Semiparametric Latent Factor Models

Matthias Seeger *mseeger@cs.berkeley.edu*
Yee-Whye Teh *ywteh@cs.berkeley.edu*
Michael I. Jordan *jordan@cs.berkeley.edu*
Division of Computer Science
University of California at Berkeley
Soda Hall, Berkeley, CA

March 2, 2005

Abstract

We propose a semiparametric model for regression and classification problems involving multiple response variables. The model makes use of a set of Gaussian processes to model the relationship to the inputs in a nonparametric fashion. Conditional dependencies between the responses can be captured through a linear mixture of the driving processes. This feature becomes important if some of the responses of predictive interest are less densely supplied by observed data than related auxiliary ones. We propose an efficient approximate inference scheme for this semiparametric model whose complexity is linear in the number of training data points.

1 Semiparametric Latent Factor Models

We are interested in predicting multiple responses $y_c \in \mathcal{Y}_c,\ c = 1, \ldots, C$ from covariates $\boldsymbol{x} \in \mathcal{X}$, and we would like to model the responses as conditionally dependent. In statistical terminology, we would like to "share statistical strength" between the y_c; in machine learning parlance this is often referred to as "transfer of learning." As we demonstrate empirically, such sharing can be especially powerful if the data for the responses is partially missing.

Models related to the one proposed here are used in geostatistics and spatial prediction under the name of *co-kringing* [4], and an example helps to demonstrate what we want to achieve with our technique. After an accidental uranium spill, a spatial map of uranium concentration is sought. We can take soil samples at locations of choice and measure their uranium content, then use kriging, Gaussian process regression or another spatial prediction technique to infer a map. However, carbon concentration is easier to measure than uranium so the space can be sampled more densely. Moreover, it is known that these two responses are often significantly correlated. In co-kriging we setup a joint spatial model for several responses with the aim of improving our prediction of one of them. The model to be described here can be used in the same context, but goes beyond simple co-kriging in several ways. First, by using latent random fields our model can represent *conditional* dependencies between responses directly. This is more flexible and expressive than schemes in which regression functions for each response are mixed in a posthoc manner [3], because the latent

fields are fitted using the data from all responses and can be used to model chacteristics of the dependencies rather than marginal relationships only. Second, the true nature of the dependencies does not have to be known in advance but can be learned from training data using empirical Bayesian techniques.

Writing $\boldsymbol{y} = (y_c)_c$ and introducing a latent variable $\boldsymbol{v} \in \mathbb{R}^C$, our model comes with a factorizing likelihood

$$P(\boldsymbol{y}|\boldsymbol{v}) = \prod_c P(y_c|v_c).$$

We intend to model the prior $P(\boldsymbol{v}|\boldsymbol{x})$ using Gaussian processes. The simplest possibility is to assume that the v_c are independent given $\boldsymbol{x}$, *i.e.* $P(\boldsymbol{v}|\boldsymbol{x}) = \prod_c P(v_c|\boldsymbol{x})$. In this case we can represent $P(v_c|\boldsymbol{x})$ as a Gaussian process (GP) with mean function 0 and covariance function $\tilde{K}^{(c)}$:

$$\mathrm{E}\left[v_c(\boldsymbol{x})v_{c'}(\boldsymbol{x}')\right] = \delta_{c,c'}\tilde{K}^{(c)}(\boldsymbol{x},\boldsymbol{x}').$$

This model will be called the *baseline model* in the sequel.

Note that under the baseline model, the inference and learning task simply decompose into C independent ones. The components of $\boldsymbol{v}$ are independent *a posteriori*, so even if there are dependencies in the data the prediction under the baseline model cannot profit from them. While this is often appropriate if the data is complete in all components y_c, it can behave suboptimal in situations where part of the y_c data is missing.

On the other end of the spectrum, we can model $P(\boldsymbol{v}|\boldsymbol{x})$ as a set of dependent Gaussian processes with $C(C+1)/2$ cross-covariance functions. Tasks such as inference, hyperparameter learning and prediction can be performed in much the same way as in a single process model. This model and related algorithms will be called the *naive method*. Approximate inference in nonparametric models typically scales superlinearly in the number of variables which can be dependent *a posteriori*. If n is the number of training datapoints, we have to deal with n variables at a time in the baseline method, but with as many as $C\,n$ in the naive one. The latter scaling is usually not acceptable for large $C\,n$.

In this paper we propose a model in which $\boldsymbol{v}|\boldsymbol{x}$ can be dependent in a flexible (and adaptive) way, yet inference and learning is more tractable than for the naive model. The key is to restrict the dependencies in a way which can be exploited in inference. We introduce a second latent variable $\boldsymbol{u} \in \mathbb{R}^P$. Here and in the following it is understood that for typical applications of our model we will have $P \ll C$. For a mixing matrix $\boldsymbol{\Phi} \in \mathbb{R}^{C,P}$ we set

$$\boldsymbol{v} = \boldsymbol{\Phi}\boldsymbol{u} + \boldsymbol{v}^{(0)}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}^{(0)}$ are independent. The components $v_c^{(0)}$ have independent GP priors with mean 0 and covariance function $\tilde{K}^{(c)}$, and the components u_p have independent zero-mean GP priors with kernel $K^{(p)}$. The baseline model is a special case ($P = 0$), but for $P > 0$ the v_c will be dependent *a posteriori*. Note that the dependencies themselves are represented by nonparametric latent random fields $\boldsymbol{u}$. We refer to this setup as *semiparametric latent factor model (SLFM)*, owing to the fact that the model combines nonparametric (the processes $\boldsymbol{u}$, $\boldsymbol{v}$) and parametric elements (the mixing matrix $\boldsymbol{\Phi}$|).

Note that by integrating out the $\boldsymbol{u}$ processes, we obtain induced cross-covariance functions for $\boldsymbol{x} \mapsto \boldsymbol{v}$:

$$\mathrm{E}[v_c(\boldsymbol{x})v_{c'}(\boldsymbol{x}')] = \delta_{c,c'}\tilde{K}^{(c)}(\boldsymbol{x},\boldsymbol{x}') + \sum_p \phi_{c,p}\phi_{c',p}K^{(p)}(\boldsymbol{x},\boldsymbol{x}').$$

We can therefore perform inference and prediction using the naive method. One goal of this paper is to exploit the structure in this particular setup in order to obtain a significantly more efficient method.

Suppose we observe some independently and identically distributed data $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \,|\, i = 1, \ldots, n\}$. We are interested in approximating the posterior $P(\boldsymbol{v}_*|\boldsymbol{x}_*, D)$. Let $\boldsymbol{v} = (v_{i,c})_{i,c}$, $v_{i,c} = v_c(\boldsymbol{x}_i)$. From the sampling model it is clear that for a test point $\boldsymbol{x}_*$, $\boldsymbol{v}_* = \boldsymbol{v}(\boldsymbol{x}_*)$ is independent of D given $\boldsymbol{v}$. Therefore,

$$P(\boldsymbol{v}_*|D) = \int P(\boldsymbol{v}_*|\boldsymbol{v}) P(\boldsymbol{v}|D)\, d\boldsymbol{v}$$

which can be computed straightforwardly if a Gaussian approximation to $P(\boldsymbol{v}|D)$ is known. In Section 2 we show how such an approximation can be represented and computed in an efficient way. Our framework can deal with missing values for $y_{i,c}$ effortlessly. While in the following we treat $\boldsymbol{y} = (y_{i,c})_{i,c}$ as completely given for notational convenience, incorporating missing values amounts to nothing more than reducing the dimensionality of the vector $\boldsymbol{y}$. In this case, we let n be the maximum number of $y_{i,c}$ given for any single class c. Variables $v_{i,c}$ corresponding to a missing $y_{i,c}$ do not have direct evidence associated with them, but are constrained through $\boldsymbol{u}$. If $\boldsymbol{y}_i = (y_{i,c})_c$ is missing completely, the corresponding datapoint can be removed.

2 Approximate Inference

In this section we show how a sparse approximation to the posterior can be represented and updated as we condition on evidence. We make use of the *informative vector machine (IVM)* [5, 8] framework which allows for inference approximations within time and memory requirements scaling only linearly in the number n of datapoints. The latter has been used successfully in the context of models with a single GP and its application to the baseline model of Section 1 is straightforward, but the application to a model with coupled processes is novel and of substantial additional complexity.

2.1 The Single Process IVM

We provide a brief introduction to the principles behind the single process IVM scheme applied to binary classification. Details can be found in [5, 8]. The IVM scheme has so far been applied to models featuring a single GP. As mentioned in Section 1, we replace the posterior $P(\boldsymbol{v}|D)$ by a Gaussian approximation $Q(\boldsymbol{v})$. In the single process case, $\boldsymbol{v} = (v_1, \ldots, v_n)$. Possible likelihoods include the Gaussian $P(y|v) = N(y|v, \sigma^2)$ for regression and the *probit* for classification:

$$P(y|v) = \Phi(y(v + \sigma)), \quad \Phi(x) = \int_{-\infty}^{x} N(t|0, 1)\, dt. \tag{1}$$

In the IVM scheme, we have

$$Q(\boldsymbol{v}) \propto P(\boldsymbol{v}) \exp\left(-\frac{1}{2}\boldsymbol{v}^T \boldsymbol{D}\boldsymbol{v} + \boldsymbol{b}^T\boldsymbol{v}\right), \quad \boldsymbol{D} = \mathrm{diag}(\pi_i) \tag{2}$$

where $P(\boldsymbol{v}) = N(\boldsymbol{0}, \tilde{\boldsymbol{K}})$ is the prior. The Gaussian term parameterized by the *site parameters* $\boldsymbol{b}$, $\boldsymbol{D}$ is called *likelihood approximation*. In general, we would allow $\pi_i > 0$ for all i, but this leads to costs of $O(n^3)$ for training and $O(n^2)$ for each prediction. In contrast, the IVM scheme is a *sparse approximation* in which we constrain $\pi_i = b_i = 0$ for all $i \notin I \subset \{1, \ldots, n\}$, $|I| = d \ll n$, which leads to $O(n\,d^2)$ training time and $O(d^2)$ cost for each prediction (in the IVM algorithm). The choice of the *active set* I is done sequentially using greedy forward selection with an information-theoretic criterion: among all remaining patterns outside the current I, we choose the one which maximizes the *(instantaneous) information gain* realized by including the pattern into I (note that this score is relative to the current approximation Q, which is why the scheme is sequential). In order to ease notation we write $\boldsymbol{I}_{\cdot,I}\boldsymbol{D}\boldsymbol{I}_{I,\cdot}$, $\boldsymbol{D} \in \mathbb{R}^{d,d}$ instead of $\boldsymbol{D}$ and $\boldsymbol{I}_{\cdot,I}\boldsymbol{b}$, $\boldsymbol{b} \in \mathbb{R}^d$ instead of $\boldsymbol{b}$.

