$ is given, this approximation is very efficient due to its calculation in linear time and with only linear memory demand.

To determine the elements of $\mathbf{D}^{-1}$, our approximation should meet the requirements 1 and 2 stated above. From linear algebra it is known that the approximation error $\tilde{\sigma}_*^2 - \sigma_*^2 = \mathbf{k}_*^\mathsf{T} (\tilde{\mathbf{K}}^{-1} - \mathbf{D}^{-1}) \mathbf{k}_*$ is bounded by:

$$\lambda_{\mathsf{min}}(\tilde{\mathbf{K}}^{-1} - \mathbf{D}^{-1}) \|\mathbf{k}_*\|^2 \leq \mathbf{k}_*^\mathsf{T} (\tilde{\mathbf{K}}^{-1} - \mathbf{D}^{-1}) \mathbf{k}_* \leq \lambda_{\mathsf{max}}(\tilde{\mathbf{K}}^{-1} - \mathbf{D}^{-1}) \|\mathbf{k}_*\|^2, \tag{5}$$

[1] Note that $\tilde{\mathbf{K}}^{-1}$ converges to a diagonal matrix for kernel scales $\sigma \rightarrow 0$ of the Gaussian kernel: $\kappa_{\text{Gaussian}}(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{\sigma^2} \|\mathbf{x} - \mathbf{x}'\|^2)$.

with $\lambda_{\mathsf{min}}(\cdot)$ and $\lambda_{\mathsf{max}}(\cdot)$ as the smallest and largest eigenvalue of the involved matrix. We can ensure requirements 1 and 2 by solving the optimization problem:

$$\min_{\mathbf{D}} \ \lambda_{\mathsf{max}}(\tilde{\mathbf{K}}^{-1} - \mathbf{D}^{-1}) \quad \text{s.t.} \quad \lambda_{\mathsf{min}}(\tilde{\mathbf{K}}^{-1} - \mathbf{D}^{-1}) \geq 0 \ . \tag{6}$$

Note that according to (5), the approximation error also depends on $\mathbf{k}_*$ and is in $\mathcal{O}(\|\mathbf{k}_*\|^2)$. Solving the optimization problem in (6) is discussed in the next section. A fast approximation that only satisfies the first requirement of being an upper bound is proposed and analyzed in Sect. 5.

4 Minimizing the Approximation Error

Optimizing the maximum eigenvalue function, which in our case is used to minimize the approximation error defined in the previous section, has been studied in the optimization community [12,4]. The goal is to minimize the maximum eigenvalue of a real symmetric $N \times N$ matrix, which is obtained by a real symmetric $N \times N$ matrix-valued affine function $\mathbf{A}(\mathbf{d})$ of $\mathbf{d} \in \mathbb{R}^N$ [12]:

$$\min_{\mathbf{d} \in \mathbb{R}^N} \lambda_{\mathsf{max}}(\mathbf{A}(\mathbf{d})) \ , \tag{7}$$

with $\mathbf{A}(\mathbf{d}) = \mathbf{A}^{(0)} + \sum_{k=1}^{N} d_k \mathbf{A}^{(k)}$ and $\mathbf{A}^{(0)}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$ being real symmetric matrices of size $N \times N$. The special case of $\mathbf{A}^{(k)} = \mathbf{e}_k \mathbf{e}_k^\mathsf{T}$ for $k = 1, \ldots, N$ with $\mathbf{e}_k$ being the k-th unit vector leads to:

$$\mathbf{A}(\mathbf{d}) = \mathbf{A}^{(0)} + \mathrm{Diag}(\mathbf{d}) \ , \tag{8}$$

where $\mathrm{Diag}(\mathbf{d})$ is a diagonal matrix with the vector elements of $\mathbf{d}$ placed on the main diagonal [12]. This formulation is exactly the one we need to minimize the upper bound for the approximation error in (6). We therefore set $\mathbf{A}^{(0)} = \tilde{\mathbf{K}}^{-1}$ and solve the problem in (7) using (8) to obtain $\mathbf{d}$. For the optimization, we use the smooth convex approximation algorithm of [4]. Note that we additionally have to ensure the constraint in (6), which we do by checking the validity of the smooth sub-problem solution [4]. Finally, we get the solution of the optimization problem in (6) via $\mathbf{D}^{-1} = -\mathrm{Diag}(\mathbf{d})$. We refer to the approximate variance based on the optimized matrix $\mathbf{D}^{-1}$ as *optimized approximate variance*.

