E. Rodner, A. Freytag, P. Bodesheim, and J. Denzler: Large-Scale Gaussian Process Classification with Flexible Adaptive Histogram Kernels. ECCV 2012, pp. 85–98.

http://link.springer.com/chapter/10.1007/978-3-642-33765-9_7

Large-Scale Gaussian Process Classification with Flexible Adaptive Histogram Kernels

Erik Rodner, Alexander Freytag, Paul Bodesheim, and Joachim Denzler

Computer Vision Group, Friedrich Schiller University Jena, Germany
{firstname.lastname}@uni-jena.de
http://www.inf-cv.uni-jena.de

Abstract. We present how to perform exact large-scale multi-class Gaussian process classification with parameterized histogram intersection kernels. In contrast to previous approaches, we use a full Bayesian model without any sparse approximation techniques, which allows for learning in sub-quadratic and classification in constant time. To handle the additional model flexibility induced by parameterized kernels, our approach is able to optimize the parameters with large-scale training data. A key ingredient of this optimization is a new efficient upper bound of the negative Gaussian process log-likelihood. Experiments with image categorization tasks exhibit high performance gains with flexible kernels as well as learning within a few minutes and classification in microseconds for databases, where exact Gaussian process inference was not possible before.

Key words: Large-scale Gaussian Processes, Histogram Intersection Kernels, Hyperparameter Optimization, Bayesian Modeling

1 Introduction

Non-linear learning with histogram kernels is currently one of the main techniques for solving complex visual recognition tasks [1–3]. This is mainly because histogram kernels, such as the histogram intersection kernel (HIK), exploit the property that histograms are normalized and lie in a very specific subspace [4], which allows providing a more suitable measure of similarity compared to standard kernels. For learning, SVM classifiers are the most prominent technique. However, it has been shown that full Bayesian techniques, *e.g.*, Gaussian process (GP) methods, do offer two important advantages: (1) they allow hyperparameter optimization by maximizing the marginal likelihood of the model, and (2) the uncertainty of the estimate can be predicted. Their main disadvantage is the cubic runtime of the learning step, which prevents them from being used in large-scale scenarios. Nevertheless, due to the large number of available image data, current tasks and research is shifting more and more towards large-scale learning scenarios, where the final goal is to efficiently handle several thousands to millions of training examples [5].
We present how to perform multi-class GP classification and hyperparameter optimization with large-scale datasets without any sparse approximation. The

memory and runtime requirements of our methods are sub-quadratic allowing for scalability. The approach is based on fast multiplications of the histogram intersection kernel matrix with an arbitrary vector. This allows for solving the GP inference equations by utilizing iterative solvers. Furthermore, we demonstrate that hyperparameter optimization with the complete GP model can also be performed in an efficient manner by exploiting an upper bound of the determinant of the kernel matrix. The upper bound depends on terms, which can be efficiently calculated. The main contributions of this paper are as follows:

1. We show how to perform training and classification in a Bayesian manner with Gaussian processes and histogram intersection kernels in sub-quadratic and constant time, respectively.
2. Hyperparameter optimization for large-scale datasets with efficient GP marginal likelihood optimization is presented, which allows for linear kernel combination and feature relevance determination.
3. We demonstrate the advantages of parameterized histogram intersection kernels.

Additionally, Gaussian process classification with label regression [6] is extended towards handling imbalanced learning data. The remainder of our paper is organized as follows. In Sect. 2, we give a short overview of related work on efficient GP classification and exploiting the efficiency of the histogram intersection kernel. Gaussian processes for classification and the key concepts of the efficiency of the histogram intersection kernel are reviewed in Sect. 3 and 4. In Sect. 5, we demonstrate how GP classifiers can be trained, optimized, and evaluated in a fast manner by making use of the HIK properties. Experimental results for medium as well as large-scale classification tasks are shown in Sect. 6 highlighting the suitability of our efficient computations for various scenarios. A summary of our findings and a discussion of future research directions conclude the paper.

2 Related Work

Fast Learning and Classification with HIK To overcome the drawback of time-consuming classification with kernel methods, Vedaldi and Zisserman [7] presented how to approximate the values of the histogram intersection kernel with explicit feature transformations. In contrast, Maji *et al.* [8] exploited the properties of HIK directly for calculating SVM decision scores in $\mathcal{O}(D \log(m))$ time compared to $\mathcal{O}(Dm)$ for standard SVM inference with m being the number of support vectors and D being the number of feature dimensions. Going one step further, Wu [9] presented fast SVM training by using the HIK properties to reformulate the SVM dual problem. The current paper, which was inspired by both works, shows that the special properties of the HIK can also be exploited for GP classification and even for hyperparameter optimization.