We briefly remind the reader of the IVM representation which is derived in [5, 8]. Our ability to score *all* remaining patterns for each inclusion comes at a cost of $O(n\,d)$ memory and is the reason for the $O(n\,d^2)$ time scaling. If $Q(\boldsymbol{u}) = N(\boldsymbol{h}, \boldsymbol{A})$, then we see from Eq. 2 that

$$\boldsymbol{A} = \left(\tilde{\boldsymbol{K}}^{-1} + \boldsymbol{I}_{\cdot,I}\boldsymbol{D}\boldsymbol{I}_{I,\cdot}\right)^{-1} = \tilde{\boldsymbol{K}} - \boldsymbol{M}\boldsymbol{M}^T, \quad \boldsymbol{M} = \tilde{\boldsymbol{K}}_{\cdot,I}\boldsymbol{D}^{1/2}\boldsymbol{L}^{-T}, \quad \boldsymbol{I} + \boldsymbol{D}^{1/2}\tilde{\boldsymbol{K}}_I\boldsymbol{D}^{1/2} = \boldsymbol{L}\boldsymbol{L}^T, \tag{3}$$

furthermore $\boldsymbol{h} = \boldsymbol{M}\boldsymbol{\beta}$, $\boldsymbol{\beta} = \boldsymbol{L}^{-1}\boldsymbol{D}^{-1/2}\boldsymbol{b}$. Here, $\boldsymbol{L}$ is lower triangular with positive diagonal (*i.e.* a Cholesky factor), and the rows of $\boldsymbol{M} \in \mathbb{R}^{n,d}$ are called m-stubs. In order to score all remaining points, we require the posterior marginal means $\boldsymbol{h}$ and variances $\operatorname{diag}\boldsymbol{A}$. A scheme to update $\boldsymbol{L}$, $\boldsymbol{M}$, $\boldsymbol{\beta}$, $\boldsymbol{h}$ and $\operatorname{diag}\boldsymbol{A}$ after inclusion of a new point into I is given in [5, 8], the cost is $O(n\,d)$. The computation of the site parameters π_i, b_i for a pattern i to be included requires a so-called *ADF projection* (or *moment matching*) [7, 6] and typically a one-dimensional numerical quadrature if Gaussian expectations over the likelihood $P(y|u)$ are not analytically tractable. Importantly, these computations can be done to high accuracy with no significant additional cost, because they are one-dimensional (see Section 2.6).

We then make use of the conditional inference scheme as a subroutine in order to drive hyperparameter learning (the parameters of K and the intercept σ). This is done using a variational bound similar to what we employ in this paper here. A look at Eq. 3 reveals how to compute the latent predictive distribution $Q(u_*) = N(u_*|h_*, a_*)$, namely

$$h_* = \boldsymbol{m}_*^T\boldsymbol{\beta}, \quad a_* = \tilde{K}(\boldsymbol{x}_*, \boldsymbol{x}_*) - \|\boldsymbol{m}_*\|^2, \quad \boldsymbol{m}_* = \boldsymbol{L}^{-1}\boldsymbol{D}^{1/2}(\tilde{K}(\boldsymbol{x}_*, \boldsymbol{x}_i))_{i\in I},$$

and the predictive distribution is then obtained as $P(y_*|\boldsymbol{x}_*, D) = \mathrm{E}_Q[P(y_*|u_*)]$ which is computed by one-dimensional quadrature in the general case, or can be computed analytically for the probit likelihood.

2.2 Gaussian Process Belief Propagation

In this section we show how the IVM representation of Section 2.1 can be combined with the standard belief propagation algorithm for inference in parametric graphical models in order to obtain an efficient inference machinery for the SLFM.

If we simply apply the IVM technology to the naive method mentioned in Section 1, we have $C\,n$ variables of which we select an active set of size $C\,d$ (say), so that the running time complexity is $O(C^3\,n\,d^2)$, and the memory requirements are $O(C^2\,n\,d)$. This should be compared to $O(C\,n\,d^2)$ time and $O(C\,n\,d)$ memory required for the baseline model if we

select d active points for each c. The ratio of C^2 is due to the fact that the naive method does not exploit the structure in the effective $\boldsymbol{v}$ prior at all.

In parametric graphical models, conditional independence statements between variables of a domain are asserted. Under this model assumption, there are algorithms which can exploit certain Markovian aspects of this structure in order to perform inference very efficiently. A typical assumption for such models is that conditioned on all parameters (of the conditional distributions in the graph), data cases are independent. Strictly speaking there are two dimensions of conditional dependence in such models, the one between different variables of the domain (the *model dimension*), and the one between datapoints (the *data dimension*). For parametric models, the latter is usually trivial.[1] In contrast to that, in nonparametric models the data dimension has a very rich structure in that typically there is no finite number of parameters which render the data independent. This means that usually there are no sufficient statistics in which the data can be described in a subtantially compressed manner, implying the scaling with the number of training points. However, common nonparametric process models have so far almost exclusively been proposed for very simple model dimensions, by essentially focussing on a single real variable. In cases where this is not sufficient, strong assumptions such as complete posterior independence of all fields are used (leading to methods such as our baseline), or the variables are taken to be fully coupled (leading to the naive method).

In this paper we are interested in a model which has nontrivial structure along both dimensions. We propose to combine structured graphical models tools along the model dimension with sparse inference approximations along the data dimension in order to speed up the total inference and learning process.

Let us specify what we require the representation to deliver. Denote $\boldsymbol{v}_c = (v_{i,c})_i \in \mathbb{R}^n$. Within the IVM framework, active sets are selected based on current marginal posterior distributions. In order to generalize this to our model, the representation has to maintain posterior means and variances for all $v_{i,c}$. The key idea is to make use of the structure of the graphical model along the model dimensional, *i.e.* for $P(\boldsymbol{u}, \boldsymbol{v})$. If we treat $\boldsymbol{u}$ as a single variable, we have a tree-structured network (see Figure 1). This allows us to employ the (exact) *belief propagation (BP)* algorithm (CITE!!) in order to maintain marginals over the $\boldsymbol{v}_c$ as more and more evidence is accommodated.

Note that running BP efficiently on the network of Figure 1 is not straightforward due to the dimensionality of the variables involved (along the data dimension). We need to combine the basic message passing with the IVM framework in order to obtain an acceptable scaling.

Just as in the case of the single process IVM, we will perform inference sequentially, including a certain number of *active* patterns from the sample one at a time. The ability to maintain marginals of the $v_{i,c}$ at any time will be crucial to select which point to include next. The evidence potentials $\boldsymbol{v}_c \mapsto P(\boldsymbol{y}_c|\boldsymbol{v}_c)$ are non-Gaussian in general. We will replace them by low rank Gaussian factors in the same way as in the single process IVM (our notation is described in Appendix A):

$$\Psi_v(\boldsymbol{v}_c) = N^U\left(\boldsymbol{I}_{\cdot,I_c}\boldsymbol{b}^{(c)}, \boldsymbol{I}_{\cdot,I_c}\boldsymbol{D}^{(c)}\boldsymbol{I}_{I_c,\cdot}\right),$$

where $I_c \subset \{1,\ldots,n\}$ is the active set and $\boldsymbol{b}^{(c)}$, $\boldsymbol{D}^{(c)}$ the site parameters. Initially, $I_c = \emptyset$

[1] The graphical symbol for this conditional data independence is the *plate*.

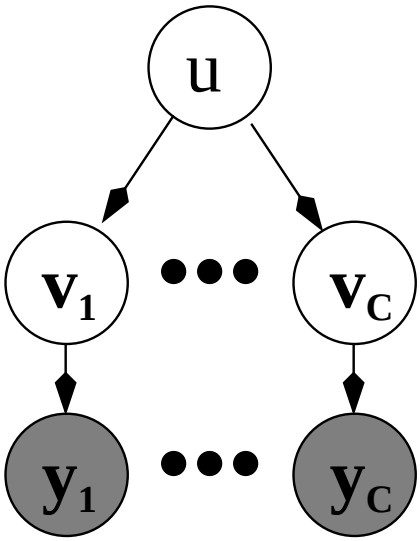

Figure 1: SLFM as a tree-structured graphical model

and $\Psi_v \equiv 1$. The edge potentials are

$$\Psi_{u\to v}(\boldsymbol{v}_c, \boldsymbol{u}) = P(\boldsymbol{v}_c|\boldsymbol{u}) = N\left((\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{u}, \tilde{\boldsymbol{K}}^{(c)}\right)$$

where $\boldsymbol{\phi}_c = \boldsymbol{\Phi}_{c,\cdot}^T$. Finally,

$$\Psi_u(\boldsymbol{u}) = P(\boldsymbol{u}) = N(\boldsymbol{0}, \boldsymbol{K})$$

where $\boldsymbol{K} = \mathrm{diag}(\boldsymbol{K}^{(p)})_p$. Now suppose new evidence is introduced in the sense that j is included into I_c with site parameters $b_{j,c}$, $\pi_{j,c}$. This will change the message $\boldsymbol{v}_c$ sends to $\boldsymbol{u}$ which is

$$m_{\boldsymbol{v}_c\to\boldsymbol{u}}(\boldsymbol{u}) \propto \int \Psi_v(\boldsymbol{v}_c)\Psi_{u\to v}(\boldsymbol{v}_c, \boldsymbol{u})\, d\boldsymbol{v}_c \tag{4}$$

which in turns modifies the messages $\boldsymbol{u}$ sends to $\boldsymbol{v}_{c'}$, $c' \neq c$:

$$m_{\boldsymbol{u}\to\boldsymbol{v}_{c'}}(\boldsymbol{v}_{c'}) \propto \int \prod_{c''\neq c'} m_{\boldsymbol{v}_{c''}\to\boldsymbol{u}}(\boldsymbol{u})\Psi_u(\boldsymbol{u})\Psi_{u\to v}(\boldsymbol{v}_{c'}, \boldsymbol{u})\, d\boldsymbol{u}. \tag{5}$$

The message $m_{\boldsymbol{u}\to\boldsymbol{v}_c}$ remains the same. Finally, all marginals have to be updated:

$$Q(\boldsymbol{v}_{c'}) \propto \Psi_v(\boldsymbol{v}_{c'}) m_{\boldsymbol{u}\to\boldsymbol{v}_{c'}}(\boldsymbol{v}_{c'}),$$

$Q(\boldsymbol{v}_c)$ because $\Psi_v(\boldsymbol{v}_c)$ changed, and $Q(\boldsymbol{v}_{c'})$ because $m_{\boldsymbol{u}\to\boldsymbol{v}_{c'}}$ changed, $c' \neq c$.

