Fast Sparse Gaussian Process Classification With Multiple Classes

Matthias Seeger
University of California at Berkeley
Soda Hall
Berkeley, US
mseeger@cs.berkeley.edu

Michael I. Jordan
University of California at Berkeley
Soda Hall
Berkeley, US
jordan@cs.berkeley.edu

Abstract

The favourable scaling behaviour of sparse approximations to Bayesian inference for Gaussian Process models makes them attractive for large-scale applications. We show how to generalize the *Informative Vector Machine (IVM)* [3] to a multi-way classification setting. While being a kernel-based approach, our method yields valid uncertainty estimates and allows for hyperparameter adaption via optimization. Our scheme operates *jointly* on the data for all classes. Crucially, we still achieve a *linear* scaling in both the number of classes and the number of training points.

1 Introduction

The *Informative Vector Machine (IVM)* [3, 6] is a sparse approximation to Bayesian inference for *Gaussian process (GP)* models using a single process. Here we extend the IVM framework to GP classification models with $C > 2$ classes. This is achieved in $O(n\,C\,d^2)$ running time and $O(n\,C\,d)$ memory requirements, linear both in the training set size n and the number of classes C, thus applicable to large tasks.[1] Our method provides an efficient Bayesian treatment of multi-way classification GP models which has not been given so far. We show how the formidable representational problem of multiple coupled processes can be addressed efficiently. We also argue that our method can be used as a major building block to address nonparametric inference and learning in larger structured graphical models employing kernels.

In contrast to many suggested multi-way extensions of the binary *support vector machine (SVM)* classifier which attempt posthoc combinations of sequences of independent binary maximum margin discriminants, our method uses the sample as a whole and can benefit from the *joint* information about all classes provided by every case. Using an efficient approximation to Bayesian inference, we obtain predictive probability estimates ("error bars") and can use powerful methods to learn free hyperparameters. We do not know of previous kernel-based approaches sharing

[1] In all our scaling quotes we assume that $C \ll d \ll n$ which is the practically interesting case.

these features and the efficient scaling of our approach.

The structure of the paper is as follows. We motivate the main problems and our approach in Section 2 and describe our inference approximation in Section 3. Section 4 is devoted to a powerful hyperparameter learning scheme which even in the case of the previous binary IVM is novel. We give experimental results on handwritten digits tasks in Section 5 and close with a discussion in Section 6. Most technical details do not have space here, we refer to [7]. We assume familiarity with the binary IVM scheme [3, 6]. Our notation is described in detail in [7, 6].

2 Motivation

Given multi-way data $D = \{(\boldsymbol{x}_i, y_i),\, i = 1, \ldots, n, \boldsymbol{x}_i \in \mathcal{X}, y_i \in \{1, \ldots, C\}\}$ assumed to be drawn i.i.d. from an unknown source, we want to estimate predictive probabilities $P(y|\boldsymbol{x}_*)$ at points $\boldsymbol{x}_*$ of interest. We use a latent variable $\boldsymbol{u} = (u^{(c)})$ for the natural parameters in the multinomial ("softmax") likelihood

$$P(y|\boldsymbol{u}) = \exp\left(v_y - \log \mathbf{1}^T \exp(\boldsymbol{v})\right), \quad \boldsymbol{v} = \boldsymbol{u} + \boldsymbol{\beta},$$

where $\boldsymbol{\beta} = (\beta_c)_c$ are intercept parameters. Given C positive definite covariance functions $K^{(c)}$, we employ *independent* zero-mean GP priors with kernels $K^{(c)}$ on the component functions $u^{(c)} : \mathcal{X} \to u^{(c)}$. In our context, a zero-mean GP is simply a way of mapping finite subsets of $\mathcal{X}$ to joint zero-mean Gaussian distributions in a consistent manner (see [8] for details). Most known approaches to approximate inference in such models employ a Gaussian process approximation[2] to the posterior processes $(u^{(c)}(\cdot))_c$, as we do here. These processes are dependent due to the coupling in the likelihood. We write $\boldsymbol{u} = (u_i^{(c)})_{i,c} \in \mathbb{R}^{nC}$ and $\boldsymbol{K} = (\delta_{c,c'} K^{(c)}(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,c,j,c'}$ for the prior covariance (or kernel) matrix. Note that $\boldsymbol{K}$ has block-diagonal structure.

Our notation is as follows. Subset indexing is defined as $\boldsymbol{A}_{I,J} := (\alpha_{i,j})_{i \in I, j \in J}$, I, J ordered index sets, $\cdot$ the full index, $\boldsymbol{A}_I := \boldsymbol{A}_{I,I}$. Vectors $(\alpha_i^{(c)})_{i,c}$ are indexed over datapoints i and classes c, the former changes faster.[3] We "overload" our subscript notation to only select w.r.t. i, so $\boldsymbol{A}_{I,J}$ is short for $\boldsymbol{A}_{I \times \mathcal{C}, J \times \mathcal{C}}$, $\mathcal{C} = \{1, \ldots, C\}$.

The coupling of the posterior processes provide a serious feasibility problem in most C GP process models, and one of the main contributions here is to show up a fairly general solution. Recall that the IVM approximation amounts to replacing the likelihood factors (sites) by Gaussian *site approximations* with precision matrix $\boldsymbol{\Pi}_i$ (a scalar in the binary case). Thus, the approximate posterior covariance matrix is

$$\boldsymbol{A} = \left(\boldsymbol{K}^{-1} + \boldsymbol{\Pi}\right)^{-1}. \tag{1}$$

Here, $\boldsymbol{K}$ is block-diagonal in our grouping, but $\boldsymbol{\Pi}$ is block-diagonal (with C-by-C blocks) only in a *different* grouping, and $\boldsymbol{A}$ does not have a simple structure. This leads to at least $O(n\,(Cd)^2)$ for a direct IVM extension. Our improvement on that is motivated by the Laplace approximation framework of [9]. There, $\boldsymbol{A}$ has the form (1), but the $\boldsymbol{\Pi}_i$ blocks have *additional structure* as Hessians[4]

$$\boldsymbol{\Pi}_i = -\nabla\nabla_{\boldsymbol{u}_i} \log P(y_i|\hat{\boldsymbol{u}}_i) = \operatorname{diag} \boldsymbol{p}_i - \boldsymbol{p}_i \boldsymbol{p}_i^T, \quad \boldsymbol{p}_i = (P(y|\hat{\boldsymbol{u}}_i))_y, \tag{2}$$

[2] The process itself is not Gaussian, because the likelihood is not.

[3] In this ordering $\boldsymbol{K}$ is block-diagonal.

[4] In other words, the log posterior is replaced by a concave quadratic centered at the mode, *i.e.* by a second-order approximation.

$\hat{\boldsymbol{u}}$ the mode of $P(\boldsymbol{u}|D)$. While we *do not* adopt a Laplace approximation approach in any sense,[5] we will enforce the special form

$$\boldsymbol{\Pi}_i = \operatorname{diag} \boldsymbol{\pi}_i - \alpha_i^{-1} \boldsymbol{\pi}_i \boldsymbol{\pi}_i^T, \quad \alpha_i = \mathbf{1}^T \boldsymbol{\pi}_i,\ \boldsymbol{\pi}_i \succ \mathbf{0} \tag{3}$$

on the site precision parameters $\boldsymbol{\Pi}_i$ in order to obtain an efficient representation. Note that this form allows to capture the coupling due to the i-th datapoint up to second order and renders approximations to the posterior processes $u^{(c)}$ which are dependent in a non-trivial way.

3 Conditional Inference

In this section we develop a sparse representation of an approximation $Q(\boldsymbol{u})$ to the posterior $P(\boldsymbol{u}|D, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ collects hyperparameters (kernel parameters and $\boldsymbol{\beta}$). All details can be found in [7]. We have $Q(\boldsymbol{u}) \propto N(\boldsymbol{u}|\mathbf{0}, \boldsymbol{K}) \exp(-(1/2)\boldsymbol{u}^T \boldsymbol{\Pi} \boldsymbol{u} + \boldsymbol{b}^T \boldsymbol{u})$, where $\boldsymbol{b}$, $\boldsymbol{\Pi} = \operatorname{diag}(\boldsymbol{\Pi}_i)_i$ are *site parameters*. We maintain an *active set* $I \subset \{1, \ldots, n\}$, $|I| = d$ of "maximally informative" datapoints and only allow $\boldsymbol{b}_I$, $\boldsymbol{\Pi}_I$ to be non-zero, see [6] for the rationale behind such *sparse approximations*. In the single process case, this reduces the complexity from $O(n^3)$ to $O(n\, d^2)$ but requires new techniques for selecting I. As motivated above, for $C > 2$ we restrict $\boldsymbol{\Pi}_i$, $i \in I$ to be of form (3). In addition to the coupling problem already mentioned, there is a number of new challenges not present in the single process case: the representation and its update must scale linearly in C, the site parameter updates are complicated by the presence of the constraint (3) and C-dimensional intractable integrals occur instead of 1-dimensional ones. We address all of these problems here.

In [7] we develop a representation of Q which retains couplings between the $u^{(c)}$, yet allows for efficient predictions on datapoints The rough idea is that $\boldsymbol{\Pi}$ is a diagonal plus a rank-d matrix. The diagonal term leads to C independent binary IVM representations which can be extended using low-rank update formulae.[6] The representation size is $O(C\, d^2)$. For a point $\boldsymbol{x}_*$ the Gaussian approximation $Q(\boldsymbol{u}_*) = N(\boldsymbol{h}_*, \boldsymbol{A}_*)$, $\boldsymbol{u}_* \in \mathbb{R}^C$ can be computed in $O(C\, d^2 + C^2\, d)$ based on this representation (apart from computing the kernel values $\boldsymbol{K}_{I,*}^{(c)}$). Then, a $P(y_*|\boldsymbol{x}_*, D)$ approximation can be obtained using quadrature (see comments below).

The main novelty of the IVM was that I was selected in a very efficient way based on information-theoretic criteria. In order to score a pattern $i \notin I$ the IVM way, we need its marginal posterior $Q(\boldsymbol{u}_i) = N(\boldsymbol{h}_i, \boldsymbol{A}_i)$, and in order to score most patterns for each new inclusion we cannot afford $O(C\, d^2)$ for each. Similar as in the binary case (but slightly more complicated), we can maintain *stub buffers* $\boldsymbol{M}^{(c)}$, $\boldsymbol{Q}^{(c)} \in \mathbb{R}^{nd}$, $c = 1, \ldots, C$ based on which we can score $O(n/C)$ patterns in time $O(n\, C\, d)$. These buffers and their update dominates time and memory requirements of our scheme, the price to be paid for a good selection of I.[7] They can be discarded after training.

Our representation and stub buffers can be updated in time $O(n\, d\, C)$ once a new $i \notin I$ is selected for inclusion and its site parameters $\boldsymbol{b}_i$, $\boldsymbol{\pi}_i$ are determined. Our pro-

[5] Experimental evidence suggests that using ADF (or ADATAP) projections can significantly outperform the Laplace method [4, 5], also a sparse approximation is more natural.

[6] Importantly our scheme has the same dominating complexity than running C independent binary IVMs.

[7] Experiments show that a random selection of I performs poorly at least in the classification context. For the IVM, *randomized greedy selection* was proposed in [3] to improve the scaling by a constant factor. Similar ideas are left for future work in the multi-way case.

cedure uses Cholesky factor updates and is numerically very stable. It is important to note that the running time is dominated by a few very large linear operations for which highly optimized software (our implementation uses BLAS) is publicly available.[8]

If $i \notin I$ is chosen for inclusion into I, its site parameters are determined by an *ADF projection* [5, 4]. The idea is to consider the *tilted* distribution $\hat{P}(\boldsymbol{u}) \propto Q(\boldsymbol{u})P(y_i|\boldsymbol{u}_i)$ and project it back onto a Gaussian retaining mean and covariance matrix. Since the likelihood factor depends on $\boldsymbol{u}_i$ only, it is enough to determine the moments $\hat{\boldsymbol{h}}_i$, $\hat{\boldsymbol{A}}_i$ of $\hat{P}(\boldsymbol{u}_i)$ which given $Q(\boldsymbol{u}_i)$ requires C-dimensional integration against a Gaussian weight function for which we use quadrature (see below). The constraint (3) turns the projection[9] into a C-dimensional non-convex optimization problem (the mean can be matched exactly by choosing $\boldsymbol{b}_i$ properly). Fortunately, the problem has much structure, namely the criterion to be minimized is the sum of a convex function in $\boldsymbol{\pi}_i$ and the concave quadratic $-\alpha_i^{-1}\boldsymbol{\pi}_i^T\hat{\boldsymbol{A}}_i\boldsymbol{\pi}_i$. We use a simple double loop scheme which solves the projection reliably and efficiently. For a non-convex problem initialization matters, and we make use of the analogy to the Laplace approximation once more. Recall from (2) that $\boldsymbol{\pi}_i = (P(y|\hat{\boldsymbol{u}}_i))_y$ is chosen there which is close to $\boldsymbol{\delta}_{y_i} = \boldsymbol{I}_{\cdot,y_i}$ for many patterns (if the data fit is any good). This suggests choosing the starting value for $\boldsymbol{\pi}_i$ as convex combination of $\boldsymbol{\delta}_{y_i}$ and the predictive vector $(P(y|\boldsymbol{x}_i, D))_y$ based on the current $Q(\boldsymbol{u}_i)$.

The scoring criteria used in the binary IVM are easily adapted. The (instantaneous) information gain is $-\mathrm{D}[Q^{new}(\boldsymbol{u}_i) \,\|\, Q(\boldsymbol{u}_i)]$, $Q^{new}(\boldsymbol{u}_i)$ the posterior marginal *after* inclusion of i. This criterion motivates to call the selected points maximally "informative". We can also use a slightly different distribution for $Q^{new}(\boldsymbol{u}_i)$, namely the Gaussian with mean and covariance of $\hat{P}(\boldsymbol{u}_i)$. In this case we do not have to apply the ADF projection, which we preferred in our experiments here.[10]

A schematic overview of our conditional inference scheme is given in Algorithm 1. A number of potential extensions are described in [7], but their significance is yet unclear. The stub buffers can be discarded at the end, but have to be retained if hyperparameters are to be learned (see Section 4). As in the binary IVM, the first 2 or 3 inclusion candidates are selected at random. The *selection index* J of size $O(n/C)$ in Algorithm 1 is updated by retaining a fraction of top-scorers and completion at random. As in the binary IVM, a strong point about the method is that only a small part of the complete kernel matrix $\boldsymbol{K}$ has to be evaluated at all, namely the $\boldsymbol{K}^{(c)}_{\cdot,I}$. Thus, the IVM is particularly attractive if kernel computations are expensive.

A C-dimensional quadrature routine is essential in our scheme. We have tried a product rule based on 3 point Gauss-Hermite and an exact monomials rule of degree 5 (see [1]). The former is more stable and usually more accurate, but more expensive to evaluate. In future work we will explore other more recent rules. The need for a fairly accurate "black box" for Gaussian expectations over $P(y|\boldsymbol{u}_i)$ and $\log P(y|\boldsymbol{u}_i)$ (see Section 4) certainly limits the size of C we can deal with in our scheme.[11]

---

[8] In contrast to popular SVM optimization schemes which are dominated by a very large number of small nonlinear operations.

[9] Moment matching is equivalent to a relative entropy minimizing where the "variational" distribution is on the right side, in contrast to variational mean field methods where it is on the left.

[10] Apart from being cheaper, it might even be the more suitable criterion to score i.

[11] For a concrete likelihood, one might be able to reduce the intractable dimensionality to $< C$.

Algorithm 1 Multi-way IVM conditional inference scheme

$I = \emptyset$. Representation and stub buffers are empty.
repeat
  for $j \in J$ **do**
    Compute $\Delta_j = -\mathrm{D}[Q^{new}(\boldsymbol{u}_j) \,\|\, Q(\boldsymbol{u}_j)]$, $Q(\boldsymbol{u}_i)$ from stubs. This needs quadrature for the tilted moments $\hat{\boldsymbol{h}}_i$, $\hat{\boldsymbol{A}}_i$.
  end for
  $i = \mathrm{argmax}_{j \in J} \Delta_j$
  Compute $\boldsymbol{b}_i$, $\boldsymbol{\pi}_i$ by restricted ADF projection.
  Include i into I and update the representation. The dominating cost is in updating the stub buffers.
  Re-select J from $\{1, \ldots, n\} \setminus I$.
until $|I| = d$

4 Hyperparameter Learning

A powerful and efficient way of learning hyperparameters $\boldsymbol{\theta}$ in the binary IVM based on the conditional inference approximation was given in [6], but has not been published elsewhere so far. We minimize an upper bound on the negative log marginal likelihood

$$-\log P(\boldsymbol{y}|\boldsymbol{\theta}) = -\log \int P(\boldsymbol{y}, \boldsymbol{u}|\boldsymbol{\theta})\, d\boldsymbol{u} \leq \mathrm{E}_Q\left[-\log P(\boldsymbol{y}|\boldsymbol{u}, \boldsymbol{\theta})\right] + \mathrm{D}[Q(\boldsymbol{u}_I|\boldsymbol{\theta}) \,\|\, P(\boldsymbol{u}_I|\boldsymbol{\theta})].$$

In variational Bayesian inference, we would choose Q as well as $\boldsymbol{\theta}$ to minimize the bound. Here, we employ our sparse inference approximation to choose Q. Our choice is motivated in the same way as the variational mean field one and may be preferable to choosing Q from a more limited variational set. A drawback of our choice in practice is that the optimization is not strictly a descent method, the criterion can increase when Q is re-selected. This requires a non-standard optimization approach. If we fix I and the site parameters, the criterion is smooth in $\boldsymbol{\theta}$ and can be minimized by a Quasi-Newton code. Now and then, we re-select I, $\boldsymbol{b}_I$, $\boldsymbol{\Pi}_I$ and restart the optimizer along the negative gradient. Note that as opposed to much earlier work in variational learning, we do not keep all of Q fixed. Q has a strong direct dependence on $\boldsymbol{\theta}$ through the kernel matrices, and we honor that in the gradient computation. Details of the rather involved derivation are in [7]. We need a $O(n\, C\, d^2)$ precomputation plus $O(n\, d)$ for every kernel parameter (the kernel derivative matrices also have to be computed), so the scaling is linear in $|\boldsymbol{\theta}|$. Nothing stops us in principle from using a large number of hyperparameters.[12]

The optimization is somewhat harder to run than in the binary IVM case. Apart from the larger $\boldsymbol{\theta}$ and heavier scaling, C-dimensional quadrature routines are less accurate, so that the correspondence between gradient and finite differences of the criterion can have significant errors. However, this typically does not cause much harm in practice.

5 Experiments

We present some experiments on subsets randomly extracted from the set of even digits in the MNIST[13] traning set ($C = 5$ classes). We compare our method against

[12] This is important because we have C kernels instead of only one.
[13] See *http://www.research.att.com/ ~yann/exdb/mnist/index.html*.

Method	train5 (800)			train8 (5000)			train9 (8000)		
	err	lh	time	err	lh	time	err	lh	time
binary	0.0750	-	445	0.0232	-	6063	0.0240	-	15538
multi	0.0717	0.6368	2390	0.0420	0.7139	10772	0.0325	0.7396	43171

Table 1: Test error (err), test set likelihood (lh), running time (time; secs) for methods *binary*, *multi*, different datasets.

using C binary IVM models fitted to the c-against-rest splits, predicting y_* for $\boldsymbol{x}_*$ as argmax over log predictive probability outputs of each binary "expert". Call this setup *binary*, ours *multi*. We employ the hyperparameter learning method of Section 4 which is novel for both schemes.

We use the following tasks (n: training set size; m: test set size; d: active set size): *train5* ($n = 800, m = 200, d = 150$), *train8* ($n = 5000, m = 2500, d = 400$), *train9* ($n = 8000, m = 2000, d = 600$). Note that *binary* can select different I for each class which should give it an advantage. We use an independent RBF covariance function

$$K^{(c)}(\boldsymbol{x}, \boldsymbol{x}') = v^{(c)} \exp\left(-\frac{w^{(c)}}{2p} \|\boldsymbol{x} - \boldsymbol{x}'\|^2\right) + v_b^{(c)}, \quad \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^p$$

for each class c, thus a total of $4\,C$ hyperparameters. Optimizers were run for $25-35$ iterations, each iteration involves several criterion computations in the line search. The optimizers were restarted every 10 iterations. Results are given in Table 1, they are a single run for *train8*, averaged over 3 for *train5*. In Figure 1 we plot ROC curves (left side) and learning curves for *multi* (right side). Note how the average test set likelihood, i.e. the judgment of the model about the (unknown) true labels is increasing as learning progresses, the test error curve gives less of a clear signal.

While these experiments are preliminary, we see that the present version of our scheme does not yet improve on the simpler binary combination scheme. While both methods have the same scaling, *multi* has a larger constant factor so the differences in running time are not surprising. A major reason for the difference in predictive performance may be that *binary* can choose a different active set of size d for every class. We are currently exploring an idea to overcome this drawback of our scheme. The idea is that if some components in the site parameter $\boldsymbol{\pi}_i$, $i \in I$ are set to 0, the resulting $\boldsymbol{\Pi}_i$ is a smaller block of the form (3). If we keep the $\boldsymbol{\pi}_i$ sparse (while not completely zero), we can allow for a larger I, because the factor $C\,d$ in the dominant scaling term is replaced by the number of nonzero $\boldsymbol{\pi}_i$ components. We will explore this idea in future work and provide experiments of larger scale to better assess the qualities of our method.

6 Discussion. Future Work

We have given a sparse approximation to Bayesian inference for multi-class Gaussian process models which achieves a scaling linear in the number of training points and the number of classes. The central idea is to allow the likelihood couplings to be represented exactly up to second order while still obtaining a representation linear in the number of classes. This idea could be applied to other C-process models in much the same way.

Compared to previous kernel-based large margin multi-class algorithms, our scheme

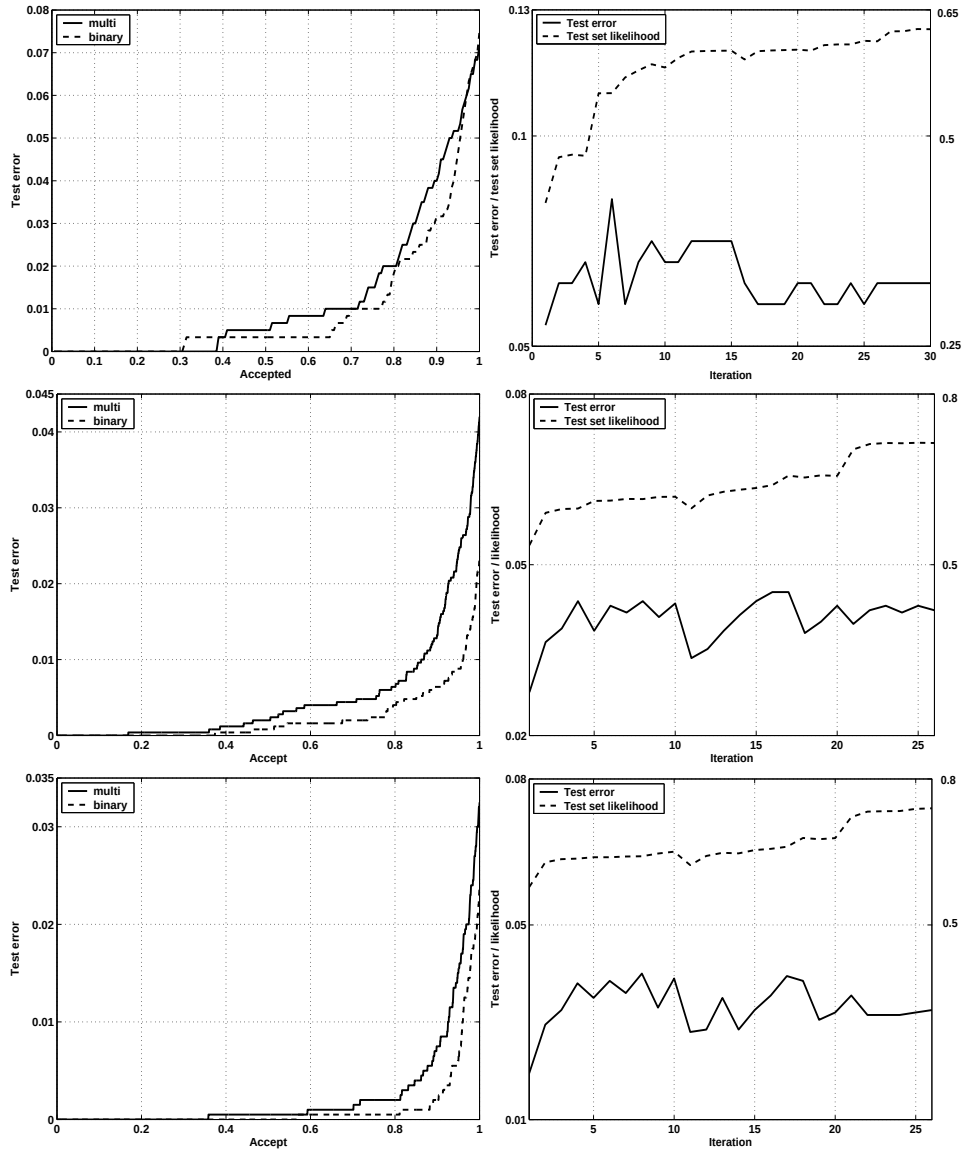

Figure 1: ROC curves *binary* against *multi* (left), learning curve (test error, test set likelihood) for *multi* (right). Upper: *train5*; middle: *train8*; lower: *train9*.

is not the solution to a well-understood convex problem and there is not much theory to assess convergence or progress. However, we obtain valid predictive probability estimates and can adjust a large number of hyperparameters by optimization. Our method operates jointly on all data and does not require imposing artificial binary partitions or combining binary discriminants using posthoc heuristics. The linear scaling in the number of training points is guaranteed *a priori*. Previous work on Bayesian GP multi-class models include [9, 2], but we are not aware of previous sparse approximations.

In future work we would like to apply our implementation to larger tasks and differ-

ent C-process models and compare our method to other multi-class kernel methods. The idea to use sparse $\boldsymbol{\pi}_i$ vectors mentioned in Section 5 will be explored. Recently there has been much interest in kernel generalizations of predictive sequence models. We can apply our framework as central building block to obtain a Bayesian GP version of a conditional random field. While inference in this setting consists simply of parallel runs of our scheme here, learning hyperparameters is more challenging and involves dynamic programming. Details and experiments will be given in a forthcoming paper. While kernel methods and probabilistic graphical models have largely been treated as different entities, we hope our contribution can be used to bridge this gap.

References

[1] P. Davis and P. Rabinovitz. *Methods of Numerical Integration*. Academic Press, 1984.

[2] Mark N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

[3] Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616. MIT Press, 2002. See `www.cs.berkeley.edu/~mseeger`.

[4] Thomas Minka. Expectation propagation for approximate Bayesian inference. In J. Breese and D. Koller, editors, *Uncertainty in Artificial Intelligence 17*. Morgan Kaufmann, 2001.

[5] Manfred Opper and Ole Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

[6] M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, July 2003. See `www.cs.berkeley.edu/~mseeger`.

[7] M. Seeger and M. I. Jordan. Sparse Gaussian process classification with multiple classes. Technical Report 661, Department of Statistics, University of California at Berkeley, 2004. See `www.cs.berkeley.edu/~mseeger`.

[8] Matthias Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2):69–106, 2004.

[9] Christopher K. I. Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.