5 Fast Approximation of Uncertainties

The key idea discussed in Sect. 3 is speeding up Gaussian processes by using a diagonal matrix instead of the full kernel matrix to allow for efficient matrix inversion. However, our optimization strategy presented in the previous section needs time cubic in the number of training samples during learning with quadratic memory demand, since the inverse of the regularized kernel matrix is required explicitly. We therefore additionally propose a fast approximation

of Gaussian process uncertainties by computing the elements of the diagonal matrix using elements of the full kernel matrix directly.

As in Sect. 3, we aim for approximating the quadratic form q in an efficient manner. First, we rewrite q as follows:

$$q = \mathbf{k}_*^\mathsf{T} \tilde{\mathbf{K}}^{-1} \mathbf{k}_* = \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{k}_*)_i (\mathbf{k}_*)_j (\tilde{\mathbf{K}}^{-1})_{ij} = \sum_{j=1}^{N} (\mathbf{k}_*)_j \mathbf{k}_*^\mathsf{T} (\tilde{\mathbf{K}}^{-1})_{\cdot j} \ , \tag{9}$$

where we denote the number of training samples with N, the j-th column of $\tilde{\mathbf{K}}^{-1}$ with $(\tilde{\mathbf{K}}^{-1})_{\cdot j}$, and vector as well as matrix elements using brackets with subscripts. Since $\mathbf{1}^\mathsf{T}(\tilde{\mathbf{K}})_{\cdot j} = \sum_{i=1}^{N} (\tilde{\mathbf{K}})_{ij}$ is the sum of all elements in the j-th column of $\tilde{\mathbf{K}}$, we use it to expand (9):

$$q = \sum_{j=1}^{N} \frac{(\mathbf{k}_*)_j \mathbf{k}_*^\mathsf{T} (\tilde{\mathbf{K}}^{-1})_{\cdot j} \mathbf{1}^\mathsf{T} (\tilde{\mathbf{K}})_{\cdot j}}{\mathbf{1}^\mathsf{T} (\tilde{\mathbf{K}})_{\cdot j}} = \sum_{j=1}^{N} \frac{(\mathbf{k}_*)_j \mathbf{k}_*^\mathsf{T} \tilde{\mathbf{Q}} (\tilde{\mathbf{K}})_{\cdot j}}{\mathbf{1}^\mathsf{T} (\tilde{\mathbf{K}})_{\cdot j}} \ , \tag{10}$$

where $\tilde{\mathbf{Q}} = [(\tilde{\mathbf{K}}^{-1})_{\cdot j} \ , \ \ldots \ , \ (\tilde{\mathbf{K}}^{-1})_{\cdot j}]$ is a matrix consisting of N times the j-th column of $\tilde{\mathbf{K}}^{-1}$. Note that there is no approximation involved so far. To avoid possible divisions by zero, we have to restrict our approximation to nonnegative kernel functions ($\kappa(\mathbf{x}, \mathbf{x}') \geq 0 \ \forall \mathbf{x}, \mathbf{x}'$) such as the family of squared exponential kernels [15] and positive noise variances $\sigma_\mathsf{n}^2 > 0$.

