Large-Scale Gaussian Process Classification Using Random Decision Forests[1]

B. Fröhlich, E. Rodner, M. Kemmler, and J. Denzler

Chair of Computer Vision, Institute of Computer Science, Friedrich Schiller University of Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany

e-mail: bjoern.froehlich@uni-jena.de, erik.rodner@uni-jena.de, michael.kemmler@uni-jena.de, joachim.denzler@uni-jena.de

Abstract—Gaussian processes are powerful modeling tools in machine learning which offer wide applicability for regression and classification tasks due to their non-parametric and non-linear behavior. However, one of their main drawbacks is the training time complexity which scales cubically with the number of examples. Our work addresses this issue by combining Gaussian processes with random decision forests to enable fast learning. An important advantage of our method is its simplicity and the ability to directly control the trade-off between classification performance and computational speed. Experiments on an indoor place recognition task and on standard machine learning benchmarks show that our method can handle large training sets of up to three million examples in reasonable time while retaining good classification accuracy.

Keywords: Gaussian processes, random decision forests.

DOI: 10.1134/S1054661812010166

1. INTRODUCTION

Gaussian process (GP) based machine learning techniques have recently gained much attention in the field of pattern recognition and computer vision, since they provide an elegant Bayesian framework and offer state-of-the-art recognition performance [9]. However, one non-negligible disadvantage of the full GP framework is the time consumption needed during learning which scales cubically, and quadratically considering memory usage, with the size of the training data. This fact hinders their use for large-scale classification tasks which often occur in object and place recognition scenarios [5].

To overcome this shortcoming, a large amount of scientific effort has been spent in the last years to develop fast inference techniques for GP regression and classification [13]. Prominent techniques usually rely on conditional independence assumptions [12] with respect to a small set of predefined variables which might either be part of the training dataset [16] or learned during training [15]. A separate branch of techniques are based on decomposition techniques, where the original large-scale problem is broken down into a collection of smaller problems. Next to simple Bagging strategies [4], unsupervised *kd*-trees neglecting label information during clustering were recently proposed [14] for GP regression. As a supervised alternative, Broderick et al. [2] combined a Bayesian decision tree with GP classifiers.

In this paper, we propose a combination of GP classifiers and random decision forests (RDFs) to enable accurate classification in large-scale settings (cf. Fig. 1). This idea is hence strongly related to the work of [2], since it is based on supervised tree-structured decomposition techniques. In contrast, our approach is very efficient and simple as it does not need sophisticated sampling methods. It is hence ideally suited for very large datasets. Please note that parallel to the conference publication [7] related to this extended journal article, Chang et al. [3] proposed a very similar idea to accelerate the learning process of SVMs.

The remainder of the paper is organized as follows. We briefly review classification and regression with Gaussian processes, which is followed by describing random decision forests. The subsequent sections detail our approach of combining these two classifiers and experimentally demonstrate the benefits of this method on a place recognition task [10] and on very large datasets in comparison to Chang et al. [3]. A summary of our findings and a discussion of future research directions conclude the paper.

2. CLASSIFICATION WITH GAUSSIAN PROCESS PRIORS

In the following we briefly review Gaussian process (GP) regression and classification. Due to the lack of space, we concentrate on the main model assumptions and the resulting prediction equation. For a presenta-

[1]The article is published in the original.