Generalized HIK and Hyperparameter Optimization Barla *et al.* [10] applied the HIK for image classification and proved it to be a Mercer kernel

for images having the same size. Since that time, a lot of improvements on this kernel have been proposed, *e.g.*, HIK with polynomial transformations [1] or the weighted multi-level extension known as pyramid match kernel (PMK) [2]. We show how to further generalize the HIK with arbitrary feature transformations and weights for each dimension. Therefore, our work is similar to [4], where a cross-validation procedure is proposed to estimate multiple weights of histogram kernels. In contrast, our hyperparameter optimization is based on a Bayesian model and can be utilized for large-scale scenarios, which is especially necessary when trying to estimate a large number of hyperparameters.

Fast GP Classification and Regression GP classifiers require a computation time and memory cubically and quadratically in the number of training examples. Therefore, their direct application to large-scale problems is limited. A growing number of publications deal with tackling this problem by introducing sparse approximations assuming conditional independence between sets of certain variables. These variables could be specified examples of the training set or can be learned during training [11]. Although these techniques lead to impressive results, the necessary independence assumptions neglect information provided in training and test data. The only work we are aware of tackling full large-scale GP inference is the greedy block technique of Bo and Sminchisescu [12], which does not require storing the full kernel matrix in memory. However, kernel values have to be calculated explicitly, which is not necessary in our case. In experiments, we show that their method can be improved by orders of magnitude in computation time by exploiting HIK properties.

3 GP regression and Hyperparameter Optimization

Let $\mathcal{X}$ be the space of all possible input data, *e.g.*, D-dimensional feature vectors. Given n training examples $\boldsymbol{x}^{(i)} \in \mathbf{X} \subset \mathcal{X}$ as well as corresponding binary labels $y_i \in \{-1, 1\}$, we would like to predict the label y_* of an unseen example $\boldsymbol{x}^* \in \mathcal{X}$. We now assume that f is a sample of a GP prior, *i.e.*, $f \sim \mathcal{GP}(0, K)$ with covariance function K, and that labels y_i are conditionally independent given $f(\boldsymbol{x}^{(i)})$. Furthermore, a simple additive Gaussian noise model with variance σ^2 is used:

$$p(y_i \mid f_i) = \mathcal{N}(y_i \mid f_i, \sigma^2) \ . \tag{1}$$

We follow [6] and solve a given binary classification problem as a regression problem, which regards y_i as real-valued function values instead of discrete labels. This is advantageous, because in this case the GP model assumptions lead to analytical solutions of the involved marginalizations and allow for directly predicting the expectation μ_* of the posterior of the label y_* given a new example $\boldsymbol{x}^*$ [13]:

$$\mu_* = \boldsymbol{k}_*^T (\mathbf{K} + \sigma^2 \cdot \mathbf{I})^{-1} \boldsymbol{y} = \boldsymbol{k}_*^T \boldsymbol{\alpha} \ . \tag{2}$$

The vector $\boldsymbol{k}_*$ contains the kernel values $(\boldsymbol{k}_*)_i = K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)$ corresponding to a test example $\boldsymbol{x}^*$, $\mathbf{K}$ is the kernel matrix of the training data, and $\boldsymbol{y}$ is the vector containing all training labels.

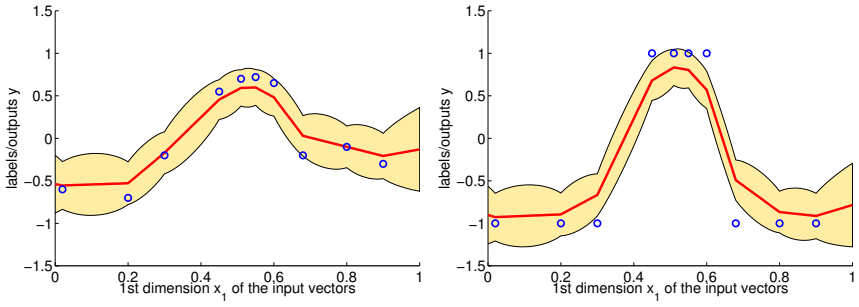


Fig. 1. Piecewise linearity of the regression function when using Gaussian process regression applied to the histogram intersection kernel: 2-dimensional input vectors $\boldsymbol{x}$ are used but due to the normalization $\|\boldsymbol{x}\|_1 = 1$, we only display the predictive mean (*red graph*) and confidence areas (*shaded area*) derived from the predictive variance with respect to the first dimension of the input vectors. Training points are shown as *blue dots* and the noise variance is set to 0.1

Hyperparameter Optimization In this paper, we use kernel functions that depend on hyperparameters $\boldsymbol{\eta}$, which have an important impact on the resulting classification model. In contrast to SVM techniques, the GP framework allows for finding their optimal values by likelihood maximization instead of expensive cross-validation. For GP regression, the negative log-likelihood is given by [13]

$$-\log p(\boldsymbol{y} \mid \mathbf{X}, \boldsymbol{\eta}) = \frac{1}{2}\boldsymbol{y}^T \left(\tilde{\mathbf{K}}_{\boldsymbol{\eta}}\right)^{-1} \boldsymbol{y} + \frac{1}{2}\log\det\left(\tilde{\mathbf{K}}_{\boldsymbol{\eta}}\right) + \frac{n}{2}\log 2\pi \qquad (3)$$

with $\tilde{\mathbf{K}}_{\boldsymbol{\eta}}$ being the parameterized kernel matrix having the noise variance σ^2 added to the main diagonal.

Multi-class Classification Multi-class classification can be done by utilizing the one-vs-all technique [6], which also offers to perform model selection by joint optimization of hyperparameters with all involved binary problems [6]. The objective function is simply the sum of all binary negative log-likelihoods.

Imbalanced Datasets If the number of positive and negative samples during training differs, the resulting decision function becomes biased towards the class more prominent in the training data. Especially for large-scale datasets with some hundred positive examples but several thousand negatives, this bias becomes crucial for the overall accuracy. To overcome this behavior, we propose using different noise levels for positive and negative examples, *i.e.*, the diagonal matrix $\mathbf{N}$ is added to the kernel matrix with $N_{ii} = 2\sigma^2 \cdot \left(\frac{|\{j \mid y_i = y_j\}|}{n}\right)$. By rewriting GP regression into a regularized least-squares problem [13, p. 144], this balancing strategy leads to an equal sum of positive and negative weights. Due to the lack of space, we refer to the supplementary material[1] for detailed derivations.

[1] Supplementary material: http://www.inf-cv.uni-jena.de/gp_hik.html

4 Efficient Kernel Calculations with Histogram Kernels

Kernel methods are one of the fundamental tools used to handle the complexity of visual recognition. It has been shown that the histogram intersection kernel

$$K^{\text{hik}}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \min(x_d, x'_d) \ , \tag{4}$$

which is often used to compare histogram feature vectors $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^D$, allows for efficient classification and learning with support vector machines (SVM) [8, 9]. In our approach, we use the HIK directly in the previously presented GP framework as a covariance function. Figure 1 shows two examples of GP regression and classification with this model. The interesting observation is that the regression function estimated by the predictive mean given in Eq. (2) is piecewise linear. We exploit this property for speeding up GP regression and hyperparameter optimization in Sect. 5.
In the following, we briefly review the techniques of [8, 9] for speeding up the computation of kernel terms and extend them towards using parameterized generalizations of the HIK.

Fast Kernel Calculation As we have seen in Eq. (2), similar to SVM and many other kernel methods, the predictive mean is a weighted sum of kernel values. The HIK allows for decomposing it in two parts [8]:

$$\boldsymbol{k}_*^T \boldsymbol{\alpha} = \sum_{i=1}^{n} \alpha_i \sum_{d=1}^{D} \min(x_d^{(i)}, x_d^*) = \sum_{d=1}^{D} \Big(\sum_{\{i: x_d^{(i)} < x_d^*\}} \alpha_i x_d^{(i)} + x_d^* \sum_{\{j: x_d^{(j)} \geq x_d^*\}} \alpha_j \Big) \ . \tag{5}$$

We can now significantly reduce the computational costs using the following trick. Let us assume that permutations π_d are given which rearrange the training examples such that they are sorted in an ascending order in each dimension d. Then, we can rewrite Eq. (5) as

$$\boldsymbol{k}_*^T \boldsymbol{\alpha} = \sum_{d=1}^{D} \Big(\underbrace{\sum_{i=1}^{r} \alpha_{\pi_d^{-1}(i)} x_k^{(\pi_d^{-1}(i))}}_{\doteq\, A(d,r)} + x_d^* \underbrace{\sum_{i=r+1}^{n} \alpha_{\pi_d^{-1}(i)}}_{\doteq\, B(d,r)} \Big) \ , \tag{6}$$

with r being the number of examples that are smaller than x_d^* in dimension d. Thus, Eq. (6) proves the piecewise linearity of the predictive mean of Gaussian process regression with HIK. If we precompute the two terms of the linear function during learning, evaluating the scores for test examples can be done with a few evaluations of A and B for each dimension. Given the vector $\boldsymbol{\alpha}$, the resulting computation time for building A and B is dominated by sorting in $\mathcal{O}(D\, n \log n)$ operations. In terms of memory usage, we only have to store $\mathcal{O}(D\, n)$ elements in contrast to the kernel matrix of size $\mathcal{O}(n^2)$. For calculating the score of a new

example, we need $\mathcal{O}(D \log n)$ operations to find the correct position r in each dimension and compute the linear function in Eq. (6) by evaluating A and B. Similar considerations hold for multiplications of an arbitrary vector $\boldsymbol{v} \in \mathbb{R}^n$ with the kernel matrix $\mathbf{K}$, which can be done in $\mathcal{O}(D \cdot n)$. Furthermore, we can exploit sparsity of feature vectors with a careful implementation.

Quantization of the Feature Space If we assume that feature values in dimension d are bounded by $x_d^* \in [l_d, u_d]$, the evaluation can be further speeded up by quantizing the feature space [8]. Using a quantization for each dimension with q bins, only q different outputs are possible for Eq. (6). With already computed matrices A and B, we can proceed with building a final lookup table T of dimension $D \times q$. Due to the already given permutations π_d, we can perform this within $\mathcal{O}\left(D \max\left(q, n\right)\right)$ operations. As a result, the time spent for evaluating the score of a new test example decreases to $\mathcal{O}(D)$. Consequently, for a given number of dimensions the score of a new test example can be computed in constant time independent of n.

Very General Histogram Intersection Kernels Boughorbel *et al.* [1] show that the HIK equipped with any positive valued function g:

$$K^{\text{ghik}}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \min\left(g\left(x_d\right), g\left(x'_d\right)\right) \quad , \tag{7}$$

still remains a positive-definite kernel. If g is an automorphism, the relative order of the training elements stays valid after evaluating g. Therefore, the proposed techniques can also be applied to these generalized variants of the HIK and we can even use the same quantization by storing the original feature values. Two common examples of such functions are the powered absolute value $g_{|\cdot|,\eta}(x) = |x|^\eta$ and the exponential $g_{e,\eta}(x) = \frac{\exp(\eta|x|)-1}{\exp(\eta)-1}$. In the remaining sections, we refer to them as generalized HIK (G-HIK) and exponential HIK (EXP-HIK). The kernel function given in Eq. (7) can be generalized even further by considering functions $g^{(d)}$ for each dimension. For example, $g^{(d)}(x_d) = \eta_d \cdot x_d$ with $\eta_d \geq 0$ allows for individually weighting input dimensions:

$$K^{\text{weights}}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \eta_d \cdot \min\left(x_d, x'_d\right) \quad . \tag{8}$$

In subsequent sections, we present how to optimize the parameters $\boldsymbol{\eta}$ even for large-scale training data. Together with the kernel function in Eq. (8), this allows for linear kernel combination [6] and automatic relevance determination [13].

5 Efficient GP Multi-class Classification

In this section, we demonstrate that GP regression and hyperparameter optimization can be performed efficiently when using histogram intersection kernels. An overview is shown in Fig. 2, whereas Table 1 summarizes the asymptotic computation times necessary for each step.

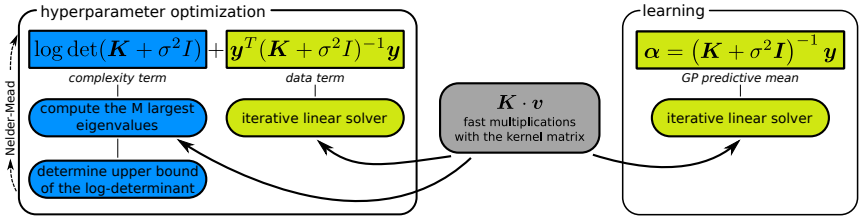


Fig. 2. Main outline of GP classification and hyperparameter optimization using fast multiplications with the kernel matrix

Table 1. Overview of asymptotic runtimes for training, testing, and optimization of hyperparameters for baseline GP compared to our approach. D denotes the number of dimensions, n the number of training examples, M the number of classes, and T_1 and T_2 the number of iterations used for the linear solver and the optimizer, respectively

	Asymptotic runtime	
Evaluation step	GP baseline	GP + HIK + Quantization
Training (Sect. 5.1)	$\mathcal{O}(\mathbf{n^3}+\mathbf{n^2}D)$	$\mathcal{O}(\mathbf{n}D(T_1M+\log\mathbf{n}))$
Hyperparameter opt. (Sect. 5.2)	$\mathcal{O}((\mathbf{n^3}+\mathbf{n^2}D)T_2)$	$\mathcal{O}(\mathbf{n}MDT_1T_2)$
Testing (Sect. 5.1)	$\mathcal{O}(\mathbf{n}MD)$	$\mathcal{O}(MD)$

5.1 Learning and Classification

Inference with a GP model requires two steps: (1) solving the linear equation system $\tilde{\mathbf{K}}_{\boldsymbol{\eta}}\cdot\boldsymbol{\alpha}=\boldsymbol{y}$ and (2) calculating the scalar product $\boldsymbol{k}_*^T\boldsymbol{\alpha}$. For large-scale datasets, storing the full kernel matrix is impossible and applying a Cholesky decomposition with a runtime of $\mathcal{O}(n^3)$ far from being practical. As we have seen in Sect. 4, multiplications with the kernel matrix can be done in linear time with histogram intersection kernels. Therefore, we use an iterative linear solver to tackle step 1. Wu [9] used a coordinate descent method to solve the quadratic program related to SVM learning. In contrast, our experiments show that a linear conjugate gradients (CG) method converges faster. The total asymptotic runtime for learning is $\mathcal{O}(nD(T_1M+\log n))$ including sorting. The total number of iterations T_1 of the CG method depends on the condition number of the kernel matrix and we also see that the runtime performance of our method is linear in the number of classes M. We stop the CG method when the maximum norm of the residual drops below 10^{-2}.
After estimation of the coefficients $\boldsymbol{\alpha}$, we use the quantization algorithm of [8] reviewed in Sect. 4 allowing for computing $\boldsymbol{k}_*^T\boldsymbol{\alpha}$ in constant time (step 2). In our experiments, we choose an equidistant quantization with $q=100$.

5.2 Large-Scale Hyperparameter Optimization

To optimize kernel hyperparameters with a large-scale dataset, we have to minimize the negative GP log-likelihood as given in Eq. (3). Due to the computational demand of evaluating Eq. (3) for large-scale datasets, we bound the negative log-likelihood with an efficiently computable function from above. Finding suitable

hyperparameters is then done by minimizing this upper bound instead of the real negative log-likelihood. Optimization is carried out with a method that does not require any gradient information, because calculating the gradient of the log-likelihood or the gradient of our upper bound is a costly operation.
Evaluating the log-likelihood requires the calculation of two different terms, the logarithm of the kernel matrix determinant and the data term involving the labels $\boldsymbol{y}$. The latter one is easy to compute, because it simply involves solving the same linear system as required for learning. However, the determinant of the kernel matrix is difficult to handle and we require some upper bound on it.

Efficient Upper Bound of the Log-Determinant Computing the determinant of a matrix is a costly algebraic operation, even with fast matrix multiplications [14]. Due to this reason, we use the upper bound provided by Bai and Golub [15], which turns out to be efficiently computable for histogram intersection kernel matrices. If the eigenvalues λ_i of $\mathbf{D}$ can be bounded by $0 < \lambda_i \leq \beta$, an upper bound of the log-determinant is given by:

$$\log\det(\mathbf{D}) \leq \begin{bmatrix}\log\beta & \log\bar{t}\end{bmatrix} \begin{bmatrix}\beta & \bar{t} \\ \beta^2 & \bar{t}^2\end{bmatrix}^{-1} \begin{bmatrix}\mu_1 \\ \mu_2\end{bmatrix} \doteq \mathrm{ub}(\beta, \mu_1, \mu_2) \tag{9}$$

where $\mu_1 = \mathrm{tr}\,(\mathbf{D})$, $\mu_2 = \|\mathbf{D}\|_F^2$, and $\bar{t} = \frac{\beta\mu_1 - \mu_2}{\beta n - \mu_1}$ [15]. It is interesting to note that this bound is tight for regularized rank-1 matrices $\mathbf{D} = \boldsymbol{u}\boldsymbol{u}^T + \tau\mathbf{I}$ [15]. For very complex classification tasks, we often observe a similar structure of the kernel matrix, which suggests that the bound is suitable in those scenarios.
To calculate the bound for the regularized kernel matrix $\tilde{\mathbf{K}}_{\boldsymbol{\eta}}$, we need the largest eigenvalue λ_1, the trace, and the squared Frobenius norm. We first compute the largest eigenvalue λ_1 with the Arnoldi iteration, which only requires matrix vector products. In our experiments, the algorithm needed approximately 10 steps to converge for various settings. Furthermore, it is easy to verify that the trace of the histogram intersection kernel matrix is the sum of all features values. The squared Frobenius norm is not directly available, but we can approximate it by $\tilde{\mu}_2 = \sum_{i=1}^{M} \lambda_i^2 \approx \sum_{i=1}^{n} \lambda_i^2 = \mu_2$ with M being the number of classes of the classification task and λ_i being the eigenvalues of the kernel matrix in decreasing order, *i.e.*, $\lambda_1 \geq \ldots \geq \lambda_n$. The motivation for this approximation is as follows: if we have M classes with very compact clusters and large distances between each other, the kernel matrix should obey a simple block structure of rank M leading to M non-zero eigenvalues. Due to the fact that our approximation of μ_2 is also a lower bound of $\|\mathbf{D}\|_F^2$, the necessary computations in Eq. (9) are still well-defined and it can be proved that we still have a proper upper bound of the log-determinant (see supplementary material for a detailed proof):

Theorem 1 (Upper bound with $\tilde{\mu}_2$). *For a given positive definite matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ with trace μ_1 and squared Frobenius norm μ_2 the following holds:*

$$\log\det(\mathbf{D}) \leq \mathrm{ub}(\beta, \mu_1, \mu_2) \leq \mathrm{ub}(\beta, \mu_1, \tilde{\mu}_2) \quad \textit{if} \quad \tilde{\mu}_2 \leq \mu_2 \ . \tag{10}$$

To summarize, we need to perform the following steps to efficiently bound the negative GP log-likelihood in each iteration of the hyperparameter optimization method:

1. Compute the data term by utilizing the CG method.
2. Compute the trace μ_1 as the sum of all feature values.
3. Calculate the first M eigenvalues with the Arnoldi iteration.
4. Approximate the Frobenius norm with the sum of squared eigenvalues.
5. Compute the bound given in Eq. (9) with the approximated Frobenius norm.

In our experiments, the resulting upper bound of the negative GP log-likelihood was successfully used for hyperparameter optimization, which we show in the next section.

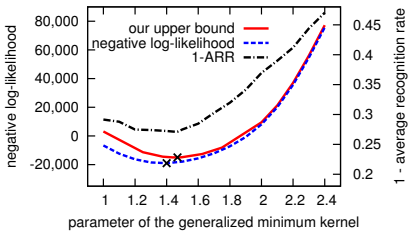
6 Experiments

We conducted experiments with several image categorization datasets. The results can be summarized as follows:

1. Using our approach, training, classification, and optimization of hyperparameters is significantly faster and has only linear memory requirement compared to baseline GP, allowing for learning on large-scale datasets.
2. Conjugate gradients with fast HIK matrix multiplications outperforms the methods of [9] and [12] in terms of convergence speed.
3. The log-determinant approximation given in Eq. (9) allows for hyperparameter optimization leading to significant performance gains.
4. Generalized histogram intersection kernels improve the classification performance significantly compared to standard HIK.
5. Determining feature relevance can be done efficiently with GP likelihood optimization and a weighted HIK.

6.1 Experimental Setup

The histogram intersection kernel is well suited for comparing histograms [4]. Therefore, all of our image categorization experiments use bag of visual words (BoV) features computed using the toolkit provided with the ILSVRC'10 database [5]. Although all types of histogram features can be utilized, we choose this basic representation without any incorporation of spatial information to focus the experiments on the machine learning part. We use the visual codebook provided with 1,000 elements. Note that the dimension of the feature vectors is an important factor for the computation time of GP large-scale inference, and the speed-up of our techniques is higher for low-dimensional features (see Table 1). As an optimization method we use the Nelder-Mead technique [16].
For multi-class classification, we use the average recognition rate (ARR) as a performance measure. Binary classification tasks are evaluated using the area under the ROC curve (AUC). To provide a fair comparison, computation times for all methods were measured on a single-core Intel 2.6GHz machine with a careful C++ implementation allowing for flexible data sizes.



Approach	ARR
GP with HIK	70.8%
GP with optimized G-HIK	72.9%
GP with optimized EXP-HIK	**74.0**%
BoV baseline [18]	64.1%
GIST features [18]	73.0%
SPMK [18]	73.4%

Fig. 3. Experiments with the normalized 15Scenes database: (*left*) comparison between upper bounded negative GP log-likelihood and real negative log-likelihood, (*right*) results of GP with adaptive kernels

6.2 Experiments with the Normalized 15Scenes Database

We use the 15Scenes database [17] for preliminary results on a medium-scale database. We follow the suggestion of [18] and scale all images to a size of 256×256 pixels to get results, which are not biased on different characteristic image sizes for specific categories. Training is done with 100 examples for each category resulting in 1,500 examples in total.

Verifying the Bound of the Negative Log-Likelihood A first experiment evaluates the upper bound of the negative GP log-likelihood presented in Sect. 5.2. The left plot in Fig. 3 shows the correct negative log-likelihood, our upper bound with respect to the hyperparameter η of a generalized HIK, and the average recognition rate when using the hyperparameter value for classification of the test set. It can be seen that our bound is sufficient for hyperparameter optimization in this setup, because the minima and the corresponding average recognition rates displayed only differ slightly. For higher values of η, our bound converges to the exact value because the influence of the log-determinant term compared to the data term of the log-likelihood decreases. Consequently, possible approximation errors become less important and the data term can be computed without any approximation even for large-scale datasets.

Different Generalized HIK The table on the right hand side of Fig. 3 gives an overview of the recognition performance we achieved on this dataset with standard HIK, G-HIK, and EXP-HIK. The hyperparameters of G-HIK and EXP-HIK have been optimized with our GP likelihood optimization technique. The latter approach resulted in the best performance and is even comparable to the result of the spatial pyramid matching kernel (SPMK) given by [18]. This highlights the power of generalized HIK and our hyperparameter optimization, because we do not incorporate any position information in our features as done in the SPMK framework.
Using the standard biased 15Scenes database with the splits and features provided by [9], we achieve an average performance of 80.0% and 79.9% with and without optimization, respectively. In contrast, the SVM solver of [9], which

Table 2. Evaluation on 200 binary classification tasks derived from the ImageNet database. Computation times are given as median values of measurements for each task (learning) and each test example (classification)

	10,090 examples ($\ell = 10$)			**50,050 examples** ($\ell = 50$)		
Method	AUC	learning time	classif. time	AUC	learning time	classif. time
GP with HIK (Cholesky)	0.836	> 3.5h	1.1s	-	-	-
GP with HIK	0.836	64s	$44\mu s$	0.856	321s	$44\mu s$
GP with optimized G-HIK	0.865	435s	$44\mu s$	0.883	2815s	$44\mu s$
GP with optimized EXP-HIK	**0.889**	579s	$44\mu s$	**0.893**	2578s	$44\mu s$

also exploits HIK properties, achieved a recognition rate of 81.3%. Nonetheless, it should be noted that our approach focuses on Bayesian inference and Bayesian hyperparameter optimization, which offers a probabilistic formulation with a wide range of further applications and extensions, *e.g.*, active and transfer learning [6, 19] as well as incorporating other noise models [13].

6.3 Large-Scale Experiments with the ImageNet Database

We also test our approach on the part of the ImageNet dataset that was used for the ILSVRC'10 competition. This dataset contains in total 150,000 images from 1,000 different categories. We apply our method to binary classification tasks of this dataset, because learning with all categories turns out to be still impractical even with our fast kernel calculations. Binary tasks are derived in a one-vs-all manner, *i.e.*, we use all images of a single class as positive examples and ℓ examples from each of the other 999 categories as negative examples. In this manner, we derive 200 tasks from the first 200 categories and use the average AUC value achieved on the ILSVRC'10 validation dataset with 50,000 examples as the resulting performance value.
The results are shown in Table 2 for $\ell = 10$ and $\ell = 50$ with 10,090 and 50,050 examples in total. First it should be noted that standard GP regression for $\ell = 50$ is not directly applicable because of limiting memory capacity ($\ell = 50$ results in a $9GB$ kernel matrix). In contrast, it can be seen that we are able to learn GP classifiers within a few minutes. Furthermore, our GP likelihood optimization method is able to handle large datasets and provides significant performance gains with hyperparameter optimization (paired t-test, $p < 10^{-7}$).

6.4 Evaluation of Linear Solvers with Fast HIK Multiplications

In the following, we compare the performance of conjugate gradients with fast HIK matrix multiplications as presented in Sect. 4 and two other coordinate descent approaches [9, 12]: (1) the coordinate descent method of [9] applied to GP and (2) the greedy block coordinate descent (GBCD) approach of [12]. The first one was originally presented for fast SVM learning with HIK and directly operates on the lookup table T (Sect. 4). GBCD calculates parts of the kernel matrix on the fly to solve sub-problems. For our experiments, the size of the

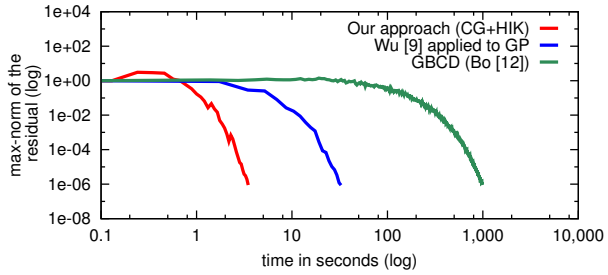


Fig. 4. Evaluation of the runtime and convergence of linear solvers: (1) our conjugate gradients method, (2) the coordinate descent method of [9], and (3) greedy block coordinate descent [12]. Note that our approach and [9] exploit fast HIK matrix multiplications, while [12] can be applied for every kernel function

sub-problems is set to 10 and the number of components κ for greedy selection is 20. We also tested other values, but did not achieve a significant speed-up.

We use a binary classification task from the ILSVRC'10 database with $\ell = 1$ (see previous paragraph) and solved the linear system $\tilde{\mathbf{K}}_{\boldsymbol{\eta}} \cdot \boldsymbol{\alpha} = \boldsymbol{y}$ with all three methods. Figure 4 shows the residual of the linear system with respect to the computation time needed. Termination is done when the maximum norm of the residual drops below 10^{-6}.

As can be seen in Fig. 4 there are orders of magnitude between all three methods. Conjugate gradients reaches a solution in 3.7 seconds, which is superior to the coordinate descent method of [9] applied to GP, which converges after $32s$. GBCD is slow (convergence after 16 minutes) due to the long time needed for explicit calculation of kernel values for 1,000-dimensional features. In the experiments of [12], only low-dimensional features ($D \leq 37$) were utilized. However, GBCD can be applied for large-scale GP regression with arbitrary kernel functions. It should also be noted that solving the linear system of GP regression needs more time than solving the optimization problem related to SVM. This is due to the additional sparsity constraints of SVM. However, the GP framework offers a proper Bayesian model with the previously mentioned advantages.

6.5 Feature Relevance Estimation

We have already seen that Gaussian Processes allow for hyperparameter optimization in a Bayesian manner. In this experiment, we show the suitability of GP equipped with optimized weighted HIK for efficient feature relevance determination leading to superior results to those of SVM-based estimations.

Since there is no exact gradient information during the optimization available, the Nelder-Mead method converges poorly for huge numbers of parameters to be optimized. Consequently, computing feature relevance for features with thousands of dimensions, as in our previous experiments, is almost impossible right

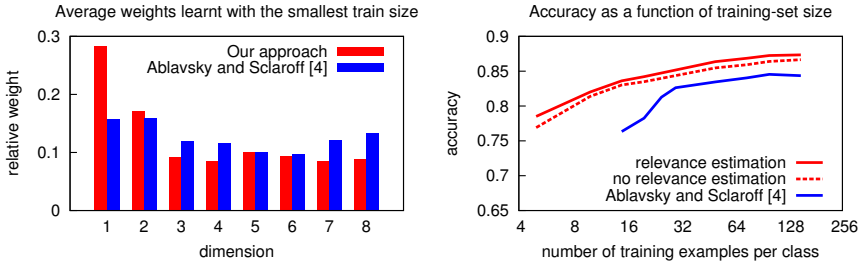


Fig. 5. Relevance determination with very generalized histogram intersection kernels and GP hyperparameter optimization. The first two features contain most of the discriminative information: (*left*) feature weights estimated with 5 examples per class, (*right*) performance compared to non-weighted histogram intersection kernels. Results are averaged over 500 runs

now. Nevertheless, as a proof of concept we follow the same synthetic experimental setup as in [4]: for different numbers of training examples, we randomly sample eight-dimensional feature vectors with relevant information only available in the first two dimensions. The performance is estimated with 500 tests. For the specific random distributions, we refer the reader to [4] and references therein. The results of our experiments can be seen in Fig. 5.
The information included in each dimension is well reflected by the estimated relative weights η_i, which can be seen in the plot on the left hand side. Furthermore, the plot on the right hand side shows the recognition accuracy for standard and weighted HIK with respect to the training size. The improvement is highly significant with $p < 10^{-7}$ using the paired t-test. In comparison with [4], our approach additionally leads to more consistent weights and higher accuracies.

7 Conclusions and Future Work

This paper presented how Gaussian Processes equipped with the histogram intersection kernel can be speeded up significantly. The involved strategies allow for training and classification in sub-quadratic and constant time with few memory requirements. This significantly overcomes the main drawbacks of GP for large-scale scenarios (cubic and quadratic runtime for training and classification, quadratic demand of memory). We further developed an efficient method for optimizing hyperparameters in a Bayesian manner by exploiting the benefits of HIK and GP as well as by providing an efficient bound of the GP marginal log-likelihood. We demonstrated the suitability of our approach on several datasets. It turned out that we are able to find suitable parameters for different parameterized histogram intersection kernels even for large-scale datasets resulting in a significant improvement of the recognition performance. Furthermore, we successfully applied our framework to feature relevance determination showing

superior results compared to state-of-the-art [4]. Our approach allows for large-scale classification with GP, which was proved in our ImageNet experiments. Future work will focus on calculating approximate gradient information of the likelihood to allow optimization with respect to a large number of parameters. Furthermore, multi-class classification could be speeded up by using label trees [20] or similar techniques. Finally, we want to extend our approach to fast computation of the predictive variance for estimating classification uncertainties. This would allow for active learning applications.

Acknowledgments. We thank Esther and Matthias Wacker for their optimization toolbox as well as the reviewers for very useful suggestions.

References

1. Boughorbel, S., Tarel, J.P., Boujemaa, N.: Generalized histogram intersection kernel for image recognition. In: ICIP. (2005) III – 161–4
2. Grauman, K., Darrell, T.: The pyramid match kernel: Efficient learning with sets of features. J. Mach. Learn. Res. **8** (2007) 725–760
3. Wang, G., Hoiem, D., Forsyth, D.: Learning image similarity from flickr groups using fast kernel machines. TPAMI **99** (2012)
4. Ablavsky, V., Sclaroff, S.: Learning parameterized histogram kernels on the simplex manifold for image and action classification. In: ICCV. (2011) 1473–1480
5. Berg, A., Deng, J., Fei-Fei, L.: Large scale visual recognition challenge (2010) http://www.imagenet.org/challenges/LSVRC/2010/.
6. Kapoor, A., Grauman, K., Urtasun, R., Darrell, T.: Gaussian processes for object categorization. IJCV **88** (2010) 169–188
7. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. In: CVPR. (2010) 3539 –3546
8. Maji, S., Berg, A., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: CVPR. (2008) 1 –8
9. Wu, J.: A fast dual method for hik svm learning. In: ECCV. (2010) 552–565
10. Barla, A., Odone, F., Verri, A.: Histogram intersection kernel for image classification. In: ICIP. (2003) 513–516
11. Quiñonero Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate gaussian process regression. J. Mach. Learn. Res. **6** (2005) 1939–1959
12. Bo, L., Sminchisescu, C.: Greedy block coordinate descent for large scale gaussian process regression. In: Uncertainty in Artificial Intelligence. (2008)
13. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning. The MIT Press (2006)
14. Yuster, R.: Matrix sparsification for rank and determinant computations via nested dissection. In: IEEE Symp. on Foundations of Computer Science. (2008) 137 –145
15. Bai, Z., Golub, G.: Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices. Annals of Num. Mathematics **4** (1997) 29–38
16. Nelder, J., Mead, R.: A simplex method for function minimization. Computer Journal **7** (1965) 308–313
17. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006) 2169–2178
18. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: CVPR. (2009) 413 –420
19. Rodner, E., Denzler, J.: One-shot learning of object categories using dependent gaussian processes. In: DAGM. (2010) 232–241
20. Bengio, S., Weston, J., Grangier, D.: Label embedding trees for large multi-class tasks. In: NIPS. (2010) 163–171