The details for the propagation scheme will be worked out in the following section. We will see that the key problem of applying BP to our nonparametric setup is that messages have to represented by a number of parameters which grows as new evidence is incorporated. This situation is very different from BP on a parametric graphical model where messages have a fixed size depending on the size of the sufficient statistics of potentials and the graph connectivity.

2.3 The Representation

In this section we work out the details for the propagation scheme introduced in Section 2.2 and give the representation for the posterior approximation Q.

We first note that a direct implementation of BP as described in the previous section does not significantly improve the scaling of the naive method mentioned there. The reason is that if the active sets I_c can be chosen independently, they may end up to be disjoint and have a combined size of about $C\,d$. Since each $v_{i,c}$ directly influences all variables $\boldsymbol{u}_i$, we basically need about $P\,C\,d$ of the components of $\boldsymbol{u}$ in order to represent each of the messages $m_{\boldsymbol{u}\to\boldsymbol{v}_c}$, and this leads to prohibitive costs. On the other hand, restricting all active sets to be the same would mean a drawback compared to the independent baseline.

We opt for an approximate inference method which limits the number of $\boldsymbol{u}$ components that the message $m_{\boldsymbol{v}_c\to\boldsymbol{u}}$ depends upon. To this end let $d_c = |I_c|$ denote the final active set sizes.[2] Let $d \leq \min\{d_c\}$ and $I \subset I_c$ for all c with $d = |I|$. In our inference approximation we will use the BP algorithm introducing evidence sequentially, but in a sense squeeze the messages from $\boldsymbol{v}_c$ to $\boldsymbol{u}$ through the bottleneck of $\boldsymbol{u}_I = (u_{i,p})_{i\in I,p} \in \mathbb{R}^{Pd}$. The common active set I is selected in the beginning, in what we call *common inclusion phase*. Once it has attained the final size d, we set $I_c = I$ for all c and continue to add elements to the I_c independently. Each inclusion is equivalent to introducing new evidence, but the corresponding message $m_{\boldsymbol{v}_c\to\boldsymbol{u}}$ is restricted not to depend on $\boldsymbol{u}_{I_c\setminus I}$. This idea will be made more precise below. We note that our inference approximation may not lead to a consistent joint posterior approximation $Q(\boldsymbol{v})$, but it does lead to joint marginal posteriors $Q(\boldsymbol{v}_c)$ over the different classes. This is not much different from some general extensions of BP on loopy networks where a consistent joint posterior approximation cannot be extracted.

The complete representation for Q is quite complicated and consists of a sequence of representations as used for the single process IVM. We denote the representations by $\mathcal{R}_1(c), \mathcal{R}_2(c), \mathcal{R}_3(c), \mathcal{R}_4$, $c = 1, \ldots, C$. We aim to use consistent notation which is the same as used in Section 2.1 or in [5, 8]. Namely, $\boldsymbol{L}$ will be lower-triangular Cholesky factors of matrices $\boldsymbol{A}$, $\boldsymbol{\beta}$ will be additional vectors of the form $\boldsymbol{\beta} = \boldsymbol{L}^{-1}\boldsymbol{c}$, and $\boldsymbol{M}$ will be matrices of the form $\boldsymbol{B}\boldsymbol{L}^{-T}$.

For notational simplicity below we will order the components in the vector $\boldsymbol{u}_I$ in a different way than usual (see Appendix A). Recall that for $\boldsymbol{v}, \boldsymbol{u}$, *etc.* we have that $\boldsymbol{v} = (v_{1,1}, \ldots, v_{n,1}, v_{1,2}, \ldots, v_{n,C})^T$. However for $\boldsymbol{u}_I$ we use the ordering $\boldsymbol{u}_I = (u_{i_1,1}, \ldots u_{i_1,P}, u_{i_2,1}, \ldots, u_{i_d,P})^T$ where $I = \{i_1, \ldots, i_d\}$. Let $\boldsymbol{\Pi}$ be the permutation matrix which converts from the latter to the standard ordering, so that $\boldsymbol{\Pi}\boldsymbol{u}_I$ is in standard ordering. Note that $\boldsymbol{\Pi}^T(\boldsymbol{\phi}_c \otimes \boldsymbol{I}) = (\boldsymbol{I} \otimes \boldsymbol{\phi}_c)$, so that $(\boldsymbol{\phi}_c \otimes \boldsymbol{I})$ in the standard ordering becomes $(\boldsymbol{I} \otimes \boldsymbol{\phi}_c)$ in the $\boldsymbol{u}_I$ ordering. If $\boldsymbol{K} = \mathrm{diag}(\boldsymbol{K}^{(p)})_p \in \mathbb{R}^{Pn,Pn}$ is the kernel matrix in the standard ordering, we define $\hat{\boldsymbol{K}} = \boldsymbol{\Pi}^T\boldsymbol{K}\boldsymbol{\Pi}$ which is the kernel matrix in the ordering used for $\boldsymbol{u}_I$. Note that although $\hat{\boldsymbol{K}}$ is just as sparse as $\boldsymbol{K}$ it does not have block-diagonal structure. In general, the superscript "$\hat{}$" indicates that the $\boldsymbol{u}_I$ ordering is used.

[2] For simplicity we denote both the current and final active set size by d_c. In complexity statements, d_c is the final size.

In order to represent $m_{\boldsymbol{v}_c \to \boldsymbol{u}}$ we need a IVM representation of size d_c:

$$\begin{aligned} \mathcal{R}_1(c): \quad \boldsymbol{L}^{(1,c)}\boldsymbol{L}^{(1,c)T} &= \boldsymbol{A}^{(1,c)} = \boldsymbol{I} + \boldsymbol{D}^{(c)1/2}\tilde{\boldsymbol{K}}^{(c)}_{I_c}\boldsymbol{D}^{(c)1/2}, \\ \boldsymbol{\beta}^{(1,c)} &= \boldsymbol{L}^{(1,c)-1}\boldsymbol{D}^{(c)-1/2}\boldsymbol{b}^{(c)}, \\ \boldsymbol{E}^{(c)} &= \boldsymbol{D}^{(c)1/2}\boldsymbol{L}^{(1,c)-T}. \end{aligned} \tag{6}$$

Suppose that we are still in the common inclusion phase, *i.e.* $I_c = I$. If $\boldsymbol{P}^{(c)} = (\boldsymbol{\phi}_c \otimes \boldsymbol{I})\boldsymbol{D}^{(c)1/2}\boldsymbol{L}^{(1,c)-T}$, it is shown in Appendix B.1 that $m_{\boldsymbol{v}_c \to \boldsymbol{u}}$ is given by $N^U(\boldsymbol{u}|\boldsymbol{I}_{\cdot,I}\boldsymbol{P}^{(c)}\boldsymbol{\beta}^{(1,c)}, \boldsymbol{I}_{\cdot,I}\boldsymbol{P}^{(c)}\boldsymbol{P}^{(c)T}\boldsymbol{I}_{I,\cdot})$, so the message depends on $\boldsymbol{u}_I$ only. The "bottleneck" approximation we are using ensures that $m_{\boldsymbol{v}_c \to \boldsymbol{u}}$ will always depend on $\boldsymbol{u}_I$ only, even if eventually $I_c \setminus I \neq \emptyset$. Namely, if

$$\hat{\boldsymbol{P}}^{(c)} = (\boldsymbol{I} \otimes \boldsymbol{\phi}_c)\boldsymbol{E}^{(c)}_{1\ldots d,\cdot} \in \mathbb{R}^{Pd,d_c} \tag{7}$$

(where we assume that I is a prefix of I_c), the message is defined to be

$$m_{\boldsymbol{v}_c \to \boldsymbol{u}}(\boldsymbol{u}_I) = N^U\left(\hat{\boldsymbol{P}}^{(c)}\boldsymbol{\beta}^{(1,c)}, \hat{\boldsymbol{P}}^{(c)}\hat{\boldsymbol{P}}^{(c)T}\right).$$

The representation $\mathcal{R}_2(c)$ is needed to form the message $m_{\boldsymbol{u} \to \boldsymbol{v}_c}$, it basically represents the distribution

$$R_c(\boldsymbol{u}) \propto P(\boldsymbol{u}) \prod_{c' \neq c} m_{\boldsymbol{v}_{c'} \to \boldsymbol{u}}(\boldsymbol{u}_I).$$

Because all $m_{\boldsymbol{v}_{c'} \to \boldsymbol{u}}$ are functions of $\boldsymbol{u}_I$ we have $R_c(\boldsymbol{u}) = R_c(\boldsymbol{u}_I)P(\boldsymbol{u} \setminus \boldsymbol{u}_I|\boldsymbol{u}_I)$. $\mathcal{R}_2(c)$ therefore needs to be of size $P\,d$ only, and its size grows only during the common inclusion phase. Its exact form is determined by what is required to finally compute the single marginals of $Q(\boldsymbol{v}_c)$. To this end, we need IVM representations $\mathcal{R}_3(c)$ of size d_c. The difference between $\mathcal{R}_1(c)$ and $\mathcal{R}_3(c)$ lies in the "prior distributions" used for the IVM representation. For $\mathcal{R}_1(c)$ this is $N(\boldsymbol{v}_c|\boldsymbol{0}, \tilde{\boldsymbol{K}}^{(c)})$ which does not depend on messages coming from $\boldsymbol{u}$. For $\mathcal{R}_3(c)$ this is replaced by the "effective prior" $N(\boldsymbol{v}_c|\boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)})$ whose parameters depend on the message $m_{\boldsymbol{u} \to \boldsymbol{v}_c}$, so that $\mathcal{R}_3(c)$ has to be modified whenever the message changes. Apart from that, the both representations share the same evidence potential $\Psi_v(\boldsymbol{v}_c)$ which is modified with each inclusion into I_c.

A glance at Eq. 5 reveals that we have

$$\boldsymbol{\Sigma}^{(c)} = \tilde{\boldsymbol{K}}^{(c)} + (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\mathrm{Var}_{R_c}[\boldsymbol{u}](\boldsymbol{\phi}_c \otimes \boldsymbol{I}). \tag{8}$$

By a standard formula,

$$\begin{aligned} \mathrm{Var}_{R_c}[\boldsymbol{u}] &= \mathrm{E}_{R_c}[\mathrm{Var}_P[\boldsymbol{u}|\boldsymbol{u}_I]] + \mathrm{Var}_{R_c}[\mathrm{E}_P[\boldsymbol{u}|\boldsymbol{u}_I]] \\ &= \boldsymbol{K} - \boldsymbol{K}_{\cdot,I}\left(\boldsymbol{K}_I^{-1} - \boldsymbol{K}_I^{-1}\boldsymbol{\Pi}\mathrm{Var}_{R_c}[\boldsymbol{u}_I]\boldsymbol{\Pi}^T\boldsymbol{K}_I^{-1}\right)\boldsymbol{K}_{I,\cdot} \end{aligned}$$

where we used that $R_c(\boldsymbol{u}|\boldsymbol{u}_I) = P(\boldsymbol{u}|\boldsymbol{u}_I)$. Next, let $d_{\setminus c} = \sum_{c' \neq c} d_c$ and

$$\begin{aligned} \hat{\boldsymbol{P}}^{(\setminus c)} &= \left(\hat{\boldsymbol{P}}^{(1)} \ldots \hat{\boldsymbol{P}}^{(c-1)}\, \hat{\boldsymbol{P}}^{(c+1)} \ldots \hat{\boldsymbol{P}}^{(C)}\right) \in \mathbb{R}^{Pd,d_{\setminus c}}, \\ \hat{\boldsymbol{\beta}}^{(\setminus c)} &= \left(\boldsymbol{\beta}^{(1,1)T} \ldots \boldsymbol{\beta}^{(1,c-1)T}\, \boldsymbol{\beta}^{(1,c+1)T} \ldots \boldsymbol{\beta}^{(1,C)T}\right)^T \in \mathbb{R}^{d_{\setminus c}}. \end{aligned}$$

The order of the columns of $\hat{\boldsymbol{P}}^{(\backslash c)}$ is not important as long as $\hat{\boldsymbol{\beta}}^{(\backslash c)}$ follows the same ordering. Recall that $\hat{\boldsymbol{K}}_I = \boldsymbol{\Pi}^T \boldsymbol{K}_I \boldsymbol{\Pi}$. We have

$$\mathrm{Var}_{R_c}[\boldsymbol{u}_I] = \left(\hat{\boldsymbol{K}}_I^{-1} + \hat{\boldsymbol{P}}^{(\backslash c)} \hat{\boldsymbol{P}}^{(\backslash c)T}\right)^{-1}, \quad \mathrm{E}_{R_c}[\boldsymbol{u}_I] = \mathrm{Var}_{R_c}[\boldsymbol{u}_I] \hat{\boldsymbol{P}}^{(\backslash c)} \hat{\boldsymbol{\beta}}^{(\backslash c)}.$$

Furthermore,

$$\boldsymbol{K}_I^{-1} - \boldsymbol{K}_I^{-1} \boldsymbol{\Pi} \mathrm{Var}_{R_c}[\boldsymbol{u}_I] \boldsymbol{\Pi}^T \boldsymbol{K}_I^{-1} = \boldsymbol{K}_I^{-1} - \boldsymbol{\Pi} \left(\hat{\boldsymbol{K}}_I + \hat{\boldsymbol{K}}_I \hat{\boldsymbol{P}}^{(\backslash c)} \hat{\boldsymbol{P}}^{(\backslash c)T} \hat{\boldsymbol{K}}_I\right)^{-1} \boldsymbol{\Pi}^T.$$

Plugging this into Eq. 8 we have

$$\boldsymbol{\Sigma}^{(c)} = \tilde{\boldsymbol{K}}^{(c)} + (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I}) \boldsymbol{K} (\boldsymbol{\phi}_c \otimes \boldsymbol{I}) - (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I}) \boldsymbol{M}^{(4)} \boldsymbol{M}^{(4)T} (\boldsymbol{\phi}_c \otimes \boldsymbol{I}) + \boldsymbol{M}^{(2,c)} \boldsymbol{M}^{(2,c)T} \quad (9)$$

where $\boldsymbol{M}^{(4)}$ does not depend on c and actually has a simple blockdiagonal structure. The role of $\mathcal{R}_2(c)$ is the maintenance of $\boldsymbol{M}^{(2,c)}$ (see Eq. 11). We also have

$$\boldsymbol{\mu}^{(c)} = (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I}) \mathrm{E}_{R_c}[\mathrm{E}_P[\boldsymbol{u}|\boldsymbol{u}_I]] = (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I}) \boldsymbol{K}_{\cdot,I} \boldsymbol{K}_I^{-1} \boldsymbol{\Pi} \left(\hat{\boldsymbol{K}}_I^{-1} + \hat{\boldsymbol{P}}^{(\backslash c)} \hat{\boldsymbol{P}}^{(\backslash c)T}\right)^{-1} \hat{\boldsymbol{P}}^{(\backslash c)} \hat{\boldsymbol{\beta}}^{(\backslash c)}. \quad (10)$$

Let

$$\begin{aligned} \mathcal{R}_2(c): \quad \boldsymbol{L}^{(2,c)} \boldsymbol{L}^{(2,c)T} &= \boldsymbol{A}^{(2,c)} = \hat{\boldsymbol{K}}_I + \hat{\boldsymbol{K}}_I \hat{\boldsymbol{P}}^{(\backslash c)} \hat{\boldsymbol{P}}^{(\backslash c)T} \hat{\boldsymbol{K}}_I, \\ \boldsymbol{\beta}^{(2,c)} &= \boldsymbol{L}^{(2,c)-1} \hat{\boldsymbol{K}}_I \hat{\boldsymbol{P}}^{(\backslash c)} \hat{\boldsymbol{\beta}}^{(\backslash c)}, \\ \boldsymbol{M}^{(2,c)} &= (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I}) \boldsymbol{K}_{\cdot,I} \boldsymbol{\Pi} \boldsymbol{L}^{(2,c)-T} \in \mathbb{R}^{n,Pd} \end{aligned} \quad (11)$$

This representation is of size $O(n\,P\,d)$. From Eq. 10 it is easy to see that

$$\boldsymbol{\mu}^{(c)} = \boldsymbol{M}^{(2,c)} \boldsymbol{\beta}^{(2,c)}. \quad (12)$$

$\mathcal{R}_4$ is required to maintain $\boldsymbol{M}^{(4)}$ which is needed in Eq. 9:

$$\mathcal{R}_4: \quad \boldsymbol{L}^{(4)} \boldsymbol{L}^{(4)T} = \boldsymbol{K}_I, \quad \boldsymbol{M}^{(4)} = \boldsymbol{K}_{\cdot,I} \boldsymbol{L}^{(4)-T}. \quad (13)$$

Since all matrices here are blockdiagonal, the representation size is only $O(n\,P\,d)$. $\boldsymbol{K}_I^{(p)}$ may be ill-conditioned as d gets large, so we use the common remedy of replacing $\boldsymbol{K}_I^{(p)}$ by $\boldsymbol{K}_I^{(p)} + \varepsilon \boldsymbol{I}$ for some small $\varepsilon > 0$.

Finally, $\mathcal{R}_3(c)$ is a normal IVM representation based on the "effective prior" $N(\boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)})$:

$$\begin{aligned} \mathcal{R}_3(c): \quad \boldsymbol{L}^{(3,c)} \boldsymbol{L}^{(3,c)T} &= \boldsymbol{A}^{(3,c)} = \boldsymbol{I} + \boldsymbol{D}^{(c)1/2} \boldsymbol{\Sigma}_{I_c}^{(c)} \boldsymbol{D}^{(c)1/2}, \\ \boldsymbol{M}^{(3,c)} &= \boldsymbol{\Sigma}_{\cdot,I_c}^{(c)} \boldsymbol{D}^{(c)1/2} \boldsymbol{L}^{(3,c)-T} \in \mathbb{R}^{n,d_c}, \\ \boldsymbol{\beta}^{(3,c)} &= \boldsymbol{L}^{(3,c)-1} \left(\boldsymbol{D}^{(c)-1/2} \boldsymbol{b}^{(c)} - \boldsymbol{D}^{(c)1/2} \boldsymbol{\mu}_{I_c}^{(c)}\right). \end{aligned} \quad (14)$$

The size is $O(n \sum_c d_c)$ for all $\mathcal{R}_3(c)$. We also maintain $\boldsymbol{\mu}^{(c)}$ and $\mathrm{diag}\,\boldsymbol{\Sigma}^{(c)}$ explicitly in $\mathcal{R}_3(c)$. It easy to see that the Gaussian posterior $Q(\boldsymbol{v}_c)$ is given by

$$\mathrm{E}_Q[\boldsymbol{v}_c] = \boldsymbol{\mu}^{(c)} + \boldsymbol{M}^{(3,c)} \boldsymbol{\beta}^{(3,c)}, \quad \mathrm{Var}_Q[\boldsymbol{v}_c] = \boldsymbol{\Sigma}^{(c)} - \boldsymbol{M}^{(3,c)} \boldsymbol{M}^{(3,c)T}.$$

Both $\boldsymbol{h}^{(c)} = \mathrm{E}_Q[\boldsymbol{v}_c]$ and $\boldsymbol{a}^{(c)} = \mathrm{diag}\,\mathrm{Var}_Q[\boldsymbol{v}_c]$ are maintained explicitly with $\mathcal{R}_3(c)$ (their maintenance is in fact the prime reason for all of the representations). The size of the combined representation is $O(n\,(\sum_c d_c + d\,C\,P))$. This should be compared to $O(n \sum_c d_c)$ for the baseline method and to $O(n\,C \sum_c d_c)$ for the naive method.

2.4 Primitives

The update of the representation after the inclusion of a point into I is most easily described in terms of some computation primitives which are required in the context of low-rank updates of matrices. We assume in general that

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T \in \mathbb{R}^{p,p}, \quad \boldsymbol{M} = \boldsymbol{B}\boldsymbol{L}^{-T} \in \mathbb{R}^{q,p},$$

where $\boldsymbol{A}$ is symmetric positive definite. The primitives update $\boldsymbol{L} \to \boldsymbol{L}'$, $\boldsymbol{M} \to \boldsymbol{M}'$ after certain modifications $\boldsymbol{A} \to \boldsymbol{A}'$, $\boldsymbol{B} \to \boldsymbol{B}'$.

2.4.1 Primitive cholext

cholext is used if $\boldsymbol{A}$ grows by a number of rows/columns. Namely,

$$\boldsymbol{A}' = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{A}_{\cdot,*} \\ \boldsymbol{A}_{\cdot,*}^T & \boldsymbol{A}_* \end{pmatrix}, \quad \boldsymbol{B}' = (\boldsymbol{B}\, \boldsymbol{B}_*), \quad \boldsymbol{A}_{\cdot,*} \in \mathbb{R}^{p,r}.$$

cholext$(\boldsymbol{L}, \boldsymbol{M}, \boldsymbol{A}_{\cdot,*}, \boldsymbol{A}_*, \boldsymbol{B}_*)$ computes

$$\boldsymbol{L}' = \begin{pmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{L}_{\cdot,*}^T & \boldsymbol{L}_* \end{pmatrix}, \quad \boldsymbol{M}' = (\boldsymbol{M}\, \boldsymbol{M}_*)$$

as

$$\boldsymbol{L}_{\cdot,*} = \boldsymbol{L}^{-1}\boldsymbol{A}_{\cdot,*}, \quad \boldsymbol{L}_*\boldsymbol{L}_*^T = \boldsymbol{A}_* - \boldsymbol{L}_{\cdot,*}^T\boldsymbol{L}_{\cdot,*}, \quad \boldsymbol{M}_* = (\boldsymbol{B}_* - \boldsymbol{M}\,\boldsymbol{L}_{\cdot,*})\,\boldsymbol{L}_*^{-T}.$$

The complexity is $O(p^2 r + qpr)$ if $r < p$.

2.4.2 Primitive chollrup

Here, $\boldsymbol{A}' = \boldsymbol{A} + s\boldsymbol{V}\boldsymbol{V}^T$, $s \in \{-1, +1\}$, $\boldsymbol{V} \in \mathbb{R}^{p,r}$. Let $\boldsymbol{P} = \boldsymbol{L}^{-1}\boldsymbol{V}$ which has to be computed if not given. We can call chollrup$(\boldsymbol{L}, \boldsymbol{M}, \boldsymbol{V}, s, false)$ or chollrup$(\boldsymbol{L}, \boldsymbol{M}, \boldsymbol{P}, s, true)$. An algorithm for *chollrup* can be found in [8] or [9]. The version for $s = -1$ is numerically less stable for a given condition number of $\boldsymbol{L}$, so if positive and negative low-rank updates have to be done, it is recommended to do all positive ones first. The complexity is $O(p^2 r)$ for the computation of $\boldsymbol{P}$ and $O(p^2 r + pqr)$ for the rest (if $r < p$). We refer to $\boldsymbol{M} \to \boldsymbol{M}'$ as $\boldsymbol{M}$ being *dragged along*.

2.5 Update of the Representation

The representation has to be updated after new inclusions are made to the I_c. This changes the evidence potentials and therefore some of the messages. The process is different during the common inclusion phase and afterwards. The former is more complicated and will be discussed here, for the latter we comment on how it differs from the former.

During the common inclusion phase, an elementary step consists of the inclusion of j into I with corresponding new site parameters $(b_{j,c})_c$, $(\pi_{j,c})_c$. We will comment in REF!! on how j and the new parameters are determined. We make use of the following conventions. If $\boldsymbol{x}$ is a quantity before the update, $\boldsymbol{x}'$ denotes its value after the update. If the update proceeds in more than one step, $\boldsymbol{x}'$ becomes $\boldsymbol{x}$ before the second step. For example, $I' = I \cup \{j\}$.

2.5.1 Update of $\mathcal{R}_1(c)$ and $\mathcal{R}_4$

First, $\mathcal{R}_1(c)$ are updated just like in the single process IVM:

$$\text{cholext}\left(\boldsymbol{L}^{(1,c)}, \boldsymbol{\beta}^{(1,c)T}, \pi_{j,c}^{1/2}\boldsymbol{D}^{(c)1/2}\tilde{\boldsymbol{K}}_{I,j}^{(c)}, 1+\pi_{j,c}\tilde{\boldsymbol{K}}_{j}^{(c)}, \pi_{j,c}^{-1/2}b_{j,c}\right).$$

Note that $d = d_c$, and that $\boldsymbol{D}^{(c)1/2}\boldsymbol{L}^{(1,c)-T}$ is upper triangular. Let $\boldsymbol{e}^{(c)} \in \mathbb{R}^{d+1}$ be the new column of $\boldsymbol{E}^{(c)}$, *i.e.*

$$\boldsymbol{e}^{(c)} = \left(\left(-l^{-1}\boldsymbol{E}^{(c)}\boldsymbol{l}\right)^T \; \pi_{j,c}/l\right)^T$$

where $(\boldsymbol{l}^T \, l)$ is the new row of $\boldsymbol{L}^{(1,c)}$. The new $\hat{\boldsymbol{P}}^{(c)\prime}$ is obtained by appending $\boldsymbol{0} \in \mathbb{R}^{P,d}$ to the bottom of $\hat{\boldsymbol{P}}^{(c)}$, then the new column $(\boldsymbol{I} \otimes \boldsymbol{\phi}_c)\boldsymbol{e}^{(c)}$ to the right.[3] The update cost is $O(\sum_c d_c) = O(C\,d)$.

Next we update $\mathcal{R}_4$. Since $\boldsymbol{L}^{(4)} = \text{diag}(\boldsymbol{L}_p^{(4)})_p$, $\boldsymbol{M}^{(4)} = \text{diag}(\boldsymbol{M}_p^{(4)})_p$, the update consists of

$$\text{cholext}\left(\boldsymbol{L}_p^{(4)}, \boldsymbol{M}_p^{(4)}, \boldsymbol{K}_{I,j}^{(p)}, \boldsymbol{K}_{j}^{(p)}, \boldsymbol{K}_{\cdot,j}^{(p)}\right), \; p = 1, \ldots, P$$

(recall that we implicitly add a small $\varepsilon > 0$ to the kernel diagonal). The cost is $O(n\,P\,d)$.

2.5.2 Update of $\mathcal{R}_2(c)$, $\mathcal{R}_3(c)$ (first part)

Next we need to update $\mathcal{R}_2(c)$, $\mathcal{R}_3(c)$ to account for the change in $m_{\boldsymbol{u}\to\boldsymbol{v}_c}$. We do this in two steps. First, we make a low-rank adjustment to $\boldsymbol{A}^{(2,c)}$ (a *chollrup* to $\mathcal{R}_2(c)$), second we extend $\boldsymbol{A}^{(2,c)}$ by a row/column (a *cholext* to $\mathcal{R}_2(c)$). Both result in low-rank adjustments to $\mathcal{R}_3(c)$ which are done directly after the changes to $\mathcal{R}_2(c)$. The first step can be described as follows. Let

$$\boldsymbol{E}^{(\backslash c)} = \left((\boldsymbol{I} \otimes \boldsymbol{\phi}_{c'})\boldsymbol{e}^{(c')}\right)_{c' \neq c} \in \mathbb{R}^{P(d+1),C-1}.$$

$(\hat{\boldsymbol{P}}^{(\backslash c)})'$ is obtained from $\hat{\boldsymbol{P}}^{(\backslash c)}$ by appending $\boldsymbol{0} \in \mathbb{R}^{P,d(C-1)}$ to the bottom, then $\boldsymbol{E}^{(\backslash c)}$ to the right. Also, let $(\hat{\boldsymbol{\beta}}^{(\backslash c)})' = (\hat{\boldsymbol{\beta}}^{(\backslash c)T}\boldsymbol{\beta}_*^T)^T$ with $\boldsymbol{\beta}_* \in \mathbb{R}^{C-1}$. Now,

$$\left(\boldsymbol{A}^{(2,c)\prime}\right)_{1\ldots d} = \boldsymbol{A}^{(2,c)} + \boldsymbol{V}\boldsymbol{V}^T, \quad \boldsymbol{V} = \hat{\boldsymbol{K}}_{I,I'}\boldsymbol{E}^{(\backslash c)} \in \mathbb{R}^{Pd,C-1}.$$

$(\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)})'$ is obtained by appending $\boldsymbol{0} \in \mathbb{R}^P$ to $\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)}$, then adding $\boldsymbol{E}^{(\backslash c)}\boldsymbol{\beta}_*$. Therefore

$$\hat{\boldsymbol{K}}_{I,I'}\left(\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)}\right)' = \hat{\boldsymbol{K}}_{I}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)} + \hat{\boldsymbol{K}}_{I,I'}\boldsymbol{E}^{(\backslash c)}\boldsymbol{\beta}_*.$$

Let

$$\boldsymbol{W} = \boldsymbol{L}^{(2,c)-1}\boldsymbol{V}, \quad \boldsymbol{w} = \boldsymbol{L}^{(2,c)-1}\boldsymbol{K}_{I,I'}\boldsymbol{E}^{(\backslash c)}\boldsymbol{\beta}_* = \boldsymbol{W}\boldsymbol{\beta}_*.$$

Then, $\mathcal{R}_2(c)$ is updated as

$$\text{chollrup}\left(\boldsymbol{L}^{(2,c)}, \left(\boldsymbol{M}^{(2,c)T}, \boldsymbol{\beta}^{(2,c)} + \boldsymbol{w}\right)^T, \boldsymbol{W}, +1, true\right).$$

[3] We do not explicitly indicate the dimensionality of $\boldsymbol{I}$ within each $(\boldsymbol{\phi}_c \otimes \boldsymbol{I})$ or $(\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})$, it is always clear from the context (here it is $d_c + 1$).

The cost is $O(n\,P\,C^2\,d)$ for all $\mathcal{R}_2(c)$. Note that some quantities required for the subsequent update of $\mathcal{R}_3(c)$ have to be computed before $\boldsymbol{M}^{(2,c)}$ is modified (notably $\tilde{\boldsymbol{M}}$).

To update $\mathcal{R}_3(c)$ accordingly, we first need to work out what changes the update of $\mathcal{R}_2(c)$ implies for $\boldsymbol{\Sigma}^{(c)}$, $\boldsymbol{\mu}^{(c)}$.[4] This is done in Appendix B.2. If we let

$$\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{I} + \boldsymbol{W}^T\boldsymbol{W},$$

the result is

$$\begin{aligned}
(\boldsymbol{\Sigma}^{(c)})' &= \boldsymbol{\Sigma}^{(c)} - \tilde{\boldsymbol{M}}\tilde{\boldsymbol{M}}^T, \quad \tilde{\boldsymbol{M}} = \boldsymbol{M}^{(2,c)}\boldsymbol{W}\,\boldsymbol{L}^{-T},\\
(\boldsymbol{\mu}^{(c)})' &= \boldsymbol{\mu}^{(c)} + \boldsymbol{\mu}_*^{(c)}, \quad \boldsymbol{\mu}_*^{(c)} = \tilde{\boldsymbol{M}}\,\boldsymbol{L}^{-1}\left(\boldsymbol{\beta}_* - \boldsymbol{W}^T\boldsymbol{\beta}^{(2,c)}\right).
\end{aligned}$$

We update $\operatorname{diag}\boldsymbol{\Sigma}^{(c)}$ by subtracting $\operatorname{diag}(\tilde{\boldsymbol{M}}\tilde{\boldsymbol{M}}^T)$. If $\boldsymbol{Q} = \boldsymbol{L}^{(3,c)-1}\boldsymbol{D}^{(c)1/2}\tilde{\boldsymbol{M}}_{I_c,\cdot} \in \mathbb{R}^{d_c,C-1}$, then $\mathcal{R}_3(c)$ is updated as

$$\text{chollrup}\left(\boldsymbol{L}^{(3,c)}, \left(\left(\boldsymbol{M}^{(3,c)} - \tilde{\boldsymbol{M}}\boldsymbol{Q}^T\right)^T, \left(\boldsymbol{\beta}^{(3,c)} - \boldsymbol{L}^{(3,c)-1}\boldsymbol{D}^{(c)1/2}(\boldsymbol{\mu}_*^{(c)})_{I_c}\right)\right)^T, \boldsymbol{Q}, -1, \mathit{true}\right).$$

It is clear that we have to compute $\tilde{\boldsymbol{M}}$ explicitly in order to obtain $\tilde{\boldsymbol{M}}\boldsymbol{Q}^T$ efficiently. The computation of $\tilde{\boldsymbol{M}}$ is $O(n\,C\,P\,d)$, so the cost for each $\mathcal{R}_3(c)$ is $O(n\,C\,(d_c + P\,d))$, the total cost adds up to $O(n\,C(C\,P\,d + \sum_c d_c))$ which is $O(n\,C^2\,P\,d)$ during the common inclusion phase.

2.5.3 Update of $\mathcal{R}_2(c)$, $\mathcal{R}_3(c)$ (second part)

Next we extend $\boldsymbol{A}^{(2,c)}$ by P new rows/columns. Let $\boldsymbol{B}^{(\backslash c)} = \hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{P}}^{(\backslash c)T}$ which is updated by padding with zeros at the bottom and right, then adding

$$\boldsymbol{B}_*^{(\backslash c)} = \boldsymbol{E}^{(\backslash c)}\boldsymbol{E}^{(\backslash c)T}.$$

Therefore,

$$\left(\boldsymbol{A}^{(2,c)\prime}\right)_{\cdot,d+1} = \hat{\boldsymbol{K}}_{I',j} + \hat{\boldsymbol{K}}_{I',I}\boldsymbol{B}^{(\backslash c)}\hat{\boldsymbol{K}}_{I,j} + \hat{\boldsymbol{K}}_{I',I'}\boldsymbol{B}_*^{(\backslash c)}\hat{\boldsymbol{K}}_{I',j}$$

(again, $\varepsilon > 0$ needs to be added to the kernel diagonal). We can use that $\boldsymbol{K}_{J,I}\boldsymbol{\Pi}(\boldsymbol{I}\otimes\boldsymbol{\phi}_c) = (\phi_{c,p}\boldsymbol{K}_{J,I}^{(p)})_p$. The update of $\mathcal{R}_2(c)$ is

$$\begin{aligned}
\text{cholext}\Bigg(\boldsymbol{L}^{(2,c)}, \left(\boldsymbol{M}^{(2,c)T}, \boldsymbol{\beta}^{(2,c)}\right)^T, (\boldsymbol{A}^{(2,c)\prime})_{1\ldots d,d+1}, (\boldsymbol{A}^{(2,c)\prime})_{d+1},\\
\left(\left((\boldsymbol{\phi}_c^T\otimes\boldsymbol{I})\boldsymbol{K}_{\cdot,j}\right)^T, \hat{\boldsymbol{K}}_{j,I'}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)}\right)^T\Bigg)
\end{aligned}$$

which costs $O(n\,P^2\,d)$ (recall that the variables $\boldsymbol{L}^{(2,c)}$, $\boldsymbol{M}^{(2,c)}$, $\boldsymbol{P}^{(\backslash c)}$, *etc.* denote the values after the first update step has been done, *not* the initial values at the beginning of the update).

[4] We do not yet accommodate the update of $\mathcal{R}_4$, this is done in the second step.

Let $\boldsymbol{M}_* \in \mathbb{R}^{n,P}$ denote the new columns of $\boldsymbol{M}^{(2,c)}$, $\boldsymbol{\beta}_* \in \mathbb{R}^P$ the new entries of $\boldsymbol{\beta}^{(2,c)}$, and $\boldsymbol{M}_{**} \in \mathbb{R}^{nP,P}$ the new columns of $\boldsymbol{M}^{(4)}$ (the latter does not depend on c and is block-diagonal). Then,

$$(\boldsymbol{\Sigma}^{(c)})' = \boldsymbol{\Sigma}^{(c)} - (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{M}_{**}\boldsymbol{M}_{**}^T(\boldsymbol{\phi}_c \otimes \boldsymbol{I}) + \boldsymbol{M}_*\boldsymbol{M}_*^T \tag{15}$$

which indicates how $\operatorname{diag} \boldsymbol{\Sigma}^{(c)}$ is updated. Furthermore,

$$(\boldsymbol{\mu}^{(c)})' = \boldsymbol{\mu}^{(c)} + \boldsymbol{M}_*\boldsymbol{\beta}_*.$$

In order to update $\mathcal{R}_3(c)$ we need two rank-P *chollrup* calls, we do the positive one first. The derivation is given in Appendix B.3. Let $\boldsymbol{Q} = \boldsymbol{L}^{(3,c)-1}\boldsymbol{D}^{(c)1/2}(\boldsymbol{M}_*)_{I_c,\cdot} \in \mathbb{R}^{d_c,P}$, then

$$\operatorname{chollrup}\left(\boldsymbol{L}^{(3,c)}, \left(\left(\boldsymbol{M}^{(3,c)} + \boldsymbol{M}_*\boldsymbol{Q}^T\right)^T, \boldsymbol{\beta}^{(3,c)} - \boldsymbol{Q}\boldsymbol{\beta}_*\right)^T, \boldsymbol{Q}, +1, true\right).$$

Next, with $\boldsymbol{Q} = \boldsymbol{L}^{(3,c)-1}\boldsymbol{D}^{(c)1/2}(\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})(\boldsymbol{M}_{**})_{I_c,\cdot} \in \mathbb{R}^{d_c,P}$ we have

$$\operatorname{chollrup}\left(\boldsymbol{L}^{(3,c)}, \left(\left(\boldsymbol{M}^{(3,c)} - (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{M}_{**}\boldsymbol{Q}^T\right)^T, \boldsymbol{\beta}^{(3,c)}\right)^T, \boldsymbol{Q}, -1, true\right).$$

Both cost $O(n\,P\,\sum_c d_c)$ in total (for all c).

2.5.4 Update of $\mathcal{R}_3(c)$ and marginals

Finally, we update $\mathcal{R}_3(c)$ to incorporate the new evidence (everything so far has only been done to update its "effective prior"). This is a standard IVM update, we only have to evaluate a new column of $\boldsymbol{\Sigma}^{(c)}$:

$$\boldsymbol{\Sigma}^{(c)}_{\cdot,j} = \tilde{\boldsymbol{K}}^{(c)}_{\cdot,j} + (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{K}_{\cdot,j}\boldsymbol{\phi}_c - (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{M}^{(4)}\left(\boldsymbol{M}^{(4)}_{j,\cdot}\right)^T\boldsymbol{\phi}_c + \boldsymbol{M}^{(2,c)}\boldsymbol{M}^{(2,c)T}_{j,\cdot}$$

where we used that $(\boldsymbol{\phi}_c \otimes \boldsymbol{I})\boldsymbol{I}_{\cdot,j} = \boldsymbol{I}_{\cdot,j}\boldsymbol{\phi}_c$. This computation is $O(n\,P\,C\,d)$ for all c. The update is

$$\operatorname{cholext}\Big(\boldsymbol{L}^{(3,c)}, \left(\boldsymbol{M}^{(3,c)T}, \boldsymbol{\beta}^{(3,c)}\right)^T, \pi_{j,c}^{1/2}\boldsymbol{D}^{(c)1/2}\boldsymbol{\Sigma}^{(c)}_{I,j}, 1 + \pi_{j,c}\boldsymbol{\Sigma}^{(c)}_j, \left(\pi_{j,c}^{1/2}\boldsymbol{\Sigma}^{(c)T}_{\cdot,j},\ \pi_{j,c}^{-1/2}b_{j,c} - \pi_{j,c}^{1/2}\boldsymbol{\mu}^{(c)}_j\right)^T\Big),$$

costing $O(n\,\sum_c d_c)$ in total. Finally, the posterior means and variances $\boldsymbol{h}^{(c)}$, $\boldsymbol{a}^{(c)}$ are recomputed from scratch (based on the new $\boldsymbol{M}^{(3,c)}$ matrices) at total cost of $O(n\,\sum_c d_c)$.

2.5.5 Remarks

The cost is dominated by the first step updates of $\mathcal{R}_2(c)$, namely of the $\boldsymbol{M}^{(2,c)}$ matrices, and by the first step updates of $\mathcal{R}_3(c)$ (the $\boldsymbol{M}^{(3,c)}$ matrices). For d inclusions it amounts to $O(n\,d^2\,C^2\,P)$. The memory cost is dominated by $O(n\,d\,C\,P)$ for the $\boldsymbol{M}^{(2,c)}$ matrices.

After the common inclusion phase, patterns are included w.r.t. specific classes, say j into I_c. That means that the update process described above uses rank 1 updates instead of rank $C-1$. As for Section 2.5.1, define $\boldsymbol{e}^{(c)} = -l^{-1}\boldsymbol{E}^{(c)}_{1\ldots d,\cdot}\boldsymbol{l} \in \mathbb{R}^d$. Then,

$$\hat{\boldsymbol{P}}^{(c)\prime} = \left(\hat{\boldsymbol{P}}^{(c)},\, (\boldsymbol{I}\otimes\boldsymbol{\phi}_c)\boldsymbol{e}^{(c)}\right),$$

and $\hat{\boldsymbol{P}}^{(c')}$ remain the same for $c' \neq c$. $\mathcal{R}_4$ is not modified (it is in fact not needed anymore). Furthermore, $\boldsymbol{K}_I$ remains the same and $\hat{\boldsymbol{P}}^{(\setminus c')}$ are modified by appending $(\boldsymbol{I}\otimes\boldsymbol{\phi}_c)\boldsymbol{e}^{(c)}$ for $c' \neq c$. In Section 2.5.2, $\mathcal{R}_2(c)$, $\mathcal{R}_3(c)$ are not updated (since they do not change). $\boldsymbol{E}^{(\setminus c')} = \boldsymbol{e}^{(c)}$, and $\boldsymbol{V}$, $\boldsymbol{W}$ have a single column. The $\mathcal{R}_2(c')$ updates cost $O(n\,P\,C\,d)$, and the $\mathcal{R}_3(c')$ updates are $O(n(P\,C\,d+\sum_c d_c))$. This is cheaper by a factor C than in the common inclusion phase, because the message updates are rank 1 instead of rank $C-1$. The second update step of Section 2.5.3 does not exist here, since I does not grow anymore. The final update of $\mathcal{R}_3(c)$ (for the single c) costs $O(n(P\,d+d_c))$. Finally, $\boldsymbol{h}^{(c')}$ and $\boldsymbol{a}^{(c')}$ are recomputed from scratch (for all c') at cost $O(n\sum_c d_c)$.

Therefore, the overall running time complexity is

$$O\left(n\left(P\,C\,d+\sum_c d_c\right)\sum_c d_c\right)$$

and the memory requirements are

$$O\left(n\left(P\,C\,d+\sum_c d_c\right)\right).$$

In large sample situations it makes sense to require $P\,C\,d$ to be of the same order as $\sum_c d_c$. In that case, the memory requirements of our method are the same as for the baseline up to a constant factor. However, it seems that modelling conditional dependencies between classes comes at a significant additional price of at least $O(n\,(\sum_c d_c)^2)$ as compared to $O(n\sum_c d_c^2)$ for the independent baseline. On the other hand, our method is faster than the naive implementation by a factor of C.

Note that if the active sets and site parameters are fixed, then the complete representation can be computed in

$$O\left(n\left(\sum_c d_c^2 + P\,d\left(C\,P\,d+\sum_c d_c\right)\right)\right)$$

which is significantly faster and actually fairly close to what the independent baseline requires. Therefore, in contrast to the situation for earlier IVM variants, conditional inference with active set selection comes at a significantly higher cost than without.[5] The problem is that while $\mathcal{R}_2(c)$ is of limited size $P\,d$, for each of the $\sum_c d_c$ inclusions $C-1$ of them have to be updated by rank 1. In other words, the matrices $\hat{\boldsymbol{P}}^{(\setminus c)}\hat{\boldsymbol{P}}^{(\setminus c)T}$ are of size $P\,d$, but are in fact updated $d_{\setminus c} > P\,d$ times by rank 1. Each such update has to be worked through the

[5] Even for other IVM variants the active set selection takes significantly more time in practice, but does not have a higher complexity.

whole representation in order to make sure that the marginals $\boldsymbol{h}^{(c)}$, $\boldsymbol{a}^{(c)}$ are up-to-date all the time. Delaying these updates does not help. We would have to delay the updates for a class c more than $P\,d$ times before it would be cheaper to simply recompute $\mathcal{R}_2(c)$, $\mathcal{R}_3(c)$ from scratch.

2.5.6 Prediction

In order to predict on test data, the dominant buffers scaling as $O(n)$ are not required. We need to compute the marginals of Q on the test point which is done just as above for the training points: compute $\boldsymbol{M}^{(2,c)}$, $\boldsymbol{\Sigma}^{(c)}_{\cdot,I_c}$, $\boldsymbol{M}^{(4)}$, and $\boldsymbol{M}^{(3,c)}$ w.r.t. the test points. The cost is the same as computing the representation for the training set from scratch (with fixed active sets and site parameters), but with n replaced by the number of test points m:

$$O\left(m\,\left(\sum_c d_c^2 + P\,d\,\left(C\,P\,d + \sum_c d_c\right)\right)\right).$$

Again, this is fairly close to the requirements of the baseline method. If m is large, the computation can be done in multiple chunks.[6] Predictive distributions can be computed from the marginals using Gaussian quadrature in general.

2.6 Selection of Points. Computing Site Parameters

In this section we show how new site parameters are computed and how the candidate j for the next inclusion into I is found. The principal tool for models with non-Gaussian likelihood factors $P(y_c|v_c)$ is the ADF projection mentioned in Section 2.1.

As mentioned in Section 2.3, there are two different phases in which patterns are included into the active sets. During the initial common inclusion phase, each pattern is included into all I_c in parallel, so that $I_c = I$ for all c during this phase. During the second phase, additional patterns are included w.r.t. specific classes c only. Both the forward selection of points to be included and the computation of site parameters is relatively straightforward in the second phase, while we need additional approximations during the common phase. We concentrate on describing the initial common phase.

Suppose that during the common inclusion phase, we have selected j for inclusion into I and want to compute the new site parameters $(b_{j,c})_c$, $(\pi_{j,c})_c$. First note that for a Gaussian likelihood, the site parameters are fixed in advance, *i.e.* if $P(y_{j,c}|v_{j,c}) = N(y_{j,c}|v_{j,c}, \sigma_c^2)$, then $\pi_{j,c} = \sigma_c^{-2}$, $b_{j,c} = y_{j,c}\sigma_c^{-2}$. Therefore, we assume that $P(y_{j,c}|v_{j,c})$ is not Gaussian. The idea behind ADF is as follows. Let $Q(\boldsymbol{v})$ be a Gaussian and $f(\boldsymbol{v}_j)$ a positive non-Gaussian potential depending only on a small number $\boldsymbol{v}_j$ of components of $\boldsymbol{v}$. If $\hat{P}(\boldsymbol{v}) \propto Q(\boldsymbol{v})f(\boldsymbol{v}_j)$, the goal is to approximate $\hat{P}$ by a Gaussian $Q'(\boldsymbol{v})$. In ADF, this is achieved by *moment matching*, *i.e.* $\hat{P}$ and Q' have the same mean and covariance matrix. Due to the special form of f, it is easy to see that $Q'(\boldsymbol{v}) = Q'(\boldsymbol{v}_j)Q(\boldsymbol{v} \setminus \boldsymbol{v}_j|\boldsymbol{v}_j)$, so we only need to match the moments of $\hat{P}(\boldsymbol{v}_j) \propto Q(\boldsymbol{v}_j)f(\boldsymbol{v}_j)$. This is feasible in general if $|\boldsymbol{v}_j|$ is very small, in our case it will be a single component. Note that we only need to know $Q(\boldsymbol{v}_j)$ in order to do the ADF projection. $\hat{P}$ is called *tilted distribution*.

[6] On the other hand, with specialized code for large matrix computations, the chunks should be made as large as memory resources permit.

In the second phase, we can apply ADF directly to obtain the site parameters, but during the common phase we would have to base it on $\boldsymbol{v}_j = (v_{j,c})_c$. This would require to compute a C-dimensional Gaussian expectation over a non-Gaussian function which is hard in general. Even for a Gaussian likelihood, we would have to have access to $Q(\boldsymbol{v}_j)$ which is a joint marginal spanning all of the $\boldsymbol{v}_c$. The representation developed above cannot be used to obtain these joint marginals efficiently, and we do not know of a way to obtain a large number of these joint marginals which is substantially more efficient than the naive method mentioned at the beginning of Section 2.2. We need a way of computing the site parameters which uses the single marginals $Q(v_{j,c})$ only. The simplest choice is to compute the parameters for (j, c) under the assumption that j is included into I_c only, requiring a one-dimensional ADF projection and $Q(v_{j,c})$ only. Since for each inclusion we use the old $Q(v_{j,c})$ marginal which does not incorporate the new information from the other $v_{j,c'}$ this is an approximation. Its effect on the overall approximation could be tested by comparing it with other more complicated schemes in which the site parameters are computed in some ordering over c and the computations are interleaved with marginal updates. We have not done this so far.

If $\hat{P}(v_{j,c}) \propto P(y_{j,c}|v_{j,c})Q(v_{j,c})$, $Q'(v_{j,c})$ is the Gaussian which minimizes $\mathrm{D}[\hat{P} \,\|\, \cdot]$. If $Q'(v_{j,c}) = N(\hat{h}_{j,c}, \hat{a}_{j,c})$, these parameters can be computed to high accuracy using one-dimensional Gauss-Hermite quadrature if the likelihood $P(y_{j,c}|v_{j,c})$ is reasonably smooth. In the case of the probit likelihood (Eq. 1) or the Gaussian likelihood we are mainly interested in here, the computation is analytic. Details are provided in Appendix C. The new site parameters then follow directly from

$$Q'(v_{j,c}) \propto Q(v_{j,c}) \exp(-\pi_{j,c} v_{j,c}^2/2 + b_{j,c} v_{j,c}).$$

Finally, we need a procedure for selecting a good inclusion candidate j among the indices not already in I. Just as in the single process IVM we make use of greedy forward selection where the selection criterion is an information measure: select the point whose subsequent inclusion changes the posterior most, *i.e.* which introduces the most new information into Q. During the second phase we need to score pairs (j, c). A convenient criterion based on the marginal $Q(v_{j,c})$ is the *information gain* $\Delta_{j,c} = -\mathrm{D}[Q'(v_{j,c}) \,\|\, Q(v_{j,c})]$, where Q' is the marginal after the inclusion of (j, c). $\Delta_{j,c}$ can be computed in $O(1)$ given $Q(v_{j,c})$. During the common inclusion phase, a good generalization of the information gain would depend on joint marginals $Q(\boldsymbol{v}_j)$ which we cannot obtain feasibly. However, a forward selection criterion which can be computed very efficiently is essential in order to be able to score most (or all) of the remaining patterns for each inclusion. In our case, we are looking for a criterion which depends on the single marginals $Q(v_{j,c})$ only, since we can afford to update all of them after each inclusion using our representation.

For the common inclusion phase, we propose to use the C information gain values $\Delta_{j,c}$ in order to construct an appropriate single score Δ_j. Possibilities are the average

$$\Delta_j^{avg} = C^{-1} \sum_c \Delta_{j,c}$$

or the maximum

$$\Delta_j^{max} = \max_c \Delta_{j,c}.$$

Note that $Q'(v_{j,c})$ is *not* the marginal after the inclusion of j. The latter would require to propagate the new evidence to $\boldsymbol{u}$ and back to the $\boldsymbol{v}_c$. Alternatively, one could replace the

Q' by these new marginals, however they may be costly to compute. We will explore this possibility in future work.

2.7 Joint Common Inclusion Phase

As mentioned in Section 2.6, there are two different phases for inclusions into the active sets, and during the first *common inclusion phase* our inability of computing *joint* marginals of $\boldsymbol{v}_j = (v_{j,c})_c$ leads to further somewhat unsatisfactory approximations. In Section 2.5.5 we determined the overall complexity and noted that a realistic scaling is obtained if $P\,C\,d \approx \sum_c d_c$. Under this assumption conditional inference with active set selection requires about $O(n(P\,C\,d)^2)$ time. In this Section we suggest a way of working with joint marginals during the common inclusion phase while still matching this scaling.

Note that the common inclusion phase requires $O(n\,P(C\,d)^2)$ and is clearly dominated by the second phase. If we run the common phase in the naive way mentioned at the beginning of Section 2.2 it scales as $O(n\,C(C\,d)^2)$ which is $O(n(P\,C\,d)^2)$ if $P^2 \approx C$. Recall that the naive method simply uses the single process IVM representation of Section 2.1 over $n\,C$ variables $\boldsymbol{v}$ with the kernel induced by $\tilde{K}^{(c)}$, $K^{(p)}$.

The memory requirement for the naive method is $O(n\,d\,C^2)$ which is by a factor C/P too large. However, we can limit the memory requirements by reducing the number of candidates which are scored for each inclusion. Suppose the (common) active set I has size $d' \leq d$. The matrix $\boldsymbol{M} \in \mathbb{R}^{m,Cd'}$ dominates the memory requirements, it has one row for every candidate which can be scored. If $m \leq n\,P\,d/d'$, then a buffer of $n\,d\,C\,P$ elements is sufficient. For the first few inclusions we can set $m = C\,n$. Eventually, rows have to removed from $\boldsymbol{M}$ after each inclusion. When selecting these rows priority is given to the ones attaining the worst current scores.

Based on the stub matrix $\boldsymbol{M}$ we can compute the joint distribution $Q(\boldsymbol{v}_j) = N(\boldsymbol{h}_j, \boldsymbol{A}_j),\ \boldsymbol{v}_j \in \mathbb{R}^C$ for every inclusion candidate j. If the likelihood $P(y|v)$ is Gaussian, then the tilted distribution $\hat{P}(\boldsymbol{v}_j)$ is Gaussian, *i.e.* $Q' = \hat{P}$, and the information gain $\Delta_j = -\mathrm{D}[Q'(\boldsymbol{v}_j) \,\|\, Q(\boldsymbol{v}_j)]$ can be computed analytically. In general, computing Δ_j requires C-dimensional quadrature which quickly becomes very costly or inaccurate. We suggest the following greedy proxy. Start with $Q_0(\boldsymbol{v}_j) = Q(\boldsymbol{v}_j),\ D_0 = \emptyset$, and $\Delta^0 = 0$. For $c = 1, \ldots, C$ compute

$$\Delta^{c,c'} = -\mathrm{D}[Q'_{c,c'}(\boldsymbol{v}_j) \,\|\, Q_{c-1}(\boldsymbol{v}_j)]$$

for all $c' \in \{1, \ldots, C\} \setminus D_{c-1}$. Here, $\hat{P}_{c,c'}(\boldsymbol{v}_j) \propto P(y_{c'}|v_{c'})Q_{c-1}(\boldsymbol{v}_j)$ and $Q'_{c,c'}$ is the Gaussian ADF projection of $\hat{P}_{c,c'}$. Note that $\Delta^{c,c'} = -\mathrm{D}[Q'_{c,c'}(v_{c'}) \,\|\, Q_{c-1}(v_{c'})]$, and that the ADF projections are one-dimensional only. If c' is the argument minimizing these $\Delta^{c,c'}$, set $\Delta^c = \Delta^{c-1} + \Delta^{c,c'}$ and $D_c = D_{c-1} \cup \{c'\}$. The final score value for j is Δ^C.

2.8 Limiting Resource Requirements

It is important to note that the scaling of $O(n)$ for the inference approximation is only due to the fact that we wish to consider all remaining points as inclusion candidates for the active sets I_c for all classes. In a situation of limited resources, we may reduce the size of the sample or restrict the active set sizes d_c, but a third option is to score only subsets of

points as candidates for later inclusions. It is quite likely that later inclusion decisions are less important than earlier ones, so that this way of limiting resource requirements may be much less intrusive than the other ones. In this section we outline a simple scheme which can be used to this end.

Let $J_c \subset \{1, \ldots, n\}$ be the selection index for class c, and $n_c = |J_c|$. For the first inclusions, ideally for all of the common inclusion phase, $n_c = n$ for all c, but during later stages the n_c decrease following some schedule which may be aimed at keeping the memory cost below a fixed threshold at all times. If we only insist on considering the points in J_c as inclusion candidates into I_c, we only need to maintain $\boldsymbol{h}_{J_c}^{(c)}$, $\boldsymbol{a}_{J_c}^{(c)}$ which means that only $\boldsymbol{M}_{J_c,\cdot}^{(2,c)}$, $\boldsymbol{M}_{J_c,\cdot}^{(3,c)}$ have to be stored and updated. A point which is excluded from J_c should not be re-included later on, because it is easy to show that doing so costs no less than just leaving it in J_c in the first place. Interestingly we can use the scores of all points in J_c in order to decide which subset to exclude, for example by retaining the top-scorers. Some randomization is recommended as well. In fact the problem of maintaining J_c is similar to organizing a cache, with the important difference that the scores we care about can be evaluated efficiently for all elements all the time, so we may draw on existing strategies developed for the latter.

3 Parameter Learning

In this section we show how free parameters can be learned automatically from the data D in an empirical Bayesian manner. It turns out that the conditional inference approximation developed above is required to drive this optimization.

We can separate the unobserved variables in our model in two categories: primary and secondary parameters. The former are the latent processes $\boldsymbol{v}(\cdot)$, or equivalently their values $\boldsymbol{v}$ at the training points, the latter consist of $\boldsymbol{\Phi}$, $\boldsymbol{\sigma}$ and the parameters of the covariance functions $\tilde{K}^{(c)}$, $K^{(p)}$. Secondary parameters are sometimes called hyperparameters in a Bayesian setting, although in a semiparametric setup such as ours one is typically careful to distinguish parameters such as $\boldsymbol{\Phi}$ from "nuisance" parameters (the kernel parameters in our context). For the former we are explicitly interested in a point estimate, while the latter would be best integrated out and are estimated only if such a marginalization proves intractable (such as in our case). Denote the vector of all secondary parameters by $\boldsymbol{\alpha}$.

Empirical Bayesian techniques combine marginalization and maