Sparse Gaussian Process Classification With Multiple Classes

Matthias Seeger
Computer Science Division
University of California at Berkeley
Soda Hall, Berkeley, CA
mseeger@cs.berkeley.edu

Michael I. Jordan
Computer Science Division and Department of Statistics
University of California at Berkeley
Soda Hall, Berkeley, CA
jordan@cs.berkeley.edu

April 27, 2004

Technical Report 661
Department of Statistics
University of California, Berkeley

Abstract

Sparse approximations to Bayesian inference for nonparametric Gaussian Process models scale linearly in the number of training points, allowing for the application of these powerful kernel-based models to large datasets. We show how to generalize the binary classification *informative vector machine (IVM)* [6] to multiple classes. In contrast to earlier efficient approaches to kernel-based non-binary classification, our method is a principled approximation to Bayesian inference which yields valid uncertainty estimates and allows for hyperparameter estimation via marginal likelihood maximization. While most earlier proposals suggest fitting independent binary discriminants to heuristically chosen partitions of the data and combining these in a heuristic manner, our method 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)* [6, 10] is a sparse approximation to Bayesian inference for binary *Gaussian process (GP)* classification models which combines *assumed density filtering (ADF)* (or *Bayesian on-line*) projection updates with greedy forward selection of an *active subset* of the training sample using information-theoretic criteria from *active learning*. In this paper we show how this framework can be extended to GP models for C classes.

Our approach can be compared to multi-class extensions of the binary *support vector machine (SVM)* classifier which is not based on a probabilistic model, but rather employs a maximum margin discriminant. Most of these extensions attempt a posthoc combination of a sequence of binary maximum margin discriminants trained on different *binary* splits of the training sample consistent with the targets. The advantage of these approaches is that optimized software for the binary case can be used directly. However, the binary partitions ("output codings") and the posthoc combination scheme have to be chosen in a heuristic and rather arbitrary way. Even if a good "code" is given, the separate training of the discriminants is suboptimal, because pattern provide *joint* information about all classes, which couples the discriminants in general. We note that both Lee *et.al.* [7] and Weston and Watkins [13] use C independent discriminants jointly like we do here, however generalizing the concept of margin to C classes in different ways. In fact, there is no canonical (or "optimal") generalization and different choices can lead to very different behaviour statistically although this is far from obvious. For example, while the margin generalization of Weston and Watkins seems more directly related to the binary setup, Lee *et.al.* show that it does not in general lead to a universally consistent method because the minimizer of the true expected loss (no RKHS restriction) does not give the same classification as the Bayes optimal rule for some data distributions. The generalization of Lee *et.al.* has this consistency property, however it is certainly possible to come up with different consistent generalizations which will all behave quite differently for a finite sample size (and $C > 2$). With these SVM extensions, it is even less clear how valid estimates of predictive class probabilities can be obtained and how free hyperparameters should be chosen.[1]

Our method addresses all of these problems by operating jointly on the data for all classes without requiring artificial "codes". Being a direct approximation to Bayesian inference, predictive probabilities can be estimated and hyperparameters can be adjusted by empirical Bayesian techniques. As is often the case with Bayesian methods, the main challenge is to find an *efficient* inference methodology. If n is the training set size, C the number of classes, and $d \ll n$ the (adjustable) "active set" size, our scheme requires $O(n\,C\,d)$ memory and $O(n\,C\,d^2)$ time to compute a representation of size $O(C\,d^3)$ from which predictive probabilities can be computed in $O(C\,d^2)$. This scaling is *linear* in both training set size and number of classes, which allows for applications to large real-world problems.[2] The case of $C > 2$ classes is significantly more difficult to handle than the binary one, and we have to face a number of new representational and algorithmic challenges.

The structure of the paper is as follows. In Section 2 the GP multi-class model is introduced. Our basic idea is motivated in Section 3. The posterior representation is developed in Section 4, and we show how to update it after an inclusion in Section 5. The problem of constrained ADF projection is treated in Section 6. We introduce our criterion for forward selection in Section 7. The simple myopic scheme is not powerful enough, and crucial extensions are covered in Section 8 where our final algorithm is specified. Results of some

[1] Traditional methods such as cross-validation are not useful, because different covariance functions (kernels) should be used for every class leading to at least $O(C)$ hyperparameters.

[2] In the light of a confused anonymous referee, it seems necessary to clarify our scaling statements. Under the assumption that $C < d \ll n$ the *dominant* contribution to the scaling is $O(n\,C\,d^2)$. As with any other method in this domain, there are additional $O(d^3)$, $O(d\,C^3)$ and other contributions which are *subdominant* under these assumptions. Especially, our method cannot be used in a large C domain without further modifications not discussed here. Unless otherwise said, claims about the scaling behaviour concentrate on the dominant term (under these assumptions) which is linear in the training set size n.

preliminary experiments are given in Section 10. We conclude in Section 11 with suggestions for further work.

Note that the scheme to be described here requires a number of novel concepts together with a very elaborate representation, and many complicated details have to be stated to allow a full and self-contained understanding. Our approach is to begin each section with a short high-level description, followed by all the details we feel are necessary for full understanding.

2 Multi-Class Gaussian Process Classification

In this section we briefly introduce GP models for C-class classification. Denote a case by $(\boldsymbol{x}, y)$, $\boldsymbol{x} \in \mathcal{X}, y \in \{1, \dots, C\}$. $\boldsymbol{x}$ is called input point (or pattern), y target (or label), distributed according to an unknown *data distribution* $P(\boldsymbol{x}, y)$. Our goal is to predict y or even $P(y|\boldsymbol{x})$ at points $\boldsymbol{x}$ of interest, thus inference *conditional* on typical input points.[3] Functions into a finite set are hard to parameterize directly. In the *binary* case $C = 2$, a standard way of constructing a model is to introduce a latent variable $u \in \mathbb{R}$ to represent the (unobserved) log odds ratio $\log(P(y = 1|\boldsymbol{x})/P(y = 2|\boldsymbol{x})) - \beta$, thus

$$P(y = 1|u) = \sigma(u + \beta) = \frac{1}{1 + e^{-(u+\beta)}}$$

($\beta \in \mathbb{R}$ is a *intercept* parameter which can be interpreted as prior for the log ratio). Note that $P(y|u)$ is a binomial distribution with (unconstrained) natural parameter u. In a GP model, $\boldsymbol{x} \mapsto u$ is given a *Gaussian process prior* with zero mean and covariance function $K(\boldsymbol{x}, \boldsymbol{x}')$. For an introduction to GPs in the context of machine learning applications, see [12]. In the context of this paper, a (zero-mean) GP is simply a consistent way of assigning covariance matrices $\boldsymbol{K}(X) \in \mathbb{R}^{m,m}$ (or zero-mean Gaussian distributions) to (ordered) point sets $X \in \mathcal{X}^m$ by means of the covariance function K: $\boldsymbol{K} = (K(\boldsymbol{x}, \boldsymbol{x}'))_{\boldsymbol{x},\boldsymbol{x}' \in X}$. Note that the covariance function may have free parameters, and one of the appealing aspects of the Bayesian framework is that such *hyperparameters* can be adjusted in an automatic fashion (via *marginal likelihood maximization*). The generalization to C classes is straightforward: let

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

be the multinomial distibution with natural parameters $\boldsymbol{v} \in \mathbb{R}^C$ (here, $\boldsymbol{1} = (1)_c$). $\boldsymbol{\beta} = (\beta_c)_c$ is a vector of intercept parameters. Given C covariance functions $K^{(c)}$, we employ *independent* zero-mean GP priors with kernels $K^{(c)}$ on the component functions $u^{(c)} : \mathcal{X} \to u^{(c)}$. Note that our parameterization of the multinomial $P(y|\boldsymbol{u})$ is *overcomplete* in the sense that $P(y|\boldsymbol{u}) = P(y|\boldsymbol{u} + \alpha\boldsymbol{1})$ for all y, $\alpha \in \mathbb{R}$ which will create some minor complications downstream. However, the usual procedure of pegging one of the $\boldsymbol{u}$ components to a fixed value in combination with the independence of the $u^{(c)}$ priors leads to a prior imbalance in the class probabilities which is (in general) not motivated by the task, while the symmetric overcomplete parameterization renders uninformative class priors (for $\boldsymbol{\beta} \propto \boldsymbol{1}$). The mapping $\boldsymbol{u} \to (P(y|\boldsymbol{u}))_y$ is called (overcomplete) *softmax* function. It is important to note that the negative logarithm of each component function is *convex* and strictly so on affine spaces orthogonal to $\boldsymbol{1}$.

[3] We are not interested in modelling $P(\boldsymbol{x})$, and as a consequence our model cannot be used to infer any useful properties of this marginal distribution.

Given a training sample $D = \{(\boldsymbol{x}_i, y_i) \,|\, i = 1, \dots, n\}$ drawn independently and identically distributed (i.i.d.) from the data distribution, the goal of *first level inference* is to obtain approximations to *predictive distributions* $P(y_*|\boldsymbol{x}_*, D)$ at test points $\boldsymbol{x}_* \in \mathcal{X}$. Conditioned on hyperparameters, these can be obtained from an (approximate) representation of the posterior $P(\boldsymbol{u}|D)$ where $\boldsymbol{u} = (u_i^{(c)})_{i,c} \in \mathbb{R}^{nC}$ $i = 1, \dots, n$, $c = 1, \dots, C$ with $u_i^{(c)} = u^{(c)}(\boldsymbol{x}_i)$. Note that the processes $u^{(c)}$ are coupled *a posteriori* due to the coupling in the likelihood. A wide range of approximate GP methods represent this posterior by a Gaussian, as we will do here. Note that the true posterior is log-concave (a consequence of the log-concavity of the likelihood) with tails dominated by the Gaussian prior, so the Gaussian approximation can be well-motivated.

3 The Basic Idea

A straightforward extension of the binary IVM [6] is possible, but would scale at least as $O(n\, C^2\, d^2)$. The problem is that the inverse posterior covariance matrix is the sum of two matrices (the kernel matrix from the GP prior and a matrix coming from the likelihood approximation) which are principally block-diagonal, but w.r.t. *different* groupings of the latent variables. While both matrices have exploitable structure, the sum does not, which leads to the unfavourable scaling. We give a principled method for finding an approximation to the blocks in the likelihood approximation matrix which leads to an efficient representation by making use of the block-diagonal structure of the kernel matrix.

We make use of a matrix and vector notation which may be unfamiliar to the reader, but is essential to manage the involved representation developed below. Note that our notation is fairly standard.[4] Vectors and matrices are set in bold face. Subset indexing is defined as $\boldsymbol{A}_{I,J} := (\alpha_{i,j})_{i \in I, j \in J}$, I, J ordered index sets. $\cdot$ denotes the full index, i the singleton $\{i\}$, and $\boldsymbol{A}_I := \boldsymbol{A}_{I,I}$. $\boldsymbol{I}$ denotes the identity matrix, $\boldsymbol{I} = (\delta_{i,j})_{i,j}$, its columns are denoted as $\boldsymbol{\delta}_i = \boldsymbol{I}_{\cdot,i}$. The vector of all ones is $\boldsymbol{1} = (1)_i$. Note that $\boldsymbol{A}_{I,J} = \boldsymbol{I}_{I,\cdot} \boldsymbol{A} \boldsymbol{I}_{\cdot,J}$, we will make frequent use of $\boldsymbol{I}_{I,\cdot}$ as "selection operator" and $\boldsymbol{I}_{\cdot,I}$ as "distribution operator".

The subset indexing notation may be familiar from our work on the binary IVM [10, 6], but in the multi-class context (and more general in C-process models, see [12], Sect. 3.2) an additional complexity arises: in both vectors and matrices, we need indexing w.r.t. datapoints $i \in \{1, \dots, n\}$ *and* w.r.t. classes $c \in \{1, \dots, C\}$. We write $\boldsymbol{u} = (\boldsymbol{u}_i)_i = (\boldsymbol{u}^{(c)})_c = (u_i^{(c)})_{i,c} \in \mathbb{R}^{nC}$, $\boldsymbol{u}_i \in \mathbb{R}^C$, $\boldsymbol{u}^{(c)} \in \mathbb{R}^n$. By "inner grouping" over c (resp. i) we mean that the index c (resp. i) changes faster. For example, $\boldsymbol{u} = (u_1^{(1)}, \dots, u_n^{(1)}, \dots, u_1^{(C)}, \dots, u_n^{(C)})$ uses inner grouping over i and outer grouping over c. Crucially, we will need *both* groupings in this paper.[5] Let the permutation matrix $\hat{\boldsymbol{P}}_{\leftrightarrow}$ convert between the groupings, i.e. $\hat{\boldsymbol{P}}_{\leftrightarrow}(\boldsymbol{u}^{(c)})_c = (\boldsymbol{u}_i)_i$ (the r.h.s. vector has inner grouping over c). For conciseness it is essential to "overload" our subscript notation to these groupings. For inner grouping over c we have $\boldsymbol{I}_{I,J} := \boldsymbol{I}_{I,J} \otimes \boldsymbol{I}$, $I, J \subset \{1, \dots, n\}$, where the left hand matrix is $\in \mathbb{R}^{C|I|,C|J|}$ and $\boldsymbol{I}_{I,J} \in \mathbb{R}^{|I|,|J|}$, $\boldsymbol{I} \in \mathbb{R}^{C,C}$ on the right hand side. For outer grouping over c we have $\boldsymbol{I}_{I,J} := \hat{\boldsymbol{P}}_{\leftrightarrow}^T(\boldsymbol{I}_{I,J} \otimes \boldsymbol{I})\hat{\boldsymbol{P}}_{\leftrightarrow}$. Here, $(\alpha_{i,j})_{i,j} \otimes \boldsymbol{B} := (\alpha_{i,j}\boldsymbol{B})_{i,j}$ is the *Kronecker product*.

[4] See for example [5], Sect. 0.7 where $\boldsymbol{A}_{I,J}$ is denoted as $\boldsymbol{A}(I,J)$.

[5] It is easiest to view matrices and vectors with inner grouping over c as "blocked objects", i.e. n-dimensional matrices and vectors with elements in $\mathbb{R}^{C,C}$ (inner grouping w.r.t. i accordingly).

For vectors $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{x} \succ \boldsymbol{y}$ ($\boldsymbol{x} \succeq \boldsymbol{y}$ resp.) means that $x_i > y_i$ ($x_i \geq y_i$) for all i. For matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{A} \succ \boldsymbol{B}$ ($\boldsymbol{A} \succeq \boldsymbol{B}$ resp.) means that $\boldsymbol{A} - \boldsymbol{B}$ is positive definite (positive semidefinite). See [1] for a detailed account of such generalized inequalities.

When trying to generalize the IVM to C classes, we run into the following problem. Recall from [6] that the IVM approximation amounts to replacing the likelihood factors (sites) by Gaussian *site approximations* with precision matrix $\boldsymbol{\Pi}_i$. We use the notation $N^U(\boldsymbol{u}_i|\boldsymbol{b}_i, \boldsymbol{\Pi}_i)$ for these factors, meaning an unnormalized Gaussian with precision (inverse covariance) matrix $\boldsymbol{\Pi}_i$ and a mean $\boldsymbol{v}$ satisfying $\boldsymbol{\Pi}_i \boldsymbol{v} = \boldsymbol{b}_i$. $\boldsymbol{b}_i$, $\boldsymbol{\Pi}_i$ are called *site parameters*, and we generally assume that $\boldsymbol{\Pi}_i \succeq \boldsymbol{0}$ (although it might be singular).[6] For a *sparse* approximation with current *active set* $I \subset \{1, \dots, n\}$, $d = |I|$, we have $\boldsymbol{\Pi}_i = \boldsymbol{0}$ for $i \notin I$. Thus, the covariance matrix of the Gaussian posterior approximation is

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

where $\boldsymbol{K} = \operatorname{diag}(\boldsymbol{K}^{(c)})_c$ (outer grouping over c) due to the independence of the priors, and $\boldsymbol{\Pi} = \operatorname{diag}(\boldsymbol{\Pi}_i)_i$ (inner grouping over c) due to the (conditional) independence of the datapoints. But the block-diagonal structure is revealed under *different* groupings only, and $\boldsymbol{A}$ does not have a simple structure. Therefore, a straightforward extension of IVM to C classes scales at least *quadratically* in $d\,C$ which is not acceptable.

An idea to make progress would be to constrain all $\boldsymbol{\Pi}_i$ to be diagonal. This is equivalent to assuming posterior independence of the processes $u^{(c)}$, $c = 1, \dots, C$. We could then use a variational mean field approximation to determine coefficients of $\boldsymbol{\Pi}$ and other site parameters. While the *prior* independence of the $u^{(c)}$ seems a sensible assumption (and is absolutely necessary to obtain a feasible scheme), we think that a posterior independence assumption is too strong in that there is no direct mechanism for the approximation scheme to represent the coupling between the $u^{(c)}$ introduced by the likelihood factors. It can either disregard the couplings or represent them indirectly via a "distorted" choice[7] of the only $O(Cn)$ variational parameters. Interestingly, it is possible to represent the likelihood couplings exactly up to second order and end up with a scheme which is essentially of the *same* complexity as the factorized mean field scheme. Thus, while an approximation based on a diagonal $\boldsymbol{\Pi}$ may be simpler to develop or implement, it will not be (significantly) more efficient than our scheme and is of no further interest to us here.

Our principal idea is to exploit an analogy to a different way of approximating the posterior $P(\boldsymbol{u}|D)$, namely by a *Laplace approximation* [14]. There, the concave log posterior is expanded to second order at its mode $\hat{\boldsymbol{u}}$, giving rise to a Gaussian distribution with a covariance of the form (1), but with $\boldsymbol{\Pi}_i = -\nabla\nabla_{\boldsymbol{u}_i} \log P(y_i|\hat{\boldsymbol{u}}_i)$. In other words, the non-Gaussian log likelihood terms are replaced by their second order expansions. Crucially, these "precision blocks"[8] are of a simple form: $\boldsymbol{\Pi}_i = \operatorname{diag} \boldsymbol{p}_i - \boldsymbol{p}_i \boldsymbol{p}_i^T$, $\boldsymbol{p}_i = (P(y|\hat{\boldsymbol{u}}_i))_y$. Note that this form captures the coupling due to the likelihood factor for the i-th datapoint up to second order. This constraint on $\boldsymbol{\Pi}_i$ can be exploited to run the Laplace approximation scheme in time and memory linear in C. However, this scheme is not a sparse approximation and scales cubically in n, furthermore there is some evidence that GP approximations based on

[6] In the binary IVM, $\boldsymbol{\Pi}_i$ is a nonnegative scalar.

[7] In the sense that there are no parameters for a coupling, so the ones of the decoupled representation have to make up for it somehow. The same problem arises in variational mean field approximations.

[8] A *precision matrix* is the inverse of a covariance matrix.

ADF projections (or *expectation propagation (EP)* projections [8], aka. ADATAP [9]) outperform the Laplace approximation. Our scheme tries to combine the best of both worlds: we propose to use ADF/EP projections onto a family of Gaussian site approximations with *constrained* precision blocks $\boldsymbol{\Pi}_i$ which must have the form

$$\boldsymbol{\Pi}_i = \operatorname{diag} \boldsymbol{\pi}_i - \alpha_i^{-1} \boldsymbol{\pi}_i \boldsymbol{\pi}_i^T, \quad \alpha_i = \mathbf{1}^T \boldsymbol{\pi}_i, \ \boldsymbol{\pi}_i \succ \mathbf{0}. \tag{2}$$

Note that $\boldsymbol{\Pi}_i$ has a single eigenvalue 0 with eigenvector $\mathbf{1}$ and is positive definite on the orthogonal complement of $\mathbf{1}$.

In the remainder of this paper we address the issues of turning this idea into an efficient algorithm. The main challenges are finding a representation of the posterior which can be stored and updated in $O(C)$ and solving the constrained ADF projection problem. Furthermore, it turns out that the simple myopic "on-line"approach used for the binary IVM is not sufficient for the $C > 2$ case, and we describe extensions involving joint EP iterations over a subset of the active set.

4 The Representation

In this section, we develop the representation which is used to approximate the posterior over $\boldsymbol{u} \in \mathbb{R}^{nC}$ and allows to compute predictive probabilities for test points. The key is to exploit the block-diagonal structure of the prior covariance matrix $\boldsymbol{K}$ together with the restricted form (2) for the precision blocks $\boldsymbol{\Pi}_i$ as motivated in Section 3. The presentation is technical and offers no useful intuition, it can be skipped by readers not interested in details.

It turns out to be crucial to exploit the block-diagonal structure of $\boldsymbol{K}$, thus in the remainder of this section we work with outer grouping over c, the site precision matrix is $\boldsymbol{\Pi} = \hat{\boldsymbol{P}}_{\leftrightarrow}^T \operatorname{diag}(\boldsymbol{\Pi}_i)_i \hat{\boldsymbol{P}}_{\leftrightarrow}$. The posterior covariance matrix $\boldsymbol{A}$ from (1) can be written as

$$\boldsymbol{A} = \boldsymbol{K} - \boldsymbol{K}\boldsymbol{\Phi}\boldsymbol{K}, \quad \boldsymbol{\Phi} = (\boldsymbol{I} + \boldsymbol{\Pi}\boldsymbol{K})^{-1}\boldsymbol{\Pi}.$$

Denote $\boldsymbol{R} = \mathbf{1}_C \otimes \boldsymbol{I} \in \mathbb{R}^{Cd,d}$, i.e. a vertical stack of C $\boldsymbol{I} \in \mathbb{R}^{d,d}$ matrices. Note that

$$\boldsymbol{R}^T \boldsymbol{v} = \sum_c \boldsymbol{r}^{(c)}, \quad \boldsymbol{R}^T \operatorname{diag}(\boldsymbol{B}^{(c)})_c \boldsymbol{R} = \sum_c \boldsymbol{B}^{(c)},$$

so $\boldsymbol{R}^T$ acts as "summation operator". Also, let $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{D}^{(c)})_c$ with $\boldsymbol{D}^{(c)} = \operatorname{diag} \boldsymbol{\pi}^{(c)} = \operatorname{diag}(pi_i^{(c)})_i \in \mathbb{R}^{n,n}$, thus $\boldsymbol{D}_I = \operatorname{diag}(\boldsymbol{D}_I^{(c)})_c \in \mathbb{R}^{Cd,Cd}$. Furthermore, $\boldsymbol{\Gamma} = \operatorname{diag}(\alpha_i)_i \in \mathbb{R}^{n,n}$.[9] Note that $\boldsymbol{R}^T \boldsymbol{D}_I \boldsymbol{R} = \boldsymbol{\Gamma}_I$.

The intuition behind our representation is that $\boldsymbol{\Pi}$ is the difference of a diagonal matrix and a matrix of rank d (this is easy to see in outer grouping over i, thus must hold as well in outer grouping over c). If $\boldsymbol{\Pi}$ was just diagonal we could use more or less the same representation as for the binary IVM (see [10] for details), which motivates the definition of $\boldsymbol{E}$ below. Dealing with the additional rank d introduces some complications but importantly does not worsen the scaling behaviour. We have

$$\begin{aligned}\boldsymbol{\Pi} &= \hat{\boldsymbol{P}}_{\leftrightarrow}^T \operatorname{diag}(\boldsymbol{\Pi}_i)_i \hat{\boldsymbol{P}}_{\leftrightarrow} = \boldsymbol{I}_{\cdot,I} \left(\boldsymbol{D}_I - \boldsymbol{D}_I \boldsymbol{R} \boldsymbol{\Gamma}_I^{-1} \boldsymbol{R}^T \boldsymbol{D}_I \right) \boldsymbol{I}_{I,\cdot} \\ &= \left(\delta_{c,c'} \boldsymbol{D}_I^{(c)} - \boldsymbol{D}_I^{(c)} \boldsymbol{\Gamma}_I^{-1} \boldsymbol{D}_I^{(c')} \right)_{c,c'}.\end{aligned}$$

[9] We set $\alpha_i = 0$ for $i \notin I$.

Define

$$\begin{aligned}
\boldsymbol{E} &= \boldsymbol{I} + \boldsymbol{D}_I^{1/2}\boldsymbol{K}_I\boldsymbol{D}_I^{1/2} = \operatorname{diag}(\boldsymbol{E}^{(c)})_c,\\
\boldsymbol{P} &= \boldsymbol{D}_I^{1/2}\boldsymbol{E}^{-1}\boldsymbol{D}_I^{1/2} = \operatorname{diag}(\boldsymbol{P}^{(c)})_c,\\
\boldsymbol{H} &= \boldsymbol{R}^T\boldsymbol{P}\boldsymbol{R} = \sum_c \boldsymbol{P}^{(c)}.
\end{aligned}$$

Note how the block-diagonal $O(C)$ structure of the kernel matrix $\boldsymbol{K}$ is inherited by $\boldsymbol{E}$, $\boldsymbol{P}$, they can be stored in $O(C)$. We show that

$$\boldsymbol{\Phi} = (\boldsymbol{I} + \boldsymbol{\Pi}\boldsymbol{K})^{-1}\boldsymbol{\Pi} = \boldsymbol{I}_{\cdot,I}\left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{R}\boldsymbol{H}^{-1}\boldsymbol{R}^T\boldsymbol{P}\right)\boldsymbol{I}_{I,\cdot}.$$

Namely,

$$\begin{aligned}
&\boldsymbol{\Pi}\boldsymbol{K}\boldsymbol{I}_{\cdot,I}\boldsymbol{P} = \boldsymbol{I}_{\cdot,I}\left(\boldsymbol{I} - \boldsymbol{D}_I\boldsymbol{R}\boldsymbol{\Gamma}_I^{-1}\boldsymbol{R}^T\right)\boldsymbol{D}_I\boldsymbol{K}_I\boldsymbol{D}_I^{1/2}\boldsymbol{E}^{-1}\boldsymbol{D}_I^{1/2}\\
&= \boldsymbol{I}_{\cdot,I}(\ldots)\boldsymbol{D}_I^{1/2}(\boldsymbol{E} - \boldsymbol{I})\boldsymbol{E}^{-1}\boldsymbol{D}_I^{1/2} = \boldsymbol{I}_{\cdot,I}(\ldots)\left(\boldsymbol{D}_I - \boldsymbol{P}\right).
\end{aligned}$$

If $\boldsymbol{F} = \boldsymbol{I} - \boldsymbol{R}\boldsymbol{H}^{-1}\boldsymbol{R}^T\boldsymbol{P}$, then

$$\boldsymbol{\Pi}\boldsymbol{K}\boldsymbol{I}_{\cdot,I}\boldsymbol{P}\boldsymbol{F}\boldsymbol{I}_{I,\cdot} = \boldsymbol{I}_{\cdot,I}\left(\boldsymbol{I} - \boldsymbol{D}_I\boldsymbol{R}\boldsymbol{\Gamma}_I^{-1}\boldsymbol{R}^T\right)\left(\boldsymbol{D}_I - \boldsymbol{P}\right)\boldsymbol{F}\boldsymbol{I}_{I,\cdot} = -\boldsymbol{I}_{\cdot,I}\boldsymbol{P}\boldsymbol{F}\boldsymbol{I}_{I,\cdot} + \boldsymbol{\Pi},$$

where we have used $\boldsymbol{R}^T\boldsymbol{D}_I\boldsymbol{R} = \boldsymbol{\Gamma}_I$, $\boldsymbol{R}^T\boldsymbol{P}\boldsymbol{R} = \boldsymbol{H}$. This proves the claim.

We choose the following posterior representation which makes use of the block-diagonal structure of $\boldsymbol{E}$, $\boldsymbol{P}$.

$$\begin{aligned}
\boldsymbol{E}^{(c)} &= \boldsymbol{L}^{(c)}\boldsymbol{L}^{(c)T},\ \boldsymbol{L}^{(c)} \text{ lower triangular},\\
\boldsymbol{P}^{(c)} &= \boldsymbol{B}^{(c)T}\boldsymbol{B}^{(c)},\ \boldsymbol{B}^{(c)} = \boldsymbol{L}^{(c)-1}\boldsymbol{D}_I^{(c)1/2} \text{ lower triangular},\\
\boldsymbol{H} &= \boldsymbol{L}\boldsymbol{L}^T,\ \boldsymbol{L} \text{ lower triangular}.
\end{aligned} \tag{3}$$

For a test point $\boldsymbol{x}_*$, the predictive covariance for $\boldsymbol{u}_* \in \mathbb{R}^C$ is given by

$$\boldsymbol{A}_* = \boldsymbol{K}_* - \boldsymbol{K}_{I,*}^T\boldsymbol{\Phi}_I\boldsymbol{K}_{I,*}$$

with $\boldsymbol{K}_* = \operatorname{diag}(K^{(c)}(\boldsymbol{x}_*, \boldsymbol{x}_*))_c \in \mathbb{R}^{C,C}$ and $\boldsymbol{K}_{I,*} = \operatorname{diag}((K^{(c)}(\boldsymbol{x}_i, \boldsymbol{x}_*))_{i\in I})_c \in \mathbb{R}^{Cd,C}$. Let $\boldsymbol{m}_*^{(c)} = \boldsymbol{B}^{(c)}\boldsymbol{K}_{I,*}^{(c)}$, $\boldsymbol{q}_*^{(c)} = \boldsymbol{L}^{-1}\boldsymbol{B}^{(c)T}\boldsymbol{m}_*^{(c)}$. Then,

$$\boldsymbol{A}_* = \operatorname{diag}\left(\boldsymbol{K}_*^{(c)} - \|\boldsymbol{m}_*^{(c)}\|^2\right)_c + \boldsymbol{Q}_*^T\boldsymbol{Q}_*, \quad \boldsymbol{Q}_* = \left(\boldsymbol{q}_*^{(1)} \ldots \boldsymbol{q}_*^{(C)}\right) \tag{4}$$

which is $O(C^2\, d)$, while $\boldsymbol{m}_*^{(c)}$, $\boldsymbol{q}_*^{(c)}$ cost $O(C\, d^2)$ given the representation (more detailed running time and memory requirement details are given below). We define the *stub vectors*

$$\boldsymbol{m}_j^{(c)} = \boldsymbol{B}^{(c)}\boldsymbol{K}_{I,j}^{(c)}, \quad \boldsymbol{q}_j^{(c)} = \boldsymbol{L}^{-1}\boldsymbol{B}^{(c)T}\boldsymbol{m}_j^{(c)}, \quad j = 1, \ldots, n \tag{5}$$

which are required to compute marginal means and covariances at training points in order to drive the active set selection. It is tempting to store only one kind of stubs and compute the other on demand, but the computation on demand costs $O(C\, d^2)$ for each stub which would drive the overall cost to $O(n\, C\, d^3)$. One of the most important features of the representation described here is that the stub vectors for a large number of points j can be updated efficiently when the active set I is expanded. Apart from that, the particular form of the representation is motivated by numerical stability issues, for example the matrices $\boldsymbol{E}^{(c)}$ are positive definite with all eigenvalues ≥ 1, thus are very well-conditioned.

Recall from [6, 10] that the posterior approximation is

$$Q(\boldsymbol{u}) = N(\boldsymbol{u}|\boldsymbol{h}, \boldsymbol{A}), \quad \boldsymbol{h} = \boldsymbol{A}\boldsymbol{b},$$

where $\boldsymbol{b} = (\boldsymbol{b}_i)_i$ are site parameters ($\boldsymbol{b}_i = \boldsymbol{0}$ for $i \notin I$) and the covariance matrix $\boldsymbol{A}$ is given by (1). The computation of $\boldsymbol{h}$ (and of predictive means in general) is complicated by the fact that $\boldsymbol{\Pi}_I$ does not have full rank dC, but rather $d(C-1)$ due to the overcomplete parameterization of the multinomial noise model. It is easy to see that

$$\begin{aligned} \operatorname{ran} \boldsymbol{\Pi}_I &= \left\{ \boldsymbol{u} \in \mathbb{R}^{Cd} \,\middle|\, \boldsymbol{R}^T \boldsymbol{u} = \sum_c \boldsymbol{u}^{(c)} = \boldsymbol{0} \right\}, \\ \ker \boldsymbol{\Pi}_I &= \left\{ \boldsymbol{u} \in \mathbb{R}^{Cd} \,\middle|\, \boldsymbol{u}^{(c)} = \boldsymbol{u}^{(c')} \text{ for all } c, c' \right\}. \end{aligned}$$

If $\boldsymbol{s} \in \operatorname{ran} \boldsymbol{\Pi}$, then $\boldsymbol{\Pi}_I \boldsymbol{D}_I^{-1} \boldsymbol{s}_I = \boldsymbol{s}_I$ because $\boldsymbol{\Pi}_I = (\boldsymbol{I} - \boldsymbol{D}_I \boldsymbol{R} \boldsymbol{\Gamma}_I^{-1} \boldsymbol{R}^T) \boldsymbol{D}_I$ and $\boldsymbol{R}^T \boldsymbol{s}_I = \boldsymbol{0}$. Thus, if $\boldsymbol{b}_I = \boldsymbol{w} + (\boldsymbol{v})_c$ with $\boldsymbol{w} = (\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^T)\boldsymbol{b}_I \in \operatorname{ran} \boldsymbol{\Pi}_I$ and $\boldsymbol{v} = \boldsymbol{R}^T \boldsymbol{b}_I \in \mathbb{R}^d$ (i.e. $(\boldsymbol{v})_c \in \ker \boldsymbol{\Pi}_I$), then $\boldsymbol{b}_I = \boldsymbol{\Pi}_I \boldsymbol{D}_I^{-1} \boldsymbol{w} + (\boldsymbol{v})_c$. Since

$$\boldsymbol{A} = \boldsymbol{K} - \boldsymbol{K}\boldsymbol{\Phi}\boldsymbol{K} = \boldsymbol{K}\left(\boldsymbol{I} + \boldsymbol{\Pi}\boldsymbol{K}\right)^{-1}, \quad \boldsymbol{b} = \boldsymbol{I}_{\cdot,I} \boldsymbol{b}_I, \quad \boldsymbol{\Pi} = \boldsymbol{I}_{\cdot,I} \boldsymbol{\Pi}_I \boldsymbol{I}_{I,\cdot},$$

we see that

$$\boldsymbol{h} = \boldsymbol{A}\boldsymbol{b} = \boldsymbol{K}\boldsymbol{\Phi}\boldsymbol{I}_{\cdot,I} \boldsymbol{D}_I^{-1} \boldsymbol{w} + \left(\boldsymbol{K}_{\cdot,I}^{(c)} \boldsymbol{v}\right)_c - \boldsymbol{K}\boldsymbol{\Phi}\boldsymbol{K}\left(\boldsymbol{I}_{\cdot,I} \boldsymbol{v}\right)_c.$$

Define

$$\begin{aligned} \boldsymbol{\beta}^{(1,c)} &= \boldsymbol{L}^{(c)-1} \boldsymbol{D}_I^{(c)-1/2} \boldsymbol{w}^{(c)}, \\ \boldsymbol{w}^{(c)} &= \boldsymbol{b}_I^{(c)} - \boldsymbol{v}, \ \boldsymbol{v} = \sum_c \boldsymbol{b}_I^{(c)}, \\ \boldsymbol{\beta}^{(2)} &= \boldsymbol{L}^{-1} \sum_c \boldsymbol{B}^{(c)T} \boldsymbol{\beta}^{(1,c)}, \\ \boldsymbol{\beta}^{(3,c)} &= \boldsymbol{B}^{(c)} \boldsymbol{K}_I^{(c)} \boldsymbol{v} = \sum_{k=1}^d v_k \boldsymbol{m}_{I_k}^{(c)}, \\ \boldsymbol{\beta}^{(4)} &= \boldsymbol{L}^{-1} \sum_c \boldsymbol{B}^{(c)T} \boldsymbol{\beta}^{(3,c)} = \sum_{k=1}^d v_k \sum_c \boldsymbol{q}_{I_k}^{(c)}. \end{aligned} \tag{6}$$

Then,

$$\boldsymbol{h}^{(1)} := \boldsymbol{K}_{\cdot,I}\left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{R}\boldsymbol{H}^{-1}\boldsymbol{R}^T\boldsymbol{P}\right)\boldsymbol{D}_I^{-1}\boldsymbol{w} = \left(\boldsymbol{m}_i^{(c)T}\boldsymbol{\beta}^{(1,c)} - \boldsymbol{q}_i^{(c)T}\boldsymbol{\beta}^{(2)}\right)_{i,c}. \tag{7}$$

Next, recalling (4) for the predictive covariance, we have

$$\boldsymbol{h}^{(2)} := -\boldsymbol{K}\boldsymbol{\Phi}\boldsymbol{K}\left(\boldsymbol{I}_{\cdot,I}\boldsymbol{v}\right)_c = \left(-\boldsymbol{m}_i^{(c)T}\boldsymbol{\beta}^{(3,c)} + \boldsymbol{q}_i^{(c)T}\boldsymbol{\beta}^{(4)}\right)_{i,c}. \tag{8}$$

Altogether,

$$\boldsymbol{h} = \boldsymbol{h}^{(1)} + \boldsymbol{h}^{(2)} + \left(\boldsymbol{K}_{\cdot,I}^{(c)}\boldsymbol{v}\right)_c.$$

Therefore, the predictive mean $\boldsymbol{h}_* \in \mathbb{R}^C$ of $Q(\boldsymbol{u}_*)$ at a test point $\boldsymbol{x}_*$ is computed as

$$\boldsymbol{h}_* = \left(\boldsymbol{m}_*^{(c)T}\left(\boldsymbol{\beta}^{(1,c)} - \boldsymbol{\beta}^{(3,c)}\right) - \boldsymbol{q}_*^{(c)T}\left(\boldsymbol{\beta}^{(2)} - \boldsymbol{\beta}^{(4)}\right) + \boldsymbol{K}_{*,I}^{(c)}\boldsymbol{v}\right)_c$$

where the test stub vectors $\boldsymbol{m}_*,\ \boldsymbol{q}_*$ are defined prior to (4). The predictive covariance $\boldsymbol{A}_*$ of $Q(\boldsymbol{u}_i)$ is given by (4). Now, the (approximate) predictive distribution is given by

$$Q(y_*|\boldsymbol{x}_*, D) = \mathrm{E}_{\boldsymbol{u}_* \sim Q}\left[P(y_*|\boldsymbol{u}_*)\right].$$

Since the likelihood is not Gaussian, we have to fall back to numerical quadrature for this C-dimensional "almost-Gaussian" integral, we give some comments in Section 6 of how to proceed.

This completes the description of the belief representation which consists of $\boldsymbol{L}^{(c)}$, $\boldsymbol{B}^{(c)}$, $\boldsymbol{L}$ defined in (3), the $\boldsymbol{\beta}$ vectors defined in (6) and $\boldsymbol{K}_{\cdot,I}^{(c)}\boldsymbol{v},\ \boldsymbol{v} = \sum_c \boldsymbol{b}_I^{(c)}$. Furthermore, the stub vectors (5) are maintained for a large number of patterns j in order to drive forward selection of I.

5 Update Of The Representation After Inclusion

One of the most important features of the representation described in Section 4 is that we can update the set of all n stub vectors (5) in $O(n\,C\,d)$ for the inclusion of a new point into the active set I. The stub vectors are required to compute marginal posterior moments for the training points which will drive the forward selection to find good inclusion candidates. In this section, we show how the representation and the stub vectors can be updated in an efficient and numerically stable manner. Again, a reader not interested in details can skip this section, but may want to note our recommendations for the *chollrup* primitive.

Suppose that $i \notin I$ is to be included into the active set $I = \{I_1, \dots, I_d\}$ (thus $I_{d+1} = i$) and that its site parameters $\boldsymbol{b}_i$, $\boldsymbol{\pi}_i \in \mathbb{R}^C$ have already been determined (we show how this is done in Section 6). We need a few primitives for updating Cholesky factorizations.[10] Let $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^T \in \mathbb{R}^{d,d}$ be a Cholesky decomposition and suppose that $\boldsymbol{B}$ is extended by a last row/column $(\boldsymbol{b}^T b)^T$. The new lower triangular Cholesky factor $\boldsymbol{L}'$ is given by appending $(\boldsymbol{l}^T l)$ as bottom row to $\boldsymbol{L}$, where

$$\boldsymbol{l} = \boldsymbol{L}^{-1}\boldsymbol{b}, \quad l = \sqrt{b - \boldsymbol{l}^T\boldsymbol{l}}.$$

Denote this procedure by $(\boldsymbol{l}, l) := \mathrm{cholext}(\boldsymbol{L}, \boldsymbol{b}, b)$. If $\boldsymbol{L} \in \mathbb{R}^{d,d}$, the complexity is $O(d^2)$ for the backsubstitution[11]. Note that the procedure breaks down iff the extended matrix $\boldsymbol{B}'$ is not (numerically) positive definite.

[10] Our representation is based on Cholesky decompositions which are the most numerically stable and efficient way to deal with symmetric positive definite matrices. The primitives *cholext* and *chollrup* can be seen as numerically sound equivalents of formulas such as Sherman-Morrison-Woodbury and partitioned inverse which are known to be unstable, and their widespread use in machine learning applications is recommended. Efficient Matlab code (MEX function) for *chollrup* can be obtained from us on request.

[11] In the literature, solving linear systems with a triangular system matrix is called backsubstitution and forward substitution, depending on whether an upper or lower triangular matrix is used. We do not follow this rather confusing nomenclature, but denote both procedures as *backsubstitution*.

Next, let $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^T$ be a Cholesky decomposition and suppose that $\boldsymbol{B}' = \boldsymbol{B} + \boldsymbol{b}\boldsymbol{b}^T$. Then, $\boldsymbol{L}' = \boldsymbol{L}\tilde{\boldsymbol{L}}$ where $\tilde{\boldsymbol{L}}$ can be maintained in $O(d)$ and computed in $O(d)$ given $\boldsymbol{L}^{-1}\boldsymbol{b}$ (which itself is $O(d^2)$). Backsubstitution with $\tilde{\boldsymbol{L}}$ costs $O(d)$ only. Thus, if $\boldsymbol{X} \in \mathbb{R}^{d,m}$ with $\boldsymbol{L}\boldsymbol{X} = \boldsymbol{C}$, we can compute $\boldsymbol{X}'$ for which $\boldsymbol{L}'\boldsymbol{X}' = \boldsymbol{C}$ in $O(d\,m)$ (we do not need $\boldsymbol{C}$ for this update). We say that the columns of $\boldsymbol{X}$ are "dragged along" the update of $\boldsymbol{L}$. Denote this procedure by $(\boldsymbol{L}', \boldsymbol{X}') := \mathrm{chollrup}(\boldsymbol{L}, \boldsymbol{b}, \boldsymbol{X})$ and extend it to rank-k updates $\mathrm{chollrup}(\boldsymbol{L}, \boldsymbol{b}_1, \dots, \boldsymbol{b}_k, \boldsymbol{X})$ by concatenation. Note that *chollrup* can also be used for *negative* updates $\boldsymbol{B} - \boldsymbol{b}\boldsymbol{b}^T$ if the resulting matrix is still positive definite. While positive updates are numerically stable (if $\boldsymbol{B}$ is well-conditioned), negative updates are more suspectible to breakdown due to roundoff error.

Now, $\boldsymbol{D}_i = \operatorname{diag}\boldsymbol{\pi}_i$, i.e. $d_i^{(c)} = \pi_i^{(c)}$. We update $\boldsymbol{L}^{(c)}$ as

$$(\boldsymbol{l}_i^{(c)}, l_i^{(c)}) = \mathrm{cholext}\left(\boldsymbol{L}^{(c)}, d_i^{(c)1/2}\boldsymbol{D}_I^{(c)1/2}\boldsymbol{K}_{I,i}^{(c)}, 1 + d_i^{(c)}\boldsymbol{K}_i^{(c)}\right),$$

then $\boldsymbol{B}^{(c)}$ by adding the row $(\boldsymbol{b}_i^{(c)T} b_i^{(c)})$ with $\boldsymbol{b}_i^{(c)} = -l_i^{(c)-1}\boldsymbol{B}^{(c)T}\boldsymbol{l}_i^{(c)},\ b_i^{(c)} = d_i^{(c)1/2}/l_i^{(c)}$. The factor $\boldsymbol{L}$ for $\boldsymbol{H}$ is updated in two stages. First, $\boldsymbol{H}$ has to be replaced by $\boldsymbol{H} + \sum_c \boldsymbol{b}_i^{(c)}\boldsymbol{b}_i^{(c)T}$, then extended by the row $(\boldsymbol{q}^T q)$ with $\boldsymbol{q} = \sum_c b_i^{(c)}\boldsymbol{b}_i^{(c)},\ q = \sum_c b_i^{(c)2}$. During the first stage, we drag along the columns of $\boldsymbol{X}_Q$ (to be specified below). Thus,

$$(\boldsymbol{L}'_{1,\dots,d}, \boldsymbol{X}'_Q) = \mathrm{chollrup}\left(\boldsymbol{L}, \boldsymbol{b}_i^{(1)}, \dots, \boldsymbol{b}_i^{(C)}, \boldsymbol{X}_Q\right), \quad (\boldsymbol{l}, l) = \mathrm{cholext}\left(\boldsymbol{L}'_{1,\dots,d}, \boldsymbol{q}, q\right).$$

The stub vector $\boldsymbol{m}_j^{(c)}$ is updated by appending the component

$$m_j^{(c)} = -l_i^{(c)-1}\boldsymbol{l}_i^{(c)T}\boldsymbol{m}_j^{(c)} + b_i^{(c)}\boldsymbol{K}_{i,j}^{(c)}.$$

If $\boldsymbol{r}_j^{(c)} = \boldsymbol{B}^{(c)T}\boldsymbol{m}_j^{(c)}$, then $\boldsymbol{L}\boldsymbol{q}_j^{(c)} = \boldsymbol{r}_j^{(c)}$. $\boldsymbol{r}_j^{(c)}$ is updated by adding $m_j^{(c)}\boldsymbol{b}_i^{(c)}$, then appending the new component $m_j^{(c)} b_i^{(c)}$. Overwrite $\boldsymbol{q}_j^{(c)}$ by

$$\boldsymbol{a}_j^{(c)} = \boldsymbol{L}^{-1}\left(\boldsymbol{r}_j^{(c)} + m_j^{(c)}\boldsymbol{b}_i^{(c)}\right) = \boldsymbol{q}_j^{(c)} + m_j^{(c)}\left(\boldsymbol{L}^{-1}\boldsymbol{b}_i^{(c)}\right)$$

(note that the $O(d^2)$ backsubstitution has to be done only once) and append the $\boldsymbol{a}_j^{(c)}$ to $\boldsymbol{X}_Q$ to be dragged along when $\boldsymbol{L}$ is updated, then the first d components of $\boldsymbol{q}_j^{(c)\prime}$ are given by $\boldsymbol{a}_j^{(c)\prime}$. The last component is

$$q_j^{(c)} = l^{-1}\left(-\boldsymbol{l}^T\boldsymbol{a}_j^{(c)\prime} + m_j^{(c)} b_i^{(c)}\right).$$

Finally, we need to update the $\boldsymbol{\beta}$ vectors from (6). $\boldsymbol{\beta}^{(1,c)}$ receives the new component

$$l_i^{(c)-1}\left(d_i^{(c)-1/2} w_{d+1}^{(c)} - \boldsymbol{l}_i^{(c)T}\boldsymbol{\beta}^{(1,c)}\right).$$

We append $\sum_{k=1}^{d} v_k(\boldsymbol{m}_{I_k}^{(c)\prime})_{d+1}$ to $\boldsymbol{\beta}^{(3,c)}$, then add $v_{d+1}\boldsymbol{m}_i^{(c)\prime}$. The update of $\boldsymbol{\beta}^{(2)}$ parallels the $\boldsymbol{q}$ stubs update, since $\boldsymbol{L}\boldsymbol{\beta}^{(2)} = \boldsymbol{r}$ with $\boldsymbol{r} = \sum_c \boldsymbol{B}^{(c)T}\boldsymbol{\beta}^{(1,c)}$. First,

$$\boldsymbol{a} = \boldsymbol{L}^{-1}\boldsymbol{r}'_{1\dots d} = \boldsymbol{\beta}^{(2)} + \sum_c \beta_{d+1}^{(1,c)\prime}\left(\boldsymbol{L}^{-1}\boldsymbol{b}_i^{(c)}\right).$$

Appending $\boldsymbol{a}$ to $\boldsymbol{X}_Q$ to be dragged along, we obtain the first d components of $\boldsymbol{\beta}^{(2)\prime}$ as $\boldsymbol{a}'$, and

$$\beta^{(2)}_{d+1}{}' = l^{-1}\left(\sum_c \beta^{(1,c)}_{d+1}{}' b^{(c)}_i - \boldsymbol{l}^T\boldsymbol{a}'\right).$$

There does not seem to be a simple incremental update rule for $\boldsymbol{\beta}^{(4)}$ so we recompute

$$\boldsymbol{\beta}^{(4)\prime} = \sum_{k=1}^{d+1} v_k \sum_c \boldsymbol{q}^{(c)\prime}_{I_k}.$$

The update of $\boldsymbol{K}^{(c)}_{\cdot,I}\boldsymbol{v}$ is obvious. This completes the description of the update of the representation after inclusion of i. The cost is $O(C\,d^2)$ for the core representation. Each stub update is $O(C\,d)$. In a simple implementation, we maintain and update stubs for all n patterns and the update is $O(n\,C\,d)$. The update of $\boldsymbol{K}^{(c)}_{\cdot,I}\boldsymbol{v}$ is $O(n\,C)$ in this case, but in general we need only those components j for which we maintain stub vectors. It is important to note that the stub vectors can be computed incrementally in a delayed fashion, so a more sophisticated implementation would maintain a cache of partially complete stubs for a subset of all patterns and update stubs on demand, as a multi-class variant of what is called *randomized greedy selection* in [6, 10]. Such an implementation is subject to future work.

6 ADF Projection Onto Restricted Gaussian Family

Recall from Section 3 that the key idea for achieving a complexity linear in C is to restrict the form of the precision matrices $\boldsymbol{\Pi}_i$ in the site approximations $N^U(\boldsymbol{u}_i|\boldsymbol{b}_i,\boldsymbol{\Pi}_i)$ which replace the true likelihood terms $P(y_i|\boldsymbol{u}_i)$ in the IVM approximation. Their structure has to comply with (2). In this section we show how ADF projections[12] can be done onto this restricted family of site approximations. Note that these projections are required not only prior to inclusion of a new pattern i into the active set I, but also in order to be able to *score* a pattern j as candidate for inclusion, using one of the information-theoretic criteria described in Section 7. It is therefore important that the projections can be computed very efficiently. It turns out that the projections require the solution of a non-convex optimization problem in $\mathbb{R}^C$ which can be addressed using a double-loop scheme with convex optimization in the inner loop. A reader not interested in details may skip this section.

Let $j \notin I$ and $Q(\boldsymbol{u}_j) = N(\boldsymbol{u}_j|\boldsymbol{h}_j,\boldsymbol{A}_j)$ be the marginal of the current posterior approximation. As shown in Section 4, the marginal moments can be computed from the stub vectors as follows:

$$\begin{aligned}
\boldsymbol{A}_j &= \operatorname{diag}\left(\boldsymbol{K}^{(c)}_j - \|\boldsymbol{m}^{(c)}_j\|^2\right)_c + \boldsymbol{Q}_j^T\boldsymbol{Q}_j, \quad \boldsymbol{Q}_j = \left(\boldsymbol{q}^{(1)}_j \dots \boldsymbol{q}^{(C)}_j\right),\\
\boldsymbol{h}_j &= \boldsymbol{h}^{(1)}_j + \boldsymbol{h}^{(2)}_j + \left(\boldsymbol{K}^{(c)}_{j,I}\boldsymbol{v}\right)_c,
\end{aligned}$$

where $\boldsymbol{h}^{(1)}$, $\boldsymbol{h}^{(2)}$ are given by (7) and (8). The cost is $O(C^2\,d)$ for each marginal. Note that in order to guarantee an overall complexity of $O(C\,d^2\,n)$, we can afford to compute $O(n/C)$

[12] An ADF projection (or Bayesian on-line update) is required if an unseen pattern is to be included into the belief representation. We do not treat EP projections here which are used to *iterate* ADF projections for previously included patterns to *refine* the belief representation. Needless to say, our framework applies to this more general case just as well.

marginals prior to each inclusion into I, thus to score about n/C inclusion candidates. This is in contrast to the binary case where we can afford to score *all* remaining points, the reason being that in the binary case the marginal moments itself can be updated incrementally, while in the general case they have to be computed from the stub vectors on demand.

For the ADF projection, we form the "tilted" distribution

$$\hat{P}(\boldsymbol{u}_j) \propto P(y_j|\boldsymbol{u}_j)Q(\boldsymbol{u}_j)$$

and project it onto the family induced by the site approximations using moment matching:

$$(\boldsymbol{b}_j, \boldsymbol{\Pi}_j) = \underset{\boldsymbol{b},\boldsymbol{\Pi}}{\operatorname{argmin}} \mathrm{D}\left[\hat{P}(\boldsymbol{u}_j) \,\middle\|\, \propto N^U(\boldsymbol{u}_j|\boldsymbol{b},\boldsymbol{\Pi})Q(\boldsymbol{u}_j)\right]$$

where $\boldsymbol{\Pi} = \operatorname{diag}\boldsymbol{\pi} - \alpha^{-1}\boldsymbol{\pi}\boldsymbol{\pi}^T$, $\alpha = \mathbf{1}^T\boldsymbol{\pi}$, $\boldsymbol{\pi} \succ \mathbf{0}$. The relative entropy satisfies a Pythagorean-like equation, which means that $\hat{P}$ can be replaced by a Gaussian with the same mean and covariance matrix. To be concrete, let $\tilde{Q}$ denote this Gaussian and Q^{new} the new marginal to be determined, then

$$\mathrm{D}\left[\hat{P} \,\middle\|\, Q^{new}\right] = \mathrm{D}\left[\hat{P} \,\middle\|\, \tilde{Q}\right] + \mathrm{E}_{\hat{P}}\left[\log\tilde{Q} - \log Q^{new}\right].$$

The integrand of the right hand side term is a quadratic in $\boldsymbol{u}_j$, so the expectation can as well be done w.r.t. $\tilde{Q}$ (which has the same moments as $\hat{P}$ up to second order). Since $\hat{P}$ is not Gaussian, we have to resort to numerical quadrature to compute its mean and covariance matrix. We use the method of *exact monomials* [3] which is suitable if $\log P(y_j|\boldsymbol{u}_j)$ is smooth and C is not too large.[13] Denote the moments by $\hat{\boldsymbol{h}}_j$, $\hat{\boldsymbol{A}}_j$. We can match the mean exactly by setting

$$\boldsymbol{b}_j = \boldsymbol{A}_j^{-1}\left(\hat{\boldsymbol{h}}_j - \boldsymbol{h}_j\right) + \boldsymbol{\Pi}_j\hat{\boldsymbol{h}}_j,$$

leaving the computation of $\boldsymbol{\pi} = \boldsymbol{\pi}_j$. The remaining relative entropy is up to constants

$$-\log\left|\boldsymbol{M}\hat{\boldsymbol{A}}_j\right| + \operatorname{tr}\boldsymbol{M}\hat{\boldsymbol{A}}_j, \quad \boldsymbol{M} = \boldsymbol{A}_j^{-1} + \boldsymbol{\Pi}.$$

Thus, the problem is to minimize

$$f = -\log\left|\boldsymbol{A}_j^{-1} + \boldsymbol{\Pi}\right| + \operatorname{tr}\hat{\boldsymbol{A}}_j\boldsymbol{\Pi}, \quad \boldsymbol{\Pi} = \operatorname{diag}\boldsymbol{\pi} - \alpha^{-1}\boldsymbol{\pi}\boldsymbol{\pi}^T,\ \alpha = \mathbf{1}^T\boldsymbol{\pi},$$

subject to $\boldsymbol{\pi} \succ \mathbf{0}$. Here, $\boldsymbol{A}_j \succeq \mathbf{0}$ and $\hat{\boldsymbol{A}}_j \succ \mathbf{0}$. There is no analytic solution, but we can use ideas from convex optimization [1] to solve this problem efficiently. First,

$$f = -\log\left|\boldsymbol{A}_j^{-1} + \boldsymbol{\Pi}\right| + \left(\operatorname{diag}\hat{\boldsymbol{A}}_j\right)^T\boldsymbol{\pi} - \alpha^{-1}\boldsymbol{\pi}^T\hat{\boldsymbol{A}}_j\boldsymbol{\pi}.$$

We show that $-\log\left|\boldsymbol{A}_j^{-1} + \boldsymbol{\Pi}\right|$ is convex in $\boldsymbol{\pi}$. First, $\boldsymbol{\Pi} \mapsto -\log\left|\boldsymbol{A}_j^{-1} + \boldsymbol{\Pi}\right|$ is convex and non-increasing w.r.t. the partial ordering $\preceq$ over symmetric matrices given by the positive

[13] Exact monomials are a generalization of Gaussian quadrature to multiple dimensions. For small to moderate dimensions C, quadrature rules are much more accurate than Monte Carlo approximations which would sample from $Q(\boldsymbol{u}_j)$ and evaluate the likelihood factor over the sample (in fact, we have seen rather disastrous errors in the covariance matrix with $C = 10$ even for as many as a few thousand Gaussian samples).

semidefinite cone. Next, we show that $\boldsymbol{\pi} \mapsto \boldsymbol{\Pi}$ is matrix-concave for $\boldsymbol{\pi} \succ \mathbf{0}$, and the composition rules in [1], Sect. 3.6.2 imply the convexity of the composition w.r.t. $\boldsymbol{\pi} \succ \mathbf{0}$. We have to show that $\boldsymbol{y}^T\boldsymbol{\Pi}\boldsymbol{y}$ is concave in $\boldsymbol{\pi} \succ \mathbf{0}$ for every $\boldsymbol{y} \in \mathbb{R}^d$. Now,

$$\boldsymbol{y}^T\boldsymbol{\Pi}\boldsymbol{y} = -\left((\mathbf{1}^T\boldsymbol{\pi})^{-1}(\boldsymbol{y}^T\boldsymbol{\pi})^2 - \boldsymbol{\pi}^T(\operatorname{diag}\boldsymbol{y}\boldsymbol{y}^T)\right).$$

The expression within the parantheses is the sum of a quadratic-over-linear function (see [1], Sect. 3.1.5) and a linear one, thus convex in $\boldsymbol{\pi}$ if $\mathbf{1}^T\boldsymbol{\pi} > 0$.

However, since $\hat{\boldsymbol{A}}_j \succ \mathbf{0}$, $\boldsymbol{\pi}^T\hat{\boldsymbol{A}}_j\boldsymbol{\pi}$ is convex, so that f is the difference of convex functions and in general *not* convex. Still, f has a very simple structure, the concave part is purely quadratic. We use a standard double-loop scheme to minimize f. The idea is to upper bound the negative quadratic by a hyperplane (i.e. making use of its Legendre-Fenchel transform), in the inner loop to minimize this convex upper bound subject to the convex constraint $\boldsymbol{\pi} \succ \mathbf{0}$, in the outer loop to reset the hyperplane upper bound to make contact at the new $\boldsymbol{\pi}$. The very same idea is used all over in machine learning and statistics, for example in the expectation maximization (EM) algorithm, the variational mean-field Bayesian approximation, convergent double loop variants of loopy belief propagation, etc. If $\boldsymbol{\pi} = \alpha\boldsymbol{x},\ \alpha = \mathbf{1}^T\boldsymbol{\pi}$, the Fenchel inequality states that

$$-\boldsymbol{x}^T\hat{\boldsymbol{A}}_j\boldsymbol{x} \le -2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{q}^T\hat{\boldsymbol{A}}_j^{-1}\boldsymbol{q} \quad \text{for all } \boldsymbol{q} \in \mathbb{R}^d.$$

Thus,

$$f \le f_{\boldsymbol{q}} := -\log\left|\boldsymbol{A}_j^{-1} + \boldsymbol{\Pi}\right| + \boldsymbol{b}^T\boldsymbol{\pi}, \quad \boldsymbol{b} = \operatorname{diag}\hat{\boldsymbol{A}}_j - 2\boldsymbol{q} + \left(\boldsymbol{q}^T\hat{\boldsymbol{A}}_j^{-1}\boldsymbol{q}\right)\mathbf{1},$$

where we use that $\alpha = \mathbf{1}^T\boldsymbol{\pi}$. For fixed $\boldsymbol{x}$, the Fenchel inequality is an equality for $\boldsymbol{q} = \hat{\boldsymbol{A}}_j\boldsymbol{x} = \hat{\boldsymbol{A}}_j\boldsymbol{\pi}/\alpha$ which shows that the outer loop update is analytic. Note also that $\boldsymbol{q}^T\hat{\boldsymbol{A}}_j^{-1}\boldsymbol{q} = \boldsymbol{q}^T\boldsymbol{x}$ at that point, so $\hat{\boldsymbol{A}}_j^{-1}$ never has to be computed. In the inner loop, we minimize $f_{\boldsymbol{q}}$ (for fixed $\boldsymbol{q}$) w.r.t. $\boldsymbol{\pi} \succ \mathbf{0}$. As a convex problem, it has a global minimum[14] which can be found very efficiently by a number of gradient-based optimizers. We parameterize the feasible set directly using $\boldsymbol{\pi} = \exp(\boldsymbol{t})$ and use a Quasi-Newton optimizer[15] to find a minimum point $\boldsymbol{t}_*$. The gradient is given by

$$df_{\boldsymbol{q}} = \left(-\left(\operatorname{diag}\boldsymbol{M}^{-1}\right)\circ\boldsymbol{\pi} - \alpha^2(\boldsymbol{\pi}^T\boldsymbol{v})\boldsymbol{\pi} + 2\alpha^{-1}\boldsymbol{\pi}\circ\boldsymbol{v} + \boldsymbol{b}\circ\boldsymbol{\pi}\right)^T d\boldsymbol{t},$$

where $\boldsymbol{M} = \boldsymbol{A}_j^{-1} + \boldsymbol{\Pi},\ \boldsymbol{v} = \boldsymbol{M}^{-1}\boldsymbol{\pi}$ and $(\alpha_i)_i \circ (\beta_i)_i := (\alpha_i\beta_i)_i$ denotes the Hadamard (or Schur) product. In addition, our implementation allows to run the inner loop optimization in either a "sloppy" or an "accurate" mode: the former is used initially, requiring convergence to low accuracy only, until the changes in $\boldsymbol{q}$ in the outer loop become small, after which the accurate mode is used to obtain convergence to high accuracy.

In our practical experience so far, the double-loop optimization converges very quickly to a local minimum of the criterion f. Since f is not convex, this might not be the global

[14] The minimum might be attained at a boundary point of the open feasible set only, in which case we opt for an ε-optimal feasible point for some small $\varepsilon > 0$.

[15] Evaluating the Hessian proves very messy, and in our practical experience the purely gradient-based inner loop optimization converges extremely quickly.

minimum.[16] It turns out that a good initialization of $\boldsymbol{\pi}_j$ is important. Again, we use the analogy to the Laplace approximation (see Section 3) where $\boldsymbol{\pi}_j$ would be the likelihood evaluated at the posterior mode $\hat{\boldsymbol{u}}_j$. If the final classifier performs well, this is close to the delta distribution $\boldsymbol{\delta}_{y_j} = \boldsymbol{I}_{\cdot,y_j}$ for most points. In our scheme, we initialize $\boldsymbol{\pi}_j$ to a convex combination of $\boldsymbol{\delta}_{y_j}$ and the current predictive distribution $Q(y_j|\boldsymbol{x}_j, D)$ obtained from $Q(\boldsymbol{u}_j)$ using quadrature.

7 Scoring Criteria For Active Set Selection

In the context of classification, individual patterns can carry a very different amount of "information" about the shape of the final predictor, however "information" is defined in this context. For example, we can ask by how much the decision boundary of the final predictor changes if we remove individual patterns, or by how much our uncertainty in the boundary position decreases if we add in a pattern. In this picture, patterns close to the decision boundary or even more so misclassified patterns carry most information. Since our goal is to extract a very small active set I from among the training sample D (i.e. $d \ll n$), it is essential that we manage to identify highly *informative* patterns in D. Of course, we cannot hope for an optimal selection of I due to our tight constraints of $O(C\, d^2\, n)$ running time, $O(C\, d\, n)$ memory, but experiments with the binary IVM show that good active sets can be found in practice using simple greedy forward selection driven by information-theoretic scoring criteria originating in active learning (sequential design). In this section we show how to generalize these criteria to the multi-class case. Note that they are especially attractive in the case of GP models because they can be computed exactly given the Gaussian approximations, in marked contrast to complex parametric architectures such as multi-layer perceptrons where Gaussian approximations can be very poor and information criteria based on these can give misleading results.

Here, we focus on the (instantaneous) *information gain score* (see [10] for other scores which can be generalized to the multi-class case in the same way). In order to score $j \notin I$, let $Q(\boldsymbol{u}_j) = N(\boldsymbol{u}_j|\boldsymbol{h}_j, \boldsymbol{A}_j)$ be the marginal before inclusion of j, $Q^{new}(\boldsymbol{u}_j) = N(\boldsymbol{u}_j|\hat{\boldsymbol{h}}_j, (\boldsymbol{A}_j^{-1} + \boldsymbol{\Pi}_j)^{-1})$ the marginal after inclusion of j (see Section 6). The score is defined as

$$\Delta_j^{info} := -\mathrm{D}\left[Q^{new}(\boldsymbol{u}_j) \,\|\, Q(\boldsymbol{u}_j)\right].$$

Using the well-known formula for the relative entropy between Gaussians we have

$$\Delta_j^{info} = -\frac{1}{2}\left(\log|\boldsymbol{M}| + \mathrm{tr}\left(\boldsymbol{M}^{-1} - \boldsymbol{I}\right) + (\hat{\boldsymbol{h}}_j - \boldsymbol{h}_j)^T \boldsymbol{A}_j^{-1}(\hat{\boldsymbol{h}}_j - \boldsymbol{h}_j)\right), \quad \boldsymbol{M} = \boldsymbol{I} + \boldsymbol{\Pi}_j \boldsymbol{A}_j. \tag{9}$$

In order to score inclusion candidate j, we have to compute its marginal moments $\boldsymbol{h}_j$, $\boldsymbol{A}_j$ in $O(d\, C^2)$, determine $\hat{\boldsymbol{h}}_j$, $\boldsymbol{\Pi}_j$ by ADF projection (each gradient costs $O(C^3)$) and compute Δ_j^{info} which can be done in $O(C^3)$ using the LU decomposition of $\boldsymbol{M}$. In order to keep within the overall $O(n\, C\, d^2)$ limits, we can afford to score about n/C candidates prior to each inclusion. These figures assume that C is small to moderate, especially $C < d$. For

[16] f attains its global minimum over the closure of the feasible set because it is continuous on the compact intersection of this closure with a sufficiently large compact ball, and is unbounded above for any sequence $\|\boldsymbol{\pi}_n\| \to \infty$ (the term $\mathrm{tr}\, \hat{\boldsymbol{A}}_j \boldsymbol{\Pi} = \alpha\, \mathrm{tr}\, \hat{\boldsymbol{A}}_j(\mathrm{diag}\, \boldsymbol{x} - \boldsymbol{x}\boldsymbol{x}^T)$, $\boldsymbol{x} = \alpha^{-1}\boldsymbol{\pi}$ dominates for large α and $\mathrm{tr}\, \hat{\boldsymbol{A}}_j(\mathrm{diag}\, \boldsymbol{x} - \boldsymbol{x}\boldsymbol{x}^T) \geq 0$ because $\mathrm{diag}\, \boldsymbol{x} - \boldsymbol{x}\boldsymbol{x}^T \succeq \boldsymbol{0}$).

large C, additional techniques would have to be used to remove the cubic scaling in C, but in this case our use of numerical quadrature rules would also be questionnable. An extension to a large number of classes is subject to future work.

A slightly cheaper variant avoids the ADF projection by using the true covariance matrix $\hat{\boldsymbol{A}}_j$ of the tilted distribution $\hat{P}$ for Q^{new}. The score has the same form as (9), but with $\boldsymbol{M}^{-1} = \boldsymbol{A}_j^{-1}\hat{\boldsymbol{A}}_j$ (the Cholesky factor of $\boldsymbol{A}$ is available from the computation of the tilted moments). In the experiments reported here, we make use of this simpler variant, but an experimental comparison between the two variants will be done in future work.

8 Extensions of the Myopic Scheme

We are now in a position to propose a generalization of the binary IVM to C classes which scales as $O(n\,C\,d^2)$, simply by combining the myopic forward selection scheme used in [6] with the additional ideas developed here. However, initial experiments with this scheme showed unsatisfactory performance. In this section we analyze this problem and propose two significant modifications to the simple myopic scheme. At this point we have not yet tested the significance of each of these modifications in isolation on larger experiments, but will do this in future work. While the modified scheme is harder to describe and implement, it is not significantly more costly than the simple myopic one (we remind the reader of out working assumption: $n \gg d > C$).

Observations of the simple myopic scheme on toy experiments show the following problem. For kernels with rather small variance and large length scales, the inclusion of patterns hardly decreases the prior uncertainty in any predictions, including the prediction at points in the active set. For a prior favouring paths of higher variance and more rapid changes, the first pattern included determines all subsequent predictions completely (all patterns are assigned to the same class with high probability). The problem may be that predictions use the training set information only through the “bottleneck” of the active set, thus if any of the patterns included early determine the belief very strongly, this can result in the information in subsequent patterns being ignored completely. In short, while the simplest possible “myopic” forward selection approach worked satisfyingly in the binary case, a more elaborate scheme may be required for $C > 2$ classes.

The first modification is the introduction of *likelihood reweighting factors*. Eventually we would like each pattern in the active set to determine the belief significantly, but we might have to downweight this influence *initially* when the active set I is still small. To this end, we introduce reweighting factors $\gamma_i \in (0, 1]$ into the likelihood terms:

$$P(y_i|\boldsymbol{u}_i) \propto \exp\left(\gamma_i u_i^{(y_i)} + \beta_{y_i}\right).$$

The γ_i should be regarded as parameters of the approximation or the algorithm (akin to a temperature parameter in annealing schemes), *not* as parameters of the model. In fact, we will have $\gamma_i = 1$ for all i in the end. Any schedule of updating the γ_i is a heuristic: a simple one is suggested in Section 8.5. Suffice to say that $\gamma_i = 1$ for all patterns i which have been in I more than a fixed number of inclusions. This is important to ensure the feasibility of the whole scheme.

Second, a *joint optimization* of the site parameters for at least a subset of the active patterns (in I) is required. A possible explanation for the simple approach to fail is that site parameters are determined once and never changed later on. The myopic scheme restricts itself to the behaviour of an on-line algorithm, while this is not required by the problem setting (the training data can be accessed in arbitrary batches). The parameters are never modified jointly (and iteratively) with others. It can happen that patterns which are included early obtain unreasonable site parameter values, simply because their computation is based on the current belief only. If these values remain fixed, the errors cannot be corrected later on,[17] in fact may lead to unreasonable subsequent decisions. The opportunity of joint optimization is especially attractive in combination with the reweighting of the likelihood factors. For example, factors of points included into I early can be raised gradually, their site parameters modified in conjunction with later patterns.

However, to remain feasible as a sparse method which makes use of the block-diagonal structure of $\boldsymbol{K}$ (in the sense of Section 4) it is necessary to freeze the site parameters of points in I eventually (and to set their $\gamma_i = 1$). This is because we still need to be able to score a large number of candidates for every inclusion, which requires some form of stub vectors. Since these depend in a complicated way on all site parameters, we can only maintain and update them for patterns whose parameters do not change in the future. In the sequel, we describe a scheme which includes all these modifications. It features expectation propagation (EP) iterations on a "liquid" subset of I. Patterns are removed gradually from this subset and their site parameters are frozen. The representation described above is maintained w.r.t. the "solid" (or "frozen") subset of I only.

8.1 General Description

At any time, the active set $I = \{I_1, \dots, I_d\}$ is partitioned into the solid (or frozen) subset $I^f = \{I_1, \dots, I_{d-L}\}$ and the liquid subset $I^l = \{I_{d-L+1}, \dots, I_d\}$ of size no larger than L. Note that I^f contains the patterns included earlier. Both sets can be empty, but if $I^f \neq \emptyset$, then $|I^l| = L$. Site parameters of patterns in the solid subset are fixed ("frozen") while parameters of patterns in I^l can be changed arbitrarily. We allow for likelihood reweighting factors $\gamma_i \in (0, 1]$ for all $i \in I^l$, while $\gamma_i = 1$ for all other patterns. These factors may change (for $i \in I^l$) in an arbitrary way between inclusions.

The representation described in Section 4 is used here as well, but it is based on the solid active set only (as are the stub buffers). In the descriptions above, replace I by I^f, d by $d-L$. We will call it *solid representation* in order to distinguish it from the EP representation to be introduced shortly. The algorithm cycles through different phases for each inclusion. For the moment, we ignore "edge effects" (empty I, empty I^f, etc.) which occur at the beginning and the end. In the *selection phase*, a large number of candidate patterns outside of I are scored to determine which to include next. This phase is described in Section 8.2. In the *inclusion phase*, the new pattern is included into I, the first pattern in I^l is frozen (moved into I^f) and the representation is updated. Also, the γ_i factors are modified. This phase is described in Section 8.3. Finally, in the *EP phase*, the site parameters for liquid patterns are updated jointly using EP iterations. This requires a different representation which cannot exploit block-diagonal matrix structure and scales cubically in LC. The EP

[17] "Deletion" and "exchange" moves are possible in the binary IVM, but typically lead to numerical instabilities in the updates of the representation.

phase is described in Section 8.4. In Section 8.5, we discuss further details such as what happens at the beginning and the end, and how the γ_i likelihood reweighting factors can be chosen.

8.2 The Selection Phase

In the selection phase, a large number of candidates from $\{1,\dots,n\} \setminus I$ are scored to selection a suitable pattern for the next inclusion. As precondition, the (solid) representation described in Section 4 (replace I by I^f, d by $d-L$) is given for the solid active set I^f. We are given a selection index J disjoint from I whose patterns are to be scored (the size of J can be roughly $O(n)$, more below). In order to compute the score described in Section 7 we need the marginal moments, thus the stub vectors (5) for $j \in J$ w.r.t. the full active set I. Recall that we have stubs available (for all j) for the solid representation.

In the selection phase, we build an *extended representation* and extended stub buffers starting from the solid representation. Extended stubs are required for all $j \in I \cup J$. We do this by "including" the patterns $I^l = \{I_{d-L+1}, \dots, I_d\}$ as described in Section 5. Note that the solid representation is not overwritten, in particular the extended q stubs must *not* overwrite the normal ones. It is most efficient to allocate separate buffers for the extended stubs, although this is redundant in case of the m stubs. Once the extended stubs are completed, the marginal moments for all $j \in J$ can be computed, the patterns can be scored and the winner selected as before.

Note that our description of the selection phase may require redundant evaluations of rows of the covariance matrices $\boldsymbol{K}^{(c)}$ (the same pattern i will be in I^l for up to L inclusions) if L subsequent selection indexes J do overlap. If additional memory is available, a n-by-L-by-C buffer should be maintained storing[18] $\boldsymbol{K}^{(c)}_{\cdot,I^l}$, $c = 1,\dots,C$ (since $L \ll d$, this is typically not a dominating buffer).

Since each stub update is $O(C\,d)$, we need $O((|J|+d)\,C\,d\,L)$ to extend all stubs $j \in J \cup I$. To stay within our resource limitations of $O(n\,C\,d^2)$, we require $|J| \le \min\{n/L, n/C\}$. Recall that C is moderate and L can be chosen fairly small.[19]

8.3 The Inclusion Phase

In this phase, pattern $i \notin I$ is to be included into I. Typically, i is the winner from the selection phase. As precondition, we require the marginal moments $\boldsymbol{h}_i$, $\boldsymbol{A}_i$ for i, which we can compute from the EP phase representation. If $|I^l| = L$, the first pattern I_{d-L+1} in I^l is moved to I^f, i.e. its site parameters are frozen. Note that it is sensible to require that $\gamma_{I_{d-L+1}} = 1$, otherwise these parameters are computed based on a wrong likelihood factor). Freezing means that the solid representation and the stubs are updated as described in Section 5. Finally, we require initial site parameters for i which are determined as described in Section 6. Furthermore the reweighting factors γ_j are updated following a schedule to be described below. We require that $\gamma_{I_{d-L+1}} = 1$ and will typically have $\gamma_i < 1$. It is not

[18] One of the advantages of the IVM is that redundant covariance function evaluations are typically not required, which is especially important if the kernel evaluations are expensive.

[19] The simple myopic scheme has $L = 1$, and already a small $L > 1$ can make a significant improvement.

necessary to update the representation used in the EP phase, since it has to be recomputed at the beginning of this phase anyway. The inclusion phase ends by including i into I.

As shown in Section 5, the cost for including a new point into I^f is $O(n\,d\,C)$ (for updating all stub vectors).

8.4 The EP Phase

In the EP phase, the site parameters for all liquid active patterns in I^l are jointly optimized using EP updates. As precondition, we require the solid representation to be given for I^f (which may be empty), furthermore all patterns in I^l must have initial site parameter values. The EP phase can only be run if $|I^l| \geq 2$ (see Section 8.5).

Before we describe the phase, let us remark that we actually have two different options for designing an EP phase. Option I (which we do not choose here) is to iterate using the full representation w.r.t. I, but to only ever choose $i \in I^l$ for site updates. It is not hard to see that each EP step would require $O(C\,d^2)$, i.e. an EP iteration over all $i \in I^l$ would be $O(L\,C\,d^2)$. Option II chosen here is to iterate EP only on the marginal distribution $Q(\boldsymbol{u}_{I^l})$. As we see shortly, this means that we cannot exploit the block-diagonal structure of the prior covariance matrix anymore, thus have to deal with unstructured (CL)-by-(CL) matrices. We will see that each EP step takes $O(C^3\,L^2)$, thus an EP iteration is $O((C\,L)^3)$. This is roughly no slower than option I if $d \geq C\,L$ (which will typically be the case). The drawback of option II is that a separate representation has to be used which complicates the implementation. Also, a $O(d\,C^2\,L^2)$ computation is required initially (which is again cheaper than option I if $d \geq C\,L$).

In order to make progress, the following argument is required. We would like to run EP iterations on the approximating distribution

$$Q(\boldsymbol{u}_I) \propto P(\boldsymbol{u}_I) N^U(\boldsymbol{u}_{I^f}|\boldsymbol{b}_{I^f}, \boldsymbol{\Pi}_{I^f}) N^U(\boldsymbol{u}_{I^l}|\boldsymbol{b}_{I^l}, \boldsymbol{\Pi}_{I^l}),$$

but will only ever change site parameters for $i \in I^l$. If we define

$$Q^f(\boldsymbol{u}_I) \propto P(\boldsymbol{u}_I) N^U(\boldsymbol{u}_{I^f}|\boldsymbol{b}_{I^f}, \boldsymbol{\Pi}_{I^f}),$$

we have

$$Q(\boldsymbol{u}_I) \propto Q^f(\boldsymbol{u}_I) N^U(\boldsymbol{u}_{I^l}|\boldsymbol{b}_{I^l}, \boldsymbol{\Pi}_{I^l}).$$

But this is just about the same situation as the original setup if $\boldsymbol{u}_I$ replaces $\boldsymbol{u}$, $\boldsymbol{u}_{I^l}$ replaces $\boldsymbol{u}_I$, and the prior $P(\boldsymbol{u})$ is replaced by the "prior" $Q^f(\boldsymbol{u}_I)$. If we do these replacements, I will be the set of "all" points, I^l will be the "active set". The only difference to the original setup is that the marginal "prior" $Q^f(\boldsymbol{u}_{I^l})$ is not zero-mean anymore, and that its covariance matrix does *not* have a block-diagonal structure. We cannot use the same representation as above, but have to work with unstructured (CL)-by-(CL) matrices.

We will denote the moments of Q^f using the superscript f, these are the moments we obtain from the solid representation (with I^f as active set). As for $Q^f(\boldsymbol{u}_{I^l}) = N(\boldsymbol{h}^f, \boldsymbol{G})$ we have

$$\boldsymbol{G} = \boldsymbol{A}^f_{I^l} = \boldsymbol{K}_{I^l} - \boldsymbol{K}_{I^l,I^f} \boldsymbol{\Phi}_{I^f} \boldsymbol{K}_{I^f,I^l} = \operatorname{diag}\left(\boldsymbol{K}^{(c)}_{I^l} - \boldsymbol{M}^{(c)T}_{I^l} \boldsymbol{M}^{(c)}_{I^l}\right)_c + \boldsymbol{Q}^T_{I^l} \boldsymbol{Q}_{I^l} \tag{10}$$

with

$$\boldsymbol{M}^{(c)}_{I^l} = \left(\boldsymbol{m}^{(c)}_j\right)_{j \in I^l}, \quad \boldsymbol{Q}^{(c)}_{I_l} = \left(\boldsymbol{q}^{(c)}_j\right)_{j \in I^l}, \quad \boldsymbol{Q}_{I_l} = \left(\boldsymbol{Q}^{(c)}_{I_l}\right)_c \in \mathbb{R}^{d-L,CL},$$

the computation is $O(d\,C^2\,L^2)$ given the stubs. The computation of $\boldsymbol{h}^f$ is described in Section 4.

The following description is very technical. Readers not interested in the details may skip the remainder of the section in which we show how a suitable representation of size $O(C^2\,L^2)$ can be maintained which allows EP updates of the site parameters in $O(C^3\,L^2)$, so that a complete iteration over all sites is $O(C^3\,L^3)$. While our main concern here is numerical stability, the deletion/inclusion nature of EP combined with frequent use of rank-one updates may cause problems. We give some comments how to deal with these.

We work with inner grouping w.r.t. c in the rest of this section. Recall our notation from Section 3. We convert $\boldsymbol{G}$ from (10) to $\hat{\boldsymbol{P}}_{\leftrightarrow}\boldsymbol{G}\hat{\boldsymbol{P}}_{\leftrightarrow}^T$, $\boldsymbol{h}^f$ to $\hat{\boldsymbol{P}}_{\leftrightarrow}\boldsymbol{h}^f$, etc. This grouping is more natural in the context here, because there is no block structure for the inner grouping w.r.t. datapoints anymore (as opposed to $\boldsymbol{K}$, the covariance matrix $\boldsymbol{G}$ has no block structure if $I^f \neq \emptyset$), while we still have $\boldsymbol{\Pi}_{I^l} = \mathrm{diag}(\boldsymbol{\Pi}_i)_i$ (inner grouping w.r.t. c). In the sequel, we drop the subscript I^l and use an absolute indexing of this subset, i.e. assume[20] that $I^l = \{1, \dots, L\}$. Matrices will be (CL)-by-(CL) and follow the inner grouping w.r.t. c unless said otherwise. We also have to use namings which may conflict with other sections, so definitions made here are meant to be local and override definitions elsewhere.

Let $\boldsymbol{\Pi} = \boldsymbol{V}\boldsymbol{V}^T$. We can choose $\boldsymbol{V} = \mathrm{diag}(\boldsymbol{V}_i)_i$ with $\boldsymbol{\Pi}_i = \boldsymbol{V}_i\boldsymbol{V}_i^T$.[21] The posterior covariance matrix is

$$\boldsymbol{A} = \left(\boldsymbol{G}^{-1} + \boldsymbol{\Pi}\right)^{-1} = \boldsymbol{G} - \boldsymbol{M}^T\boldsymbol{M}, \quad \boldsymbol{M} = \boldsymbol{L}^{-1}\boldsymbol{V}^T\boldsymbol{G}, \quad \boldsymbol{B} = \boldsymbol{I} + \boldsymbol{V}^T\boldsymbol{G}\boldsymbol{V} = \boldsymbol{L}\boldsymbol{L}^T. \tag{11}$$

Note that $\boldsymbol{B}$ is positive definite with all eigenvalues ≥ 1, thus very well-conditioned. The posterior mean is

$$\boldsymbol{h} = \tilde{\boldsymbol{b}} - \boldsymbol{M}^T\boldsymbol{\beta}, \quad \tilde{\boldsymbol{b}} = \boldsymbol{G}\boldsymbol{b} + \boldsymbol{h}^f,\ \boldsymbol{\beta} = \boldsymbol{L}^{-1}\boldsymbol{V}^T\tilde{\boldsymbol{b}}. \tag{12}$$

The EP representation consists of $\boldsymbol{L}$, $\boldsymbol{M}$, $\boldsymbol{\beta}$. We also require $\boldsymbol{L}^{-1}$ explicitly, maintaining $\boldsymbol{P}^{(i)} = (\boldsymbol{L}^{-1})_{\cdot,i}$.

In order to do an EP step, we require the marginal $Q(\boldsymbol{u}_i) = N(\boldsymbol{h}_i, \boldsymbol{A}_i)$, where $\boldsymbol{h}_i$, $\boldsymbol{A}_i$ are computed via (11) and (12). Before dealing with the EP projection, we describe how the representation is updated afterwards. Suppose the parameters of pattern i are to be updated. Let $\Delta\boldsymbol{V}_i = \boldsymbol{V}_i' - \boldsymbol{V}_i$. We reject i for update if $\boldsymbol{\Pi}_i' - \boldsymbol{\Pi}_i$ is too small in some matrix norm. Let $\boldsymbol{G}_i = \boldsymbol{Q}_i\boldsymbol{Q}_i^T$ (the factors should be precomputed). Since $\boldsymbol{V}' = \boldsymbol{V} + \boldsymbol{I}_{\cdot,i}\Delta\boldsymbol{V}_i\boldsymbol{I}_{i,\cdot}$, we have

$$\boldsymbol{B}' = \boldsymbol{B} + \left(\boldsymbol{I}_{\cdot,i}\Delta\boldsymbol{V}_i^T\boldsymbol{Q}_i + \tilde{\boldsymbol{V}}\right)\left(\boldsymbol{I}_{\cdot,i}\Delta\boldsymbol{V}_i^T\boldsymbol{Q}_i + \tilde{\boldsymbol{V}}\right)^T - \tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T, \quad \tilde{\boldsymbol{V}} = \boldsymbol{V}^T\boldsymbol{G}_{\cdot,i}\boldsymbol{Q}_i^{-T}.$$

Therefore, we can update $\boldsymbol{L}$ using L positive followed by L negative applications of *chollrup* described in Section 5. We drag along the columns of $\boldsymbol{M}$ and all $\boldsymbol{P}^{(j)}$ which are required to update these variables, but can also conveniently be used as follows. *chollrup* requires subsequent columns of $\boldsymbol{L}^{-1}\tilde{\boldsymbol{V}}$ and $\boldsymbol{L}^{-1}\boldsymbol{I}_{\cdot,i}\Delta\boldsymbol{V}_i^T\boldsymbol{Q}_i$ with $\boldsymbol{L}$ being up-to-date. Note that $\boldsymbol{L}^{-1}\tilde{\boldsymbol{V}} = \boldsymbol{M}_{\cdot,i}\boldsymbol{Q}_i^{-T}$ and $\boldsymbol{L}^{-1}\boldsymbol{I}_{\cdot,i}\Delta\boldsymbol{V}_i^T\boldsymbol{Q}_i = \boldsymbol{P}^{(i)}(\Delta\boldsymbol{V}_i^T\boldsymbol{Q}_i)$. Since we drag along the columns of $\boldsymbol{M}$ and all $\boldsymbol{P}^{(j)}$, columns of the latter expressions are available based on the correct $\boldsymbol{L}$ just when they are required. Our implementation precomputes and stores $\boldsymbol{Q}_i$ and $\boldsymbol{Q}_i^{-T}$.

Recall that *chollrup* results in $\boldsymbol{L}' = \boldsymbol{L}\tilde{\boldsymbol{L}}$ with $\tilde{\boldsymbol{L}}$ having an $O(C\,L^2)$ representation.[22] The

[20] I^l may be smaller than L in the beginning, the modifications to the description are obvious though.

[21] Use the spectral decomposition $\boldsymbol{\Pi}_i = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$, then $\boldsymbol{V}_i = \boldsymbol{U}\boldsymbol{D}^{1/2}$.

[22] $\tilde{\boldsymbol{L}}$ is the product of L factors with $O(C\,L)$ representation.

$\boldsymbol{P}^{(j)}$ are updated simply by dragging along their columns. The update of $\boldsymbol{M}$ is

$$\boldsymbol{M}' = \tilde{\boldsymbol{L}}^{-1}\boldsymbol{M} + \boldsymbol{P}^{(i)\prime}\Delta\boldsymbol{V}_i^T\boldsymbol{G}_{i,\cdot}.$$

Next, $\tilde{\boldsymbol{b}}' = \tilde{\boldsymbol{b}} + \boldsymbol{G}_{\cdot,i}\Delta\boldsymbol{b}_i$ (where $\Delta\boldsymbol{b}_i = \boldsymbol{b}_i' - \boldsymbol{b}_i$). Finally,

$$\boldsymbol{\beta}' = \tilde{\boldsymbol{L}}^{-1}\boldsymbol{\beta} + \boldsymbol{M}'_{\cdot,i}\Delta\boldsymbol{b}_i + \boldsymbol{P}^{(i)\prime}\Delta\boldsymbol{V}_i^T\tilde{\boldsymbol{b}}_i,$$

so $\boldsymbol{\beta}$ has to be dragged along as well. The complete update is $O(C^3\,L^2)$ (the "dragging along" dominates the cost). Numerical stability should be ensured by the fact that $\boldsymbol{B}$ is always well-conditioned. Still, our implementation allows a roll-back together with rejecting i for update should any of the negative *chollrup* break down.

It remains to describe the EP update itself. Let $Q(\boldsymbol{u}_i) = N(\boldsymbol{h}_i, \boldsymbol{A}_i)$ be the marginal. As opposed to the ADF update described in Section 6 we cannot assume that $\boldsymbol{\Pi}_i = \boldsymbol{0}$, thus have to remove the site approximation i from Q. Let

$$Q^{\setminus i}(\boldsymbol{u}_i) = N\left(\boldsymbol{h}_i^{\setminus i}, \boldsymbol{\Lambda}\right), \quad \boldsymbol{\Lambda} = \left(\boldsymbol{I} - \boldsymbol{A}_i\boldsymbol{\Pi}_i\right)^{-1}\boldsymbol{A}_i,\ \boldsymbol{h}_i^{\setminus i} = \left(\boldsymbol{I} - \boldsymbol{A}_i\boldsymbol{\Pi}_i\right)^{-1}\left(\boldsymbol{h}_i + \boldsymbol{A}_i\boldsymbol{b}_i\right)$$

the "cavity" distribution. Note that $Q^{\setminus i} = Q$ if $\boldsymbol{\Pi}_i = \boldsymbol{0}$, $\boldsymbol{b}_i = \boldsymbol{0}$ which is the ADF case. An EP update is done in the same way as an ADF update (see Section 6) with the only difference that the marginal $Q(\boldsymbol{u}_i)$ is replaced by the cavity marginal $Q^{\setminus i}(\boldsymbol{u}_i)$.[23] We cycle over $i \in I^l$ in some random ordering. Candidates i are rejected if the change $\Delta\boldsymbol{\Pi}_i$ is too small, or in the unlikely case that an $\boldsymbol{L}$ update breaks down.[24] The repeated use of rank-one updates may introduce numerical errors, so the representation should be refreshed (*i.e.* recomputed from scratch) after $O(L)$ updates (which costs $O(C^3\,L^3)$). Note that we do not have to run EP updates until convergence[25], but can stop after a fixed number of them.

This completes the description of the EP phase. The representation used here should be retained until the next EP phase, it is required in inclusion phase to compute the marginal moments for the new pattern. The latter works as follows. Let $i \notin I$ be the new pattern. We simply have to apply the initial computation of the EP representation above to the case where I^l is replaced by $\tilde{I} = I^l \cup \{i\}$. Denote $J = \{1, \dots, L\}$ for conciseness. If $\tilde{\boldsymbol{G}} = \boldsymbol{A}^f_{\tilde{I}} \in \mathbb{R}^{C(L+1),C(L+1)}$, then $\tilde{\boldsymbol{G}}_J = \boldsymbol{G}$, so only $\tilde{\boldsymbol{G}}_{\cdot,L+1} \in \mathbb{R}^{C(L+1),C}$ has to be computed from the stubs. After the computation, we permute the components such that the inner grouping is w.r.t. c. Then,

$$\boldsymbol{A}_i = \tilde{\boldsymbol{G}}_{L+1} - \tilde{\boldsymbol{M}}^T\tilde{\boldsymbol{M}}, \quad \tilde{\boldsymbol{M}} = \boldsymbol{L}^{-1}\boldsymbol{V}^T\tilde{\boldsymbol{G}}_{J,L+1}.$$

and

$$\boldsymbol{h}_i = \boldsymbol{h}_i^f + \tilde{\boldsymbol{G}}_{L+1,J}\boldsymbol{b} - \tilde{\boldsymbol{M}}^T\boldsymbol{\beta}.$$

Note that the computation is $O(C^3\,L^2)$: it would not be feasible to compute a large number of marginals that way, the "detour" via the extended representation in the selection phase is necessary.

An iteration over all liquid patterns in the EP phase costs $O((L\,C)^3)$ which is subdominant to other costs if $d \geq L\,C$ which we assume to be true. As mentioned above, our scheme in

[23] For increased stability one can consider "damped" EP updates, but we have not found this necessary in our case.

[24] This did not happen in our experiments so far.

[25] At any rate, convergence is not guaranteed in EP, since it does not descent on a criterion.

the present form is not suitable for the case of large C. Note that the overall contribution of all EP phases is $O(d\,(L\,C)^3)$ which is subdominant if $d \ge L\,C$ and $n \ge C\,L^2$ (the latter is given because $n \gg d$ and L can be chosen small).

8.5 Further Details

What happens in the beginning? First, it does not make sense to run the selection phase if the active set is still very small. As in the binary IVM scheme, we pick the first two or three patterns for I at random.[26] We note that the selection phase can be run with an empty solid set, simply by extending an empty representation, but it requires at least one liquid active pattern. The inclusion phase can be run even if I is empty, as long as marginal moments for the new pattern are supplied. For the very first pattern, we use the prior moments $\boldsymbol{h}_i = \boldsymbol{0}$, $\boldsymbol{A}_i = \boldsymbol{K}_i$, for later patterns we use the EP representation (see details at the end of Section 8.4).

Finally, once the active set I has the desired final size, we run a final EP phase on the liquid set, then use the first part of the selection phase in order to complete the representation (freeze all liquid patterns). The stubs are not required anymore at this point and do not have to be updated.

We propose the following simple update schedule for the likelihood reweighting factors γ_i, $i \in I^l$. Suppose $I^l = \{i_1, \dots, i_k\}$, $k \le L$. If $k = L$, we require that $\gamma_{i_1} = 1$. γ_{i_j} should be nonincreasing in j, and the factors should scale with $d = |I|$. Pick $L_0 < L$, $\gamma^{(0)} \in (0,1)$, $\alpha_0 \in (0,1)$, then

$$\begin{aligned}
\gamma_{i_k} &= 1 - \left(1 - \gamma^{(0)}\right) \exp\left(-\alpha_0 (d-1)\right), \\
\gamma_{i_j} &= \min\left\{1,\, \gamma_{i_k} + (k-j)\Delta\right\}, \quad \Delta = (1 - \gamma_{i_k})/L_0.
\end{aligned}$$

This means that $\gamma_{i_1} = \gamma^{(0)}$ if $d = 1$, then γ_{i_k} is increasing towards 1 with d. For fixed d, the γ_{i_j} decrease linearly towards γ_{i_k} with at most L_0 of them being different from 1.

9 Model Selection

The scheme described so far shows how to approximate Bayesian inference for the latent $\boldsymbol{u}$ conditioned on fixed hyperparameters, such as the parameters of the covariance functions $K^{(c)}$ for each $u^{(c)}$ and the intercept parameters $\boldsymbol{\beta} \in \mathbb{R}^C$. Recall that the dominating cost for an inference step is $O(n\,C\,d^2)$. One of the major advantages of adopting a full Bayesian strategy in practice is that normally *model selection*, *i.e.* the task of assigning appropriate hyperparameter values given the data, is easy to do given a valid inference approximation. Our development here is a direct generalization of the model selection strategy in the binary case, as detailed in [11], Sect. 4.5.2.

It is well known that the correct way of dealing with hyperparameters in Bayesian analysis is to place *hyperpriors* on them and integrate them out. A model consisting of hyperparameters and "primary parameters" (the latent processes $u^{(c)}$ in our case) is referred to

[26]Depending on the task, a more informed "cheap and cheerful" heuristic may be available. We should certainly ensure that the initial patterns come from different classes.

as *hierarchical model*, the hyperparameters are higher up the hierarchy ("away from the observed data") than the primary ones. Strictly speaking there is no reason to treat hyperparameters differently from primary ones, but in practice integrating out the former (even approximately) is often much harder, and a crude but often very effective approximation is to replace the marginalization by a maximization. This is justified by the fact that if the hyperparameters are chosen properly, the hyperposterior tend to approach a sharp peak and can eventually be replaced by a Delta distribution at a mode without losing too much in terms of prediction accuracy.

Let $\boldsymbol{\theta}$ collect all hyperparameters in our model. The *empirical Bayesian* technique of *marginal likelihood maximization* advocates choosing hyperparameter values by maximizing the marginal likelihood

$$P(\boldsymbol{y}|\boldsymbol{\theta}) = \int P(\boldsymbol{y}|\boldsymbol{u}, \boldsymbol{\theta}) P(\boldsymbol{u}|\boldsymbol{\theta}) \, d\boldsymbol{u}$$

of the observed data. Importantly, this criterion does not depend on primary parameters which are integrated out. Intractability of inference, *i.e.* computing $P(\boldsymbol{u}|\boldsymbol{y}, \boldsymbol{\theta})$ typically translates into intractability of computing this score, but interestingly there is a generic way of obtaining a lower bound using any approximate inference scheme. If $Q(\boldsymbol{u})$ is some approximation to $P(\boldsymbol{u}|\boldsymbol{y}, \boldsymbol{\theta})$, then a simple application of Legendre-Fenchel duality gives

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

The slack in this inequality is $\mathrm{D}[Q(\boldsymbol{u}) \,\|\, P(\boldsymbol{u}|\boldsymbol{y}, \boldsymbol{\theta})]$ which measures closeness of Q to the posterior.

In our case, $Q(\boldsymbol{u})$ is given by the representation learned as described above. The marginal likelihood as well as lower bound approximations are typically not convex, so non-convex gradient-based optimization has to be employed. Our optimization strategy is similar to the one detailed in [11]. For fixed active set I and site parameters $\boldsymbol{b}$, $\boldsymbol{\pi}$ the criterion and its gradient w.r.t. $\boldsymbol{\theta}$ can be computed from the representation and the stub vectors as shown below. Since a good choice of I depends on the hyperparameters in a way which is hard to specify, the overall criterion may even be nondifferentiable which precludes simply sticking it into a standard optimizer. We use a Quasi Newton optimizer which is modified as follows. The criterion and gradient computation to drive the computation of search directions is done in "major mode" which means that active set I and site parameters $\boldsymbol{b}$, $\boldsymbol{\pi}$ are determined from scratch as detailed above. On the other hand, during line searches along directions we run conditional inference in "minor mode", keeping I, $\boldsymbol{b}$, $\boldsymbol{\pi}$ fixed. Since the dependence of the latter on $\boldsymbol{\theta}$ is ignored when computing gradients, this seems necessary to achieve a consistent line search. Note that while "minor mode" and "major mode" inference have the same dominating complexity, the former runs much faster because the dominating operations can be done in large blocks.

There are variations of the theme which save running time. For example, I could be fixed over longer periods avoiding the need of the complicated re-selection. It is tempting to fix I very early during optimization and stick with it, but given the assumption that the choice of I is significant for prediction accuracy it does not seem sensible to do so. Note that the algorithm is not strictly a descent method,[27] because the criterion can increase when I, $\boldsymbol{b}$, $\boldsymbol{\pi}$ are recomputed for a new search direction. Especially, due to the discrete nature of I and

[27] As shown below, we will *minimize* the negative log marginal likelihood.

its forward selection, we cannot except to observe convergence to high accuracy, rather the terminal behaviour of the method will be to oscillate around a local minimum point.

It is important to note that we deviate from the conventional method of doing variational Bayesian model selection in several ways. First, we do not choose our inference approximation Q in order to minimize the slack in the bound, as variational Bayesian inference approximation would require. The drawback is that we do not always descend on the criterion and have to employ non-standard optimization code. On the other hand, our posterior approximations may well be much better than any tractable variational Bayesian choice. In fact, the insistence on bounds seems often more of a hindrance and is weakened by the fact that one cannot judge the tightness of the bounds.[28] Second, we do not fix the entire posterior approximation $Q(\boldsymbol{u})$ during gradient computation and line search, but *only* the parameters whose analytical dependence on $\boldsymbol{\theta}$ would be unreasonably hard to specify. In our opinion, this is of central importance in Gaussian process models where the dependence of $Q(\boldsymbol{u})$ on the prior is very strong. Recall that the covariance matrix of Q (which is Gaussian) has the form $\boldsymbol{A} = (\boldsymbol{K}^{-1} + \boldsymbol{\Pi})^{-1}$. Only $\boldsymbol{\Pi}$ depends on I and the site parameters and can be written in terms of $O(Cd)$ parameters. Arguably the strongest dependence on $\boldsymbol{\theta}$ is *direct* through the kernel matrices $\boldsymbol{K}^{(c)}$. We argue that freezing $\boldsymbol{A}$ during line searches can hinder performance significantly.

Finally, the reader may wonder why we do not choose a model selection criterion more "compatible" with the fact that site parameters are chosen using EP (or ADF) projections. In fact, the ADATAP variational free energy would probably fit in better and has been used for model selection in single process models in [2]. The problem is that even in the binary case, the ADATAP criterion is much more complicated to derive than the simple variational bound we use here. Furthermore, its gradient w.r.t. $\boldsymbol{\theta}$ is the same as ours if we fix I, $\boldsymbol{b}\,\boldsymbol{\Pi}$.

9.1 The Criterion and the Gradient

We minimize an upper bound to the negative log marginal likelihood given by

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

Here, the hyperparameter vector $\boldsymbol{\theta}$ decomposes into vectors $\boldsymbol{\theta}^{(c)}$ for each covariance function $K^{(c)}$ and the intercept vector $\boldsymbol{\beta} \in \mathbb{R}^C$.

It is possible to compute $\mathcal{G}$ and its gradient w.r.t. $\boldsymbol{\theta}$ using a procedure which requires $O(n\,C\,d^2)$ for precomputation (in addition to the conditional inference step), then $O(n\,d\,|\boldsymbol{\theta}|)$ for the gradient. The procedure does not need additional memory, but will overwrite the buffer for the $\boldsymbol{m}_j^{(c)}$ stubs.

Experiments with this model selection criterion are work in progress. The dominant computation for a criterion and gradient in "major mode" is the conditional inference step which runs much faster in "minor mode". Note that even though line searches tend to convergence quickly on average, most of the gradient computations are done using "minor mode". There are obvious ideas how to speed up the optimization, such as relying on the fact that the active set should not change very much over time. One could make local changes rather

[28] Slacks between corresponding upper and lower bounds are typically huge in high dimensions. Intuitively it seems clear that approximating a complicated function in high dimensions is much easier than bounding it.

than a complete re-selection in most "major mode" iterations. For example, the technique of exchange moves (see [10]) could be useful for this purpose. Exploring such possibilities is subject to future work.

10 Experiments

We present preliminary experiments on a dataset of $C = 5$ classes and $n = 800$ patterns. We extracted 200 patterns at random for each even-digit class from the training set of the MNIST handwritten digits database[29]. From these, we pick 200 at random as test set. All experiments below use the same training/test set split.

In these preliminary experiments, we did not address the model selection problem of adjusting the covariance function parameters or the intercept parameters $\boldsymbol{\beta}$. The latter were fixed to $\mathbf{0}$ since our dataset is fairly balanced. While we should (and will) use independent kernels $K^{(c)}$ for the different processes $u^{(c)}$, in these first runs we used a *shared* Radial Basis Functions (RBF) covariance function

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

We then ran our algorithm for a range of (w, v) parameters, fixing $d_{final} = 150$ (active set size) and $L = 25$ (liquid set size). In each EP phase, we cycle 2–3 times over the liquid patterns. We reserve the issue of automatic model selection by empirical Bayesian methods for future work, which could certainly handle the case of independently parameterized $K^{(c)}$ covariance functions.

Table 1 gives test set results for the final predictor ($|I| = d_{final}$). The figures are the test set error *err* and the predictive probability of the true label *lh*, we also provide the maximum and minimum entropy of the predictive distributions over the test set. While *err* is often of principal interest, *lh* shows the uncertainty for the true class. *maxent* and *minent* are less important, but give an idea about the range of uncertainties in the predictive distributions. The performance is satisfying and quite stable over a range of different (w, v) (worse results of err around 0.10 were obtained for much smaller w values).

Figures 1 and 2 show learning curves (growing active set size) for test set error and average test set likelihood for $v = 5, w = 50$ vs. $v = 5, w = 25$, Figures 3 and 4 show the same for $v = 1, w = 40$ vs. $v = 5, w = 20$. We observe that larger likelihoods are obtained for smaller values of w (which translates to larger length scales, i.e. processes whose paths are varying less rapidly). More interestingly, the likelihood is much more variable between different configurations than the test error which further motivates developing marginal likelihood maximization techniques for model selection.

It is also interesting to observe that while test error decays rapidly with growing active set size d, the growth of the test set likelihood is almost linear in d, giving empirical evidence what recent theoretical analyses suggest: the value of including more training points is not so much in decreasing test error ever further (if the selection of I is done appropriately), but in reducing the uncertainty in the predictions. However, more extensive experiments are required to support any such claims.

[29] Available online at *http://www.research.att.com/ ~yann/exdb/mnist/index.html*.

v	w	err	lh	minent	maxent
0.5	40	0.03	0.250980	1.577508	1.608262
2	40	0.03	0.329869	1.465741	1.603344
3	50	0.03	0.318090	1.475719	1.605220
5	40	0.03	0.378992	1.362901	1.598936
5	50	0.03	0.334296	1.439076	1.603916
0.5	50	0.035	0.234370	1.596795	1.608964
0.5	75	0.035	0.214800	1.604509	1.609399
1	50	0.035	0.259384	1.572066	1.608069
1	75	0.035	0.225446	1.594087	1.609345
2	30	0.035	0.366623	1.403297	1.597981
2	50	0.035	0.297387	1.493768	1.606813
3	30	0.035	0.402858	1.227390	1.595801
3	75	0.035	0.248621	1.573701	1.609153
5	25	0.035	0.458838	1.013761	1.586853
5	30	0.035	0.434059	1.194334	1.591372
5	75	0.035	0.254237	1.555642	1.608953
1	30	0.04	0.312231	1.497278	1.603975
1	40	0.04	0.284071	1.541226	1.607647
2	75	0.04	0.237900	1.574186	1.609215
3	25	0.04	0.427809	1.252592	1.589746
5	20	0.04	0.477860	1.069909	1.591897

Table 1: Results on 5-class toy dataset. *err*: Error on test set. *lh*: Avg. likelihood test set. *minent*: Smallest entropy pred. distribution. *maxent*: Largest entropy pred. distribution.

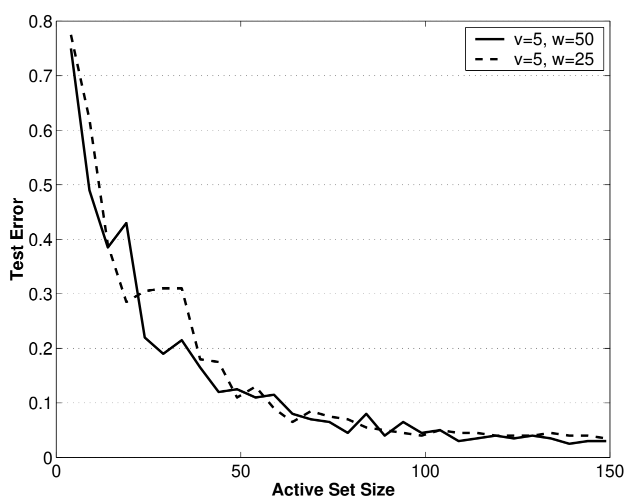

Figure 1: Active set size vs. error on test set, 5-class toy dataset.

11 Conclusions. Future Work

We have described 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

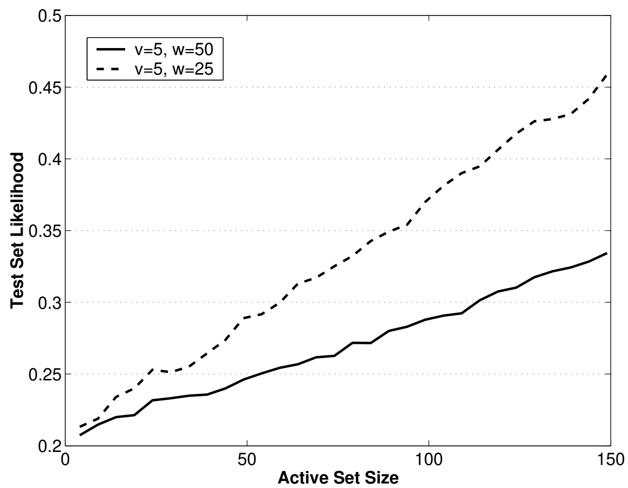

Figure 2: Active set size vs. average test set likelihood, 5-class toy dataset.

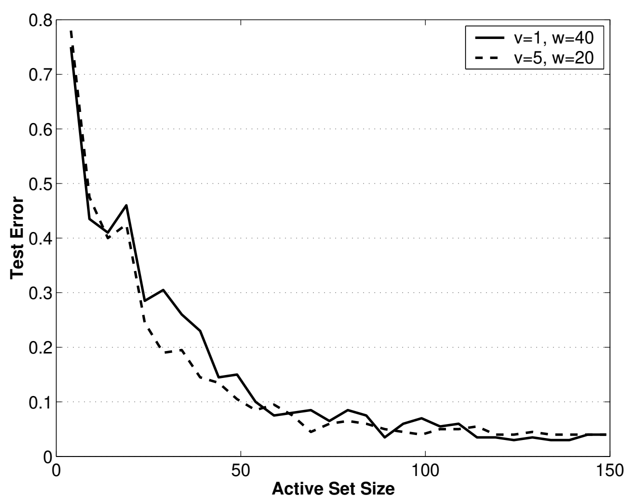

Figure 3: Active set size vs. error on test set, 5-class toy dataset.

number of classes. The central idea is to use an analogy to the Laplace approximation framework of [14] which allows for a $O(C)$ representation given independent priors, even though the posterior processes are strongly coupled. Note that this idea could be applied to other C-process likelihoods as well: namely, if the log likelihood is concave, its Hessian provides the value for $\mathbf{\Pi}_i$ under the Laplace approximation, and if this Hessian has a simple structure (implied by a "simple" coupling up to second order through the likelihood) an efficient $O(C)$ representation may arise.

Compared to previous kernel-based large margin multi-class algorithms, our scheme is somewhat harder to implement and may run slower on particular problems. It is also not the solution to a well-understood convex problem, but rather approximates a hard combinato-

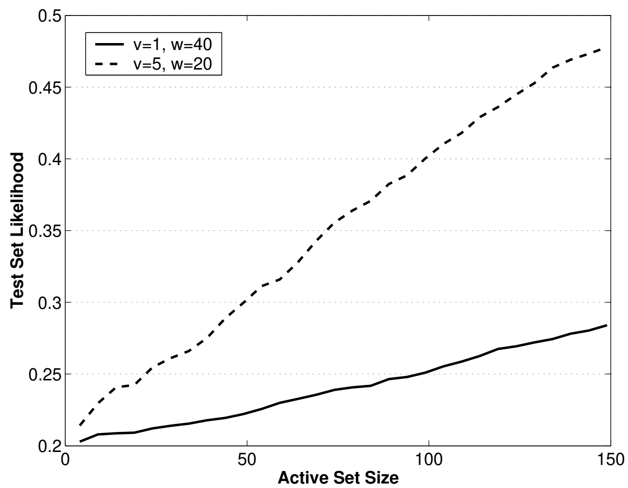

Figure 4: Active set size vs. average test set likelihood, 5-class toy dataset.

rial one. However, it offers a number of strong advantages. Being a Bayesian approximation, valid predictive probability ("error bars") estimates can be obtained and hyperparameters can be adjusted by empirical Bayesian methods. Our method operates jointly on the data from all classes and does not require imposing artificial binary partitions or combining binary discriminants in a posthoc heuristic fashion. The linear scaling in the number of training points is guaranteed *a priori*, as opposed to the degree of sparsity in SVM which depends in an unknown way on the data distribution and the kernel parameters. Previous work on Bayesian GP multi-class problems include [14, 4], but we are not aware of previous sparse approximations.

In future work, we will address the model selection problem by implementing marginal likelihood maximization. This works in the same way as for the binary IVM (see [10] for details), and the cost for computing the criterion and its gradient will be dominated by the conditional inference procedure described here. We have mentioned above that in order to cope with very large training sets, a randomized selection strategy together with intelligent cacheing of the stub vectors would be required.[30]

The present scheme does not generalize to a large number C of classes, namely there is a $O(d\,(C\,L)^3)$ scaling component and the numerical quadrature routines cannot be used anymore to perform ADF projections or compute predictive probabilities. We are also interested in using our methodology to address generalizations of larger structured graphical models (such as sequence models and conditional random fields) using nonparametric Gaussian process priors, which can be seen as classification problems with a very large but highly structured label space. However, such generalizations certainly require a satisfying probabilistic solution for the C-class model with moderate C.

[30] Recall that we are already forced to use some randomization in the selection, because no more than $\min\{n/C, n/L\}$ patterns can be scored for each iteration.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2002. Available online at `www.stanford.edu/~boyd/cvxbook.html`.

[2] Lehel Csató. *Gaussian Processes — Iterative Sparse Approximations*. PhD thesis, Aston University, Birmingham, UK, March 2002.

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

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

[5] Roger Horn and Charles Johnson. *Matrix Analysis*. Cambridge University Press, 1st edition, 1985.

[6] 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`.

[7] Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and applications to the classification of microarray data and satellite radiance data. Technical Report 1064, University of Wisconsin, September 2002.

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

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

[10] 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`.

[11] M. Seeger and M. I. Jordan. Fast sparse Gaussian process classification with multiple classes. Technical report, University of California at Berkeley, 2004. See `www.cs.berkeley.edu/~mseeger`. Submitted.

[12] Matthias Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2):1–38, 2004.

[13] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, London, 1998.

[14] 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.