Received October 5, 2011

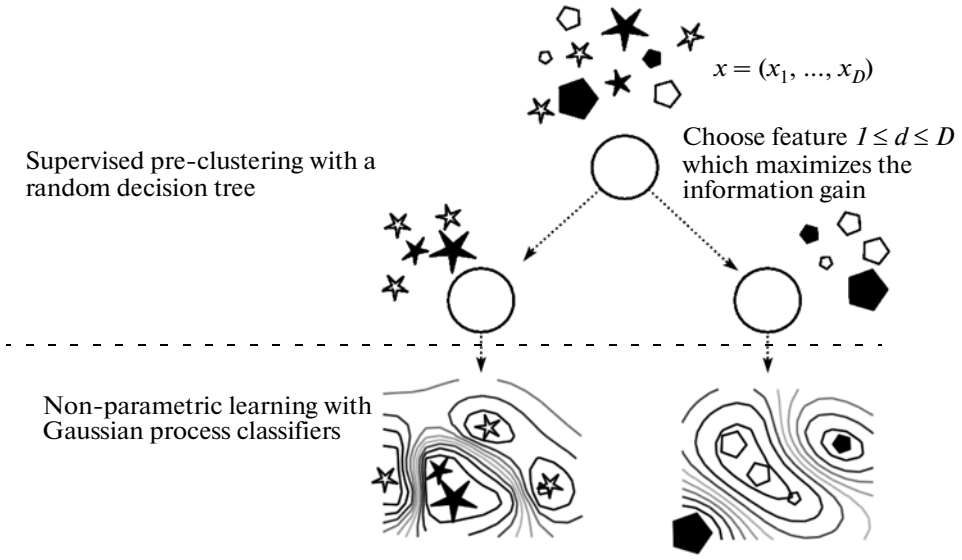

Fig. 1. An outline of our approach: RDF is used to cluster the data in a supervised manner and a GP classifier is used to separate classes in each leaf.

tion of the full Bayesian treatment we refer to Rasmussen and Williams [13].

2.1. Basic Principles of GP Priors

Given n training examples $x_i \in \mathbb{R}^D$ denoting input feature vectors and corresponding binary labels $y_i \in \{-1, 1\}$, we would like to predict the label y_* of an unseen example $\mathbf{x}_*$. Therefore a learner has to find the intrinsic relationship between inputs $\mathbf{x}$ and labels y. It is often assumed that the desired mapping can be modeled by $y = f(\mathbf{x}) + \varepsilon$, where f is a noise-free latent function (which is not observed during training) and ε denotes a noise term.

One common modeling approach is to assume that f belongs to some parametric family and to learn the parameters which best describe the training data. However, the main benefit of the GP framework is the ability to model the underlying function f directly, i.e. without any fixed parameterization, by assuming that f is a sample of a specific distribution. Defining a distribution on functions in a non-parametric manner can be done with a Gaussian process, which is a special kind of a stochastic process. An informal and rough definition would be that a GP is an infinite dimensional normal distribution.

2.2. Bayesian Framework for Regression and Classification with GP

To use the modeling ideas described in the previous section we formalize and correctly specify the two main modeling assumptions for regression and classification with Gaussian processes:

(1) The latent function f is a sample from a GP prior $f \sim GP(\mathbf{0}, K(\cdot,\cdot))$ with zero mean and covariance or kernel function K: $\mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R}$.

(2) Labels y are conditionally independent given latent function values $f(\mathbf{x})$ and are described using some noise model $p(y|f(\mathbf{x}))$.

The Gaussian process prior enables to model the correlation between labels using the similarity of inputs, which is described by the kernel function. It is thus possible to model the assumption of smoothness, i.e. that similar inputs should lead to similar labels. Please note, that this assumption is one of the underlying requirements of machine learning in general, because learning a non-smooth relation between inputs and labels without additional prior knowledge is infeasible.

For classification purposes, sigmoid functions are often employed as noise models [13]. In contrast, we follow Kapoor et al. [9] and use zero-mean Gaussian noise with variance σ_n^2:

$$p(y|f(\mathbf{x})) = N(y|f(\mathbf{x}), \sigma_n^2), \quad (1)$$

which is the standard assumption for GP regression. The advantage of this label regression approach is that tractable predictions for unseen points $\mathbf{x}_*$ are possible, without using approximate inference methods [13].

Let $\mathbf{K}$ be the kernel matrix with pairwise kernel values of the training examples $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{k}_*$ be kernel values $(\mathbf{k}_*)_i = K(\mathbf{x}_i, \mathbf{x}_*)$ corresponding to test example $\mathbf{x}_*$. The most likely outcome $\bar{y}_*$ for a given input $\mathbf{x}_*$ and labeled training data can be predicted analytically using the following equation:

$$\bar{y}_*(\mathbf{x}_*) = \mathbf{k}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}. \quad (2)$$

This prediction equation is equivalent to kernel ridge regression, but with a clear probabilistic meaning. For example, the GP framework allows for predicting the standard deviation σ_*^2 of the estimation:

$$\sigma_*^2(\mathbf{x}_*) = K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_* + \sigma_n^2. \quad (3)$$

Although we do not utilize this additional information in this paper it is directly available and might be suitable to detect instances from unseen classes.

We restricted the theoretical review of GP classification to binary tasks. However, by applying the one-vs.-all strategy in combination with a majority voting scheme, multi-class classification problems can be solved without much additional computational effort [9].

3. RANDOM DECISION FOREST

In contrast to GP techniques, a *random decision forest* (RDF [1]) is an ensemble classifier that can handle large sets of training examples with high dimensionality. This advantage is mainly due to the simplicity of the linear base classifiers (decision stumps) which cluster the feature space. Compared to standard decision tree approaches, which suffer from severe over-fitting problems, a RDF is an ensemble (forest) of T decision trees generated using randomized learning techniques [8]. Each tree is learned with only a random fraction of the available training data and the data is recursively split by axis orthogonal hyperplanes which are learned by maximizing the information gain of a randomly selected feature set. The procedure is stopped if the current set contains only examples of a single class or the number of examples falls below a certain value L.

To classify a new example $\mathbf{x}_*$, each tree t in the ensemble is traversed until a leaf node $v_t(\mathbf{x}_*)$ is reached. Each such leaf node contains a histogram which models the posterior probability $p(y_* = \kappa | v_t(\mathbf{x}_*))$ obtained during learning. The final posterior probability is then estimated by simple averaging over all leaf probabilities:

$$p(y_* = \kappa | \mathbf{x}_*) = \frac{1}{T}\sum_{t=1}^{T} p(y_* = \kappa | v_t(\mathbf{x}_*)). \quad (4)$$

Table 1. Computational complexity of all three methods considered: n denotes the number of training examples, L refers to the maximum number of examples a GP classifier is learned within a leaf, T is the number of decision trees in the forest

	Training	Classification of one Example
GP	$O(n^3)$	$O(n)$
RDF	$O(Tn\log n)$	$O(T\log n)$
RDF-GP	$O(Tn\log n + TnL^2)$	$O(T\log n + TL)$

4. COMBINING RDF AND GP

For speeding up vanilla GP classifiers, we propose to efficiently combine RDF and GP to a new classifier called RDF-GP. In principle, GP classifiers can be utilized as weak classifiers in each node of the underlying decision trees. This strategy, however, would even amplify the computational cost of the learning algorithm. The main idea of our approach is thus to learn a RDF whose inner nodes are simple decision stumps and only leaf nodes are equipped with powerful GP classifiers. It allows the description of the posterior distribution in each leaf using a non-linear decision function and to achieve a significant speed-up due to the clustering of the training data performed by the RDF. From a different perspective, we are constructing an ensemble of GP classifiers where each single classifier is trained on a preclustered subset of examples. These subsets are given by $X_v = \{\mathbf{x}_i \in \mathcal{X} | v_t(\mathbf{x}_i) = v\}$, i.e. the training data reaching the corresponding leaf nodes v. If the training set in one leaf consists only of examples belonging to the same class, the posterior probability of this class is set to one and no GP classifier is learned.

Upper bounds of the computational time for training and learning are listed in Table 1, where n is the number of training samples, T is the number of trees and L is the maximum number of examples in a leaf. The bounds assume that the learned trees of the ensemble are balanced.

5. EXPERIMENTS

In our experiments we empirically support the following hypotheses:

1) RDF-GP clearly outperforms the standard RDF;

2) RDF-GP is substantially faster than the full GP classifier approach;

3) RDF-GP allows for utilizing Gaussian process classifiers on very large datasets;

4) RDF-GP achieves similar performance compared to DT-SVM [3].

First we show the behavior of our classifier combination in comparison to GP and RDF on a place recognition application before we evaluate the algorithm

Fig. 2. Place recognition task: example images obtained from different locations and rooms. The categories in our setting are (listed as displayed from top left to bottom right) *Corridor, Elevator Area, Entrance Area, PhD Lab, Kitchen, Robot Lab, and Student Lab*.

on common machine learning benchmarks and compare it with the one of Chang et al. [3].

5.1. Multi-Sensor Place Classification

Our method is assessed using the place recognition database utilized in Kemmler et al. [10]. The goal of this application is to predict the room a robot is situated in, given the current sensor signals of the robot. By predicting the current room or scene, a rough self localization is possible which can also be further utilized in object recognition tasks [17]. The dataset of [10] consists of seven different room categories and images captured by a standard CCD camera and range data obtained using a time-of-flight sensor [11]. Example images of this dataset are depicted in Fig. 2.

We derive two classification experiments from this database: (1) a small-scale scenario using two sequences with 697 examples for training and six sequences with 2759 examples for testing, which follows directly the setup used in [10]; (2) a medium-scale scenario which was derived by swapping both sets and using the larger one for training.

5.1.1. Experimental setup of the place recognition experiment. In our experimental setting, we follow [10] and employ a combination of different image-based features using both a standard camera and a time-of-flight sensor. For both sensors, the following features are computed:

1) a bank of reduced Gabor filter responses with 8 different orientations and 4 different scales and;

2) 16 slopes and offsets of oriented Fourier power spectra.

For range images, additional information in terms of;

3) range histograms and

4) surface normal histograms are taken into account.

All features are concatenated and thus treated as one combined feature vector. As covariance function, we use a radial basis function kernel (RBF-kernel):

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2), \quad (5)$$

where γ denotes the bandwidth parameter of the kernel.

For comparison of RDF, GP classifier and our combined approach from the previous section (RDF-GP), average recognition rate is used as an unbiased accuracy measure. In our setting, we utilize RDFs with five decision trees and a varying number $L \in \{5, 6, ..., 200\}$ of the maximum number of examples in leaf nodes. Note that [10] uses more decision trees resulting in a higher recognition rate. The parameters of the kernel function can be optimized efficiently using the GP framework by maximizing the marginal likelihood of the training data [13]. However, we applied this model selection technique to each classifier in a leaf of the decision tree but without a significant performance gain. This might be due to the small amount of training data in some leaves, which leads to overfitting.

5.1.2. Evaluation of the place recognition experiments. The results of all methods for the place recognition scenario are illustrated in Fig. 3 for the small-scale and in Fig. 4 for the medium-scale experiment. The left plots show the classification accuracy and it can be seen that the GP classifier performs best and RDF clearly exhibits inferior performance compared to the other methods. Recognition rates for $\lambda \geq 100$ are not displayed for the RDF classifier, since no improvement was observed with an increasing number of examples in leaf nodes. By augmenting the RDF with GP classifiers in each node, however, progress in terms of accuracy is achieved. The accuracy increases with increasing λ which is due to the fact that more and more training examples are fed into the GP classifiers located in the leaf nodes.

In order to compare the computational demand of the respective methods, the runtime behavior for training and testing is investigated. The results on a standard office computer (2.80 GHz) are depicted in the plots on the right in Figs. 3 and 4 and training and testing times are summed up to yield one final value. The gap between RDF-GP and GP is apparent and

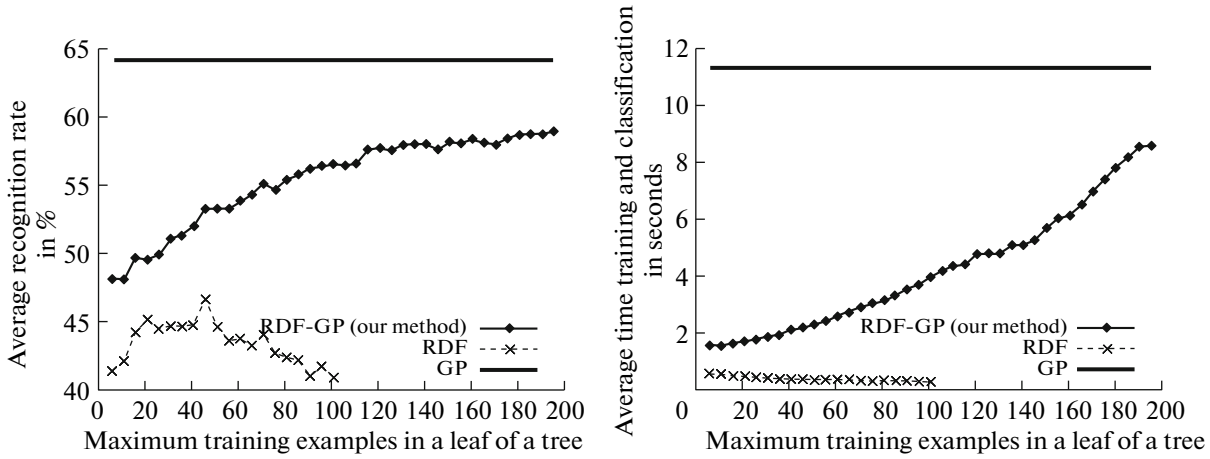

Fig. 3. Small-scale experiment using Gaussian processes, random decision forests and RDF-GP: (left) average recognition rate (right) computation time for training and testing.

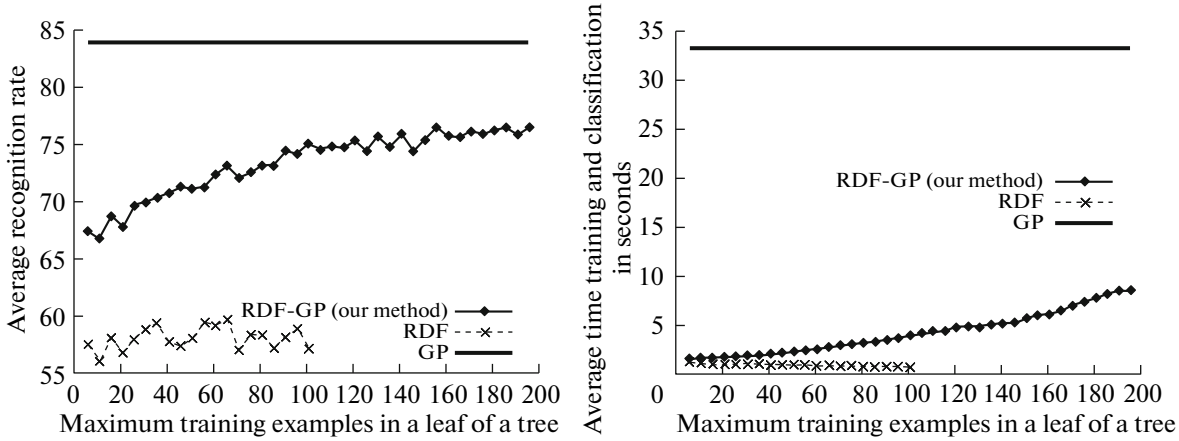

Fig. 4. Medium-scale experiment using Gaussian processes, random decision forests and RDF-GP: (left) average recognition rate (right) computation time for training and testing.

corresponds to a significant speedup, which is more prevalent in the medium scale experiment. Increasing the parameter L leads to a slower runtime as expected by the theoretical analysis given in Table 1.

These experiments validate hypotheses 1 and 2, which state that RDF-GP clearly outperforms RDF for large L and RDF-GP is substantially faster than the full GP classifier approach for a large number of samples for training.

5.2. Large-Scale Problems and Comparison with Previous Work

In the following experiments, we show that RDF-GP is also applicable for very large databases. In most of these settings, standard GP classifiers are not applicable due to high memory requirements and cubic runtime behavior.

5.2.1. Experimental setup of the large-scale problems. The datasets used for comparison are described in Table 2. We follow [3] and use the same experimental setup as proposed in their article to allow a fair comparison. Each dataset is split into three parts, where two thirds are used for training and the last third is equally split into a validation and test set. According to this partition strategy, the problem size for training ranges between seven thousand and three million examples. Unlike [3], we are thus not able to compare our approach to the original (GP) classifier for every dataset since the Cholesky decomposition of $\mathbf{K}$ demands memory capacity beyond available resources.

The parameters L and γ are computed automatically by maximizing recognition accuracy on the validation datasets. The upper bound for λ is given by the limited memory capacity (12GB in our case) and usually varies between 200 and 5 000. Since GP classification has no tunable trade-off parameter C, optimization only takes place in the two-dimensional space using the heuristics mentioned in [3] and a modified version of the related available implementation.[1]

[1] http://ocrwks11.iis.sinica.edu.tw/~dar/Download/WebPages/

Table 2. Details of large-scale databases used in our experiments and the evaluation of [3]

Data set	Classes	Dimension	Training	Validation	Testing	All
Pen Hand Written (PHW)	10	16	7227	1872	1893	10992
Letter	26	16	13294	3336	3370	20000
Shuttle	7	9	38664	9573	9763	58000
Poker	10	10	16674	4165	4171	25010
Consus Income (CI)	2	14	30148	7537	7537	45222
Forest	7	54	387343	96835	96834	581012
PPI	2	14	836544	206635	206635	1249814
KDD	5	41	3265623	816405	816403	4898431

Table 3. Recognition rates obtained on different large datasets. DT: decision tree, DT-GP: decision tree in combination with GP classifier, DTSVM: decision tree in combination with SVM, LIBSVM: SVM using RBF kernel, LIBLINEAR: SVM using linear kernel, GP-Reg: Gaussian process regression with RBF Kernel

	PHW	Letter	Shuttle	Poker	CI	Forest	PPI	KDD
DT-GP (ours)	99.31	95.31	99.92	55.79	**84.99**	93.90	**92.29**	**99.99**
DTSVM [3]	99.52	**97.66**	99.89	**56.75**	84.81	**94.59**	**92.29**	**99.99**
DT [3]	95.51	87.18	**99.95**	49.72	80.97	93.31	88.11	**99.99**
LIBSVM [3]	**99.63**	**97.66**	99.91	56.25	84.13	–	–	–
LIBLINEAR [3]	91.13	68.58	91.95	43.30	80.54	71.50	87.43	99.72
GP-Reg [13]	99.32	95.97	–	–	–	–	–	–

It was noted in [3] that a single deterministic decision tree (DT) in combination with SVMs was sufficient and multiple trees did not lead to superior results. For a fair comparison, we will follow this approach by using a DT instead of an RDF in the following experiments. The resulting combined classifier will be denoted by DT-GP.

5.2.2. Evaluation of the large-scale experiments. The results of DT-GP and the method proposed in [3] (DTSVM) are illustrated in Table 3. To enable a comparison of the involved base classifiers, we further added the results of standard decision trees, standard SVM (LIBSVM) and linear SVM (LIBLINEAR) published in [3] and the results obtained with a Gaussian process classifier.

While all tree-based classifiers can cope with very large amounts of data, standard SVM cannot be applied on three datasets. Although this fact can be circumvented by utilizing an efficient linear SVM implementation [6], computational feasibility is accompanied by a loss in prediction accuracy due to the low capacity of linear decision functions. This behavior is not shared by DT-GP, DTSVM and DT, since both tractability and flexibility are simultaneously achieved. Both treed kernel methods achieve excellent accuracies and substantially outperform standard decision trees in most cases. This highlights the additional modeling power embedded in DT-GP and DT-SVM by utilizing non-linear decision boundaries. These findings clearly support hypothesis 3 and 4.

We also analyzed the speed of DT-GP on a standard office computer (one core of an Intel(R) Core(TM) i7 CPU 930 2.8G Hz). As opposed to [3], we decided to provide absolute runtimes (in seconds) to give an impression of how DT-GP scales for real world applications. The performances are given in Table 4 and clearly support the suitability of our approach to large-scale problems, since all experiments were completed in at most a few minutes. The highest computational load was measured for the second largest dataset (PPI) with a duration of approximately 25 minutes including both training and testing. It is hence important to note, that the runtime does not solely depend on the number of examples and the dimensionality of the input space. This stems from the fact that the partitioning of the feature space accomplished by the DT has a high effect on speed. If the DT favors large homogeneous leaf nodes, only very few GP classifiers with a moderate size of training examples are learned on the remaining heterogeneous leaves.

Table 4. Runtimes for training and testing a DT-GP classifier (in seconds)

	PHW	Letter	Shuttle	Poker	CI	Forest	PPI	KDD
Training	59	181	0.9	97	196	930	1436	243
Testing	0.90	1.84	0.03	1.41	0.88	5.27	34.3	1.27

6. CONCLUSIONS AND FURTHER WORK

This work presents a new approach for boosting the runtime behavior of Gaussian process (GP) classifiers using a random decision forest (RDF) to compute a pre-clustering of the input space. We empirically evaluate this combined classifier on a place recognition task containing a moderate number of examples. While maintaining a good level of accuracy, our method is substantially faster than the full GP classifier. Moreover, in order to assess the suitability for large-scale scenarios, several benchmark datasets containing up to three million training examples are utilized. The latter experiment is also used to compare against [3], a combination of SVM and decision trees, which was published in parallel to our conference publication [7]. Results clearly show that our combined classifier can easily cope with very large amounts of data, as all computation was done in several minutes and encouraging recognition rates, comparable to those in [3], are produced.

One important topic is hyperparameter estimation in the proposed framework. It would be interesting to investigate whether the possibility to automatically search for appropriate hyperparameters can be inherited from GP classifiers to improve upon initial parameter estimates without invoking external validation experiments. This future research direction also extends to automatic selection of suitable kernels and multiple kernel learning.

In [3], theoretical error error bounds for treed SVMs are derived using results from statistical learning theory. It is an interesting problem whether it is possible to extend their results to more general situations which also include our method.

ACKNOWLEDGMENTS

We would like to thank the authors of [3] for releasing the source code of their approach and the corresponding details of the experimental setup.

REFERENCES

1. L. Breiman, "Random Forests," Mach. Learn. **45** (1), 5–32 (2001).
2. T. Broderick and R. B. Gramacy, "Treed Gaussian Process Models for Classification," in *Classification as a Tool for Research, Studies in Classification, Data Analysis, and Knowledge Organization* (2010), pp. 101–108.
3. Fu Chang, Chien-Yang Guo, Xiao-Rong Lin, and Chi-Jen Lu, "Tree Decomposition for Large–Scale SVM Problems," J. Mach. Learn. Res. **11**, 2935–2972 (2010).
4. Tao Chen and Jianghong Ren, "Bagging for Gaussian Process Regression," Neurocomputing **72** (7–9), 1605–1610 (2009).
5. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A Largescale Hierarchical Image Database," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (Miami Beach, 2009).
6. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "Liblinear: A Library for Large Linear Classification," J. Mach. Learn. Res. **9**, 1871–1874 (2008).
7. B. Fröhlich, E. Rodner, M. Kemmler, and J. Denzler, "Efficient Gaussian Process Classification Using Random Decision Forests," in *Proc. 10th Int. Conf. on Pattern Recognition and Image Analysis: New Information Technologies (PRIA–10–2010)* (St. Petersburg, 2010).
8. P. Geurts, D. Ernst, and L. Wehenkel, "Extremely Randomized Trees," Mach. Learn. **63** (1), 3–42 (2006).
9. A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Gaussian Processes for Object Categorization," Int. J. Comput. Vision **88** (2), 169–188 (2010).
10. M. Kemmler, E. Rodner, and J. Denzler, "Global Context Extraction for Object Recognition Using a Combination of Range and Visual Features," in *Dyn3D'09: Proc. DAGM 2009 Workshop on Dynamic 3D Imaging* (Jena, 2009), pp. 96–109.
11. R. Lange, 3D "Time–of–Flight Distance Measurement with Custom Solid-State Image Sensors in CMOS/CCD–Technology," PhD Thesis (University of Siegen, 2000).
12. J. Q. Candela and C. E. Rasmussen, "A Unifying View of Sparse Approximate Gaussian Process Regression," J. Mach. Learn. Res. **6**, 1939–1959 (2005).
13. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (The MIT Press, 2005).
14. Y. Shen, A. Ng, and M. Seeger, "Fast Gaussian Process Regression Using kd–Trees," in *Advances in Neural Information Processing Systems 18*, Ed. by Y. Weiss, B. Scholkopf, and J. Platt (MIT Pressm 2006), pp. 1225–1232.
15. E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes Using Pseudoinputs," in *Advances in Neural Information Processing Systems 18*, Ed. by Y. Weiss, B. Scholkopf, and J. Platt (MIT Press, 2006).
16. M. E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," J. Mach. Learn. Res. **1**, 211–244 (2001).
17. A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin, "Context–Based Vision System for Place and Object Recognition," in *ICCV'03: Proc. 9th IEEE Int. Conf. on Computer Vision* (Nice, 2003), p. 273.

Björn Fröhlich, born in 1984, earned the degree "Diplom-Informatiker" from the Friedrich Schiller University of Jena in the year 2009. He is currently a holder of a scholarship in the Graduate School on Image Processing and Image Interpretation from the Free State of Thuringia (Germany) and a PhD student at the chair of Computer Vision, Institute of Computer Science, Friedrich Schiller University in Jena. Research interests are focused on object recognition and image segmentation.

Michael Kemmler, born in 1983, received the Diploma degree in Computer Science with honors in 2009 from the Friedrich Schiller University of Jena, Germany. As a PhD student of the Jena Graduate School for Microbial Communication, he is currently pursuing his studies under supervision of Joachim Denzler at the Chair for Computer Vision, University of Jena. His research interests are in the area of machine learning, object recognition and bioinformatics, including kernel methods, visual image and scene classification as well as bacterial classification.

Erik Rodner, born May 22, 1983 received the Diploma degree in Computer Science with honours in 2007 from the University of Jena, Germany. He is currently a PhD student under supervision of Joachim Denzler at the Chair for Computer Vision, University of Jena. His research interests include transfer learning, kernel methods and visual object recognition.

Joachim Denzler born in 1967, earned the degrees "Diplom-Informatiker", "Dr.-Ing.", and "Habilitation" from the University of Erlangen in the years 1992, 1997, and 2003, respectively. Currently, he holds a position of full professor for computer science and is head of the Chair for Computer Vision, Faculty of Mathematics and Informatics, Friedrich-Schiller-University of Jena. His research interests comprise active computer vision, object recognition and tracking, 3D reconstruction, and plenoptic modeling, as well as computer vision for autonomous systems. He is author and coauthor of over 90 journal papers and technical articles. He is a member of the IEEE, IEEE computer society, DAGM, and GI. For his work on object tracking, plenoptic modeling, and active object recognition and state estimation, he was awarded the DAGM best paper awards in 1996, 1999, and 2001, respectively.