If we now replace $\tilde{\mathbf{Q}} = [(\tilde{\mathbf{K}}^{-1})_{\cdot j} \ , \ \ldots \ , \ (\tilde{\mathbf{K}}^{-1})_{\cdot j}]$ by the whole inverse of $\tilde{\mathbf{K}}$, we obtain an approximation $\tilde{q}$ of the quadratic form q leading to:

$$\tilde{q} = \sum_{j=1}^{N} \frac{(\mathbf{k}_*)_j \mathbf{k}_*^\mathsf{T} \tilde{\mathbf{K}}^{-1} (\tilde{\mathbf{K}})_{\cdot j}}{\mathbf{1}^\mathsf{T} (\tilde{\mathbf{K}})_{\cdot j}} = \sum_{j=1}^{N} \frac{(\mathbf{k}_*)_j \mathbf{k}_*^\mathsf{T} \mathbf{e}_j}{\mathbf{1}^\mathsf{T} (\tilde{\mathbf{K}})_{\cdot j}} = \sum_{j=1}^{N} \frac{(\mathbf{k}_*)_j^2}{\mathbf{1}^\mathsf{T} (\tilde{\mathbf{K}})_{\cdot j}} \ , \tag{11}$$

where $\mathbf{e}_j$ is the j-th unit vector. Defining a diagonal matrix $\tilde{\mathbf{D}}$ by $(\tilde{\mathbf{D}})_{jj} = \mathbf{1}^\mathsf{T} (\tilde{\mathbf{K}})_{\cdot j} = \sum_{i=1}^{N} (\tilde{\mathbf{K}})_{ij}$, the approximation $\tilde{q}$ can be written as:

$$\tilde{q} = \sum_{j=1}^{N} \frac{(\mathbf{k}_*)_j^2}{(\tilde{\mathbf{D}})_{jj}} = \mathbf{k}_*^\mathsf{T} \tilde{\mathbf{D}}^{-1} \mathbf{k}_* \ . \tag{12}$$

Using this approximation $\tilde{q}$ of the quadratic form q, we obtain our fast approximation of the predictive variance as in (4) via $\tilde{\sigma}_*^2 = k_{**} - \mathbf{k}_*^\mathsf{T} \tilde{\mathbf{D}}^{-1} \mathbf{k}_* + \sigma_\mathsf{n}^2$. In the rest of the paper, we refer to this approximation as *fast approximate variance*. From (12), we observe that our fast approximation of q and thus our fast variance approximation is similar to a weighted squared Parzen density estimation. While the original Parzen density estimation [2] averages the similarities between the test sample and all available training samples, our fast approximate variance involves the squared similarity between a test sample and each training sample weighted by the overall similarity to other training samples. In contrast to sparse approximations of Gaussian processes (see Sect. 2.2), each of the N training samples directly contributes to the prediction in our fast approximation.

Table 1. A comparison of our optimized and fast variance approximation with the exact calculation regarding asymptotic runtime and memory demand during learning and testing based on the number of training samples N

		Learning		Testing	
		Time	Memory	Time	Memory
exact variance	(Sect. 2)	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$
opt. variance approximation	(Sect. 4)	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
fast variance approximation	(Sect. 5)	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$

The learning step of our fast approximation takes $\mathcal{O}(N^2)$ time, basically to calculate pairwise similarities of training samples followed by computing elements of the diagonal matrix $\tilde{\mathbf{D}}$. These computations can easily be parallelized due to the independent entries of $\tilde{\mathbf{D}}$. For a test sample we need $\mathcal{O}(N)$ time to compute the fast approximate variance. Additionally, the memory demand is only linear in the number of training samples during both learning[2] and testing. Usually, the memory demand is quadratic, since at least once the whole kernel matrix has to be kept in memory. A comparison of the exact and our approximate variance calculations regarding asymptotic runtime and memory demand is given in Table 1.

It can be shown that our fast approximation is a true upper bound of the exact variance. The proof is given in our technical report [3]. Theorem 1 of [3] shows that the approximation error depends on $\|\mathbf{k}_*\|^2$, which can be seen as a modified Parzen estimate with quadratic kernel terms. For samples similar to the training data we get a high approximation error whereas for outliers the error is small. Since we are interested in novelty detection and active learning, we only care about the resulting ranking of test samples and we will see in Sect. 7 that the approximation does not affect the recognition performance at all.

An example showing the exact and our fast approximate variance can be seen in Fig. 1. Utilizing the ImageNet database [5], a Gaussian process one-class classifier is learned with samples of a single target class and tested with $50,000$ samples of $1,000$ classes including the target class. From the plots in the figure, we observe that the approximate variance has a similar shape as the exact variance (right part of the plots). Note that in novelty detection, we want to differentiate between samples from the target class with a low predictive variance and samples from unknown classes with a high predictive variance. The decrease of the variance in the right part of the plots is mainly caused by samples of the target class and can be observed for the same samples considering the exact and the approximate variance. Therefore, the difference between the target class and other classes can also be recognized using our fast variance approximation but much faster than computing the exact variance.

[2] Elements of $\tilde{\mathbf{D}}$ can be calculated one after another without computing $\tilde{\mathbf{K}}$ beforehand.

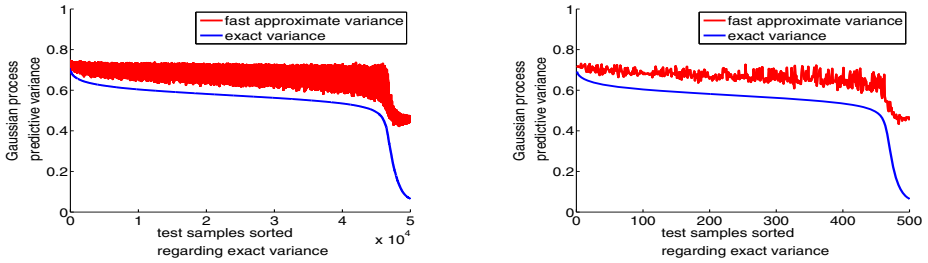

Fig. 1. The exact predictive variances of 50,000 test samples obtained from a novelty detection experiment are sorted in descending order and the corresponding fast approximations are shown. The right plot displays each 100-th sample of the left one.

6 Approximating the Predictive Mean

Since our proposed variance approximations lead to substituting the regularized kernel matrix in (2) by a diagonal matrix, we can do the same substitution in the calculation of the Gaussian process predictive mean. This leads to an approximate predictive mean $\tilde{\mu}_*$ defined as:

$$\tilde{\mu}_* = \mathbf{k}_*^\mathsf{T} \tilde{\mathbf{D}}^{-1} \mathbf{1} \ , \tag{13}$$

which turns out to be similar to a weighted Parzen density estimation with the same weights as in our proposed variance approximation (Sect. 5). Indeed, this does not give a computation speedup in the testing step, because the asymptotic runtime remains $\mathcal{O}(N)$ and thus linear in the number of training samples. However, the complexity for learning the model can be reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$, since computing the inverse of the regularized kernel matrix is replaced by calculating the elements of the diagonal matrix $\tilde{\mathbf{D}}^{-1}$. In our experiments, we tested both the optimized and the fast approximation of the predictive mean leading to results illustrated in the next section.

7 Evaluations of Experiments

We compare our fast approximations (GP-FA-Var and GP-FA-Mean) as well as our optimized approximations (GP-OA-Var and GP-OA-Mean) with the exact calculations (GP-Var and GP-Mean) and the FITC approximation of [18] (GP-FITC-Var and GP-FITC-Mean) in novelty detection and active learning[3]. Results for a standard regression task can be found in our technical report [3]. We used the GPML toolbox [14] as well as the provided code of [10,19] and evaluated each method within the same MATLAB framework. Re-implementations in C++ are used to measure computation times.

[3] MATLAB source code for our approximations is available at:
`http://www.inf-cv.uni-jena.de/Forschung/paperProjects/Approximations+of+Gaussian+Process+Regression.html`

Table 2. Computation times and median AUC scores from novelty detection experiments. **Bold**: best method per dataset, blue: best approximations per dataset, red: our approximations are faster than exact computations. *Nonoptimized SVDD code.

Method	Median AUC for Different Datasets C-Pascal	Caltech-256	ImageNet	Median Time Learning (100 samples)	Median Time Testing (average of 50,000 samples)
GP-FA-Var	75.86 %	77.46 %	76.48 %	0.09 ms	1.08 μs
GP-OA-Var	76.26 %	74.91 %	74.80 %	587.76 ms	1.08 μs
GP-Var [10]	**76.37**%	**77.92**%	75.51 %	0.29 ms	10.14 μs
GP-FITC-Var [18]	74.87 %	77.17 %	75.88 %	0.30 ms	0.54 μs
GP-FA-Mean	75.64 %	77.19 %	76.44 %	0.09 ms	0.04 μs
GP-OA-Mean	76.09 %	74.65 %	74.76 %	590.49 ms	0.04 μs
GP-Mean [10]	73.78 %	77.48 %	76.92 %	0.30 ms	0.04 μs
GP-FITC-Mean [18]	74.26 %	76.76 %	76.51 %	0.25 ms	0.03 μs
SVDD [19]/1SVM [17]	74.17 %	77.43 %	**76.99**%	11.50 ms*	1.16 μs*

7.1 Experimental Setup

Our evaluations are based on experiments on three different datasets: Cropped-Pascal (C-Pascal) [6], Caltech-256 [8], and ImageNet [5]. We choose the same 1,000 object categories from the ImageNet database [5] as done for ILSVRC 2010[4]. As image representations, we use the provided quantized local features[5] (densely sampled SIFT descriptors) to calculate histograms following the bag-of-visual-words (BoV) approach. Such features are also available for the Caltech-256 dataset[6] and used in our experiments. In contrast, provided histograms of HOG descriptors[7] are used to represent objects from C-Pascal images.

Similarities between histograms are computed using the histogram intersection kernel [1]: $\kappa_{\text{HIK}}(\mathbf{x},\mathbf{x}') = \sum_{d=1}^{D} \min(\mathbf{x}_d, \mathbf{x}'_d)$ or the corresponding generalized rbf kernel [20]: $\kappa_{\text{EXPHIK}}(\mathbf{x},\mathbf{x}') = \exp(2 \cdot \kappa_{\text{HIK}}(\mathbf{x},\mathbf{x}') - \kappa_{\text{HIK}}(\mathbf{x},\mathbf{x}) - \kappa_{\text{HIK}}(\mathbf{x}',\mathbf{x}'))$. We only present results obtained using the EXPHIK, since the HIK achieves inferior performance in all our experiments. Note that due to this reason, we outperform [7] in terms of recognition performance, since their approach is restricted to the HIK. For the Gaussian noise model, we use $\sigma_{\mathsf{n}}^2 = 0.1$. We randomly picked 25% of the training samples for the inducing subset of the FITC method in order to avoid costly optimizations.

Novelty Detection Setup. Each category of the Caltech-256 dataset is tested, where we randomly select 50% of the samples for training. During testing, the remaining samples serve as positives and all samples of the other 255 categories are considered as novelties. The same setup is used for the C-Pascal dataset, where we randomly select 25% of the samples for training. For both datasets, results are averaged over 20 random splits for each category.

The ImageNet dataset is split into two subsets for training and validation. The training set consists of 100 images per category resulting in a total of 100,000

[4] http://www.image-net.org/challenges/LSVRC/2010
[5] http://www.image-net.org/download-features
[6] http://homes.esat.kuleuven.be/~tuytelaa/unsup_features.html
[7] http://www.d2.mpi-inf.mpg.de/content/ralf

(a) Training images of the category *trolleybus*.

(b) Top five ranked test images using the exact variance (98.56 % AUC).

(c) Top five ranked test images using our fast variance approximation (98.47 % AUC).

Fig. 2. Image rankings from a novelty detection experiment on the ImageNet dataset. Red boxes indicate novelties, green boxes highlight images of the target class.

images and the validation set used for testing consists of 50 images per category resulting in a total of 50,000 images. In every task, we use 100 images of a single category during learning. Runtimes are obtained from our ImageNet experiments on a computer with 3.4 GHz CPU excluding the calculation of similarities with the kernel function, since computing the kernel values of training samples as well as similarities for test samples is shared among all methods. Note that due to this reason, we do not fully exploit the sparsity of the SVDD model.

Active Learning Setup. Active learning experiments are done using the ImageNet dataset. Binary classification tasks are built by randomly sampling ten categories, where one delivers positive samples and samples of the remaining nine serve as negatives. We start with five randomly picked samples per category, which results in five positives and 45 negatives. Performance is measured using AUC scores averaged over ten random initializations and 50 binary tasks.

7.2 Experimental Results

Novelty Detection. Besides the Gaussian process techniques, we also tested the support vector data description (SVDD) of [19] using an outlier ratio of $\nu = 0.1$. For SVDD, we used a nonoptimized implementation, which makes use of standard quadratic programming algorithms. The asymptotic runtimes of SVDD are similar to those of the standard Gaussian process methods without approximation. One-class SVM (1SVM) achieves recognition results equivalent to SVDD when using kernels that lead to constant self-similarities $\kappa(\mathbf{x}, \mathbf{x})$ [17]. Since this is the case in our experiments, we implicitly compare to both methods.

Computation times needed for both learning and testing the models as well as AUC scores averaged over all categories of three different datasets are summarized in Table 2. First of all, we note that our fast approximations achieve a clear speedup during learning and the time necessary to compute the variance in the test step reduces by a factor of 10. Note that the FITC approximation of the variance is faster in the test step due to its sparsity properties. During learning, the FITC variance approximation is slightly slower compared to the exact model caused by the number of samples used. Although appealing from a theoretical

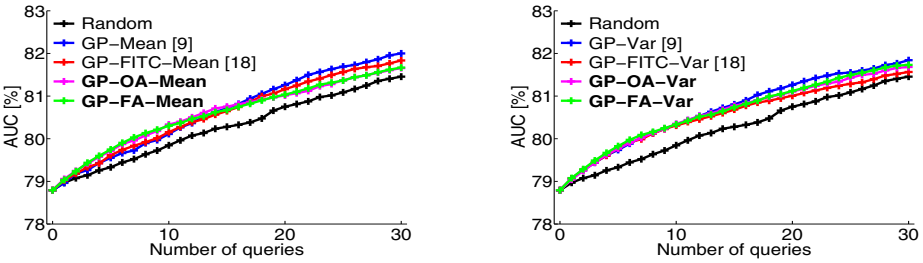

Fig. 3. Active learning results on the ImageNet dataset comparing predictive mean approximations (*left*) and predictive variance approximations (*right*).

point of view, our optimized approximations take a lot of time during learning. However, the optimized approximations are the best approximations regarding median AUC on the C-Pascal dataset.

The median AUC scores using our fast approximations are superior to those of the FITC method on all three datasets. Furthermore, our approximations achieve recognition results comparable to the exact calculations and the support vector based methods. The benefit of our fast approximate variance can be observed in terms of lower computation time during both learning and testing while keeping recognition performances. Our fast mean approximation achieves results as good as our fast variance approximation.

The obtained novelty scores can be used to create a ranking of the images showing which images fit best to the category. We present the top five ranked images obtained from a novelty detection experiment using the exact variance and our fast variance approximation in Fig. 2. Four out of these five test images are the same for both methods. Additionally, we observe that they rank the same sample at the top position, which does not belong to the target class *trolleybus* but to the visual similar class *police van*. This confusion is caused by similar appearance compared to training samples of the target class.

Active Learning. The results of our active learning experiments are shown in Fig. 3. For both mean and variance, we observe that our fast and our optimized approximations achieve performances superior to random sampling and comparable to the exact calculations as well as the FITC approximation. Therefore, our approximations are also suitable for active learning, since informative samples can be selected much faster from a large pool of unlabeled data. Note that the combinations of mean and variance used in [9,7] result in similar behavior in our experiments compared to solely using mean or variance.

8 Conclusions and Future Work

In this paper, we have proposed approximations of Gaussian process regression reducing both runtimes and memory demands during learning and testing. These approximations allow for using Gaussian process techniques in practice, where

runtime and memory is limited. Our key idea is to perform computations with a diagonal approximation of the kernel matrix, which allows for efficient model evaluations during testing. We presented two strategies to find the diagonal approximation: (1) an optimization approach that minimizes the approximation error and (2) a fast approach using squared similarities in a weighted Parzen-like estimation. Moreover, we have proved in [3] that the fast approximate variance is a true upper bound of the exact variance. We compared our approximations with the exact calculations as well as the FITC approximations in novelty detection and active learning demonstrating their suitability for various computer vision applications. Despite its simplicity, the fast approximation turned out to be competitive with the exact calculations and the FITC approximations while requiring less memory and computational time.

Future work will concentrate on similar approximations for Gaussian processes with noise models more complex than the Gaussian noise model. Additionally, we plan to extend our proposed approximations to multi-sample active learning, where a set of test samples has to be treated jointly instead of independently from each other.

References

1. Barla, A., Odone, F., Verri, A.: Histogram intersection kernel for image classification. In: ICIP, pp. 513–516 (2003)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
3. Bodesheim, P., Rodner, E., Freytag, A., Denzler, J.: An efficient approximation for gaussian process regression. Tech. Rep. TR-FSU-INF-CV-2013-01, Computer Vision Group, Friedrich Schiller University Jena, Germany (2013), `http://www.inf-cv.uni-jena.de/dbvmedia/TR_FSU_INF_CV_2013_01.pdf`
4. Chen, X., Qi, H., Qi, L., Teo, K.L.: Smooth convex approximation to the maximum eigenvalue function. Journal of Global Optimization 30(2), 253–270 (2004)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR, pp. 248–255 (2009)
6. Ebert, S., Larlus, D., Schiele, B.: Extracting structures in image collections for object recognition. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 720–733. Springer, Heidelberg (2010)
7. Freytag, A., Rodner, E., Bodesheim, P., Denzler, J.: Rapid uncertainty computation with gaussian processes and histogram intersection kernels. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) ACCV 2012, Part II. LNCS, vol. 7725, pp. 511–524. Springer, Heidelberg (2013)
8. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Tech. Rep. 7694, California Institute of Technology (2007), `http://authors.library.caltech.edu/7694`
9. Kapoor, A., Grauman, K., Urtasun, R., Darrell, T.: Gaussian processes for object categorization. IJCV 88(2), 169–188 (2010)
10. Kemmler, M., Rodner, E., Denzler, J.: One-class classification with gaussian processes. In: Kimmel, R., Klette, R., Sugimoto, A. (eds.) ACCV 2010, Part II. LNCS, vol. 6493, pp. 489–500. Springer, Heidelberg (2011)
11. Nickisch, H., Rasmussen, C.E.: Approximations for binary gaussian process classfication. JMLR 9, 2035–2078 (2008)

12. Overton, M.L.: On minimizing the maximum eigenvalue of a symmetric matrix. SIAM Journal on Matrix Analysis and Applications 9(2), 256–268 (1988)
13. Quinonero-Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate gaussian process regression. JMLR 6, 1939–1959 (2005)
14. Rasmussen, C.E., Nickisch, H.: Gpml gaussian processes for machine learning toolbox (2010), http://mloss.org/software/view/263/
15. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
16. Rodner, E., Freytag, A., Bodesheim, P., Denzler, J.: Large-scale gaussian process classification with flexible adaptive histogram kernels. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part IV. LNCS, vol. 7575, pp. 85–98. Springer, Heidelberg (2012)
17. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation 13(7), 1443–1471 (2001)
18. Snelson, E., Ghahramani, Z.: Sparse gaussian processes using pseudo-inputs. In: NIPS, pp. 1257–1264 (2005)
19. Tax, D.M.J., Duin, R.P.W.: Support vector data description. Machine Learning 54(1), 45–66 (2004)
20. Vempati, S., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Generalized rbf feature maps for efficient detection. In: BMVC, pp. 2.1–2.11 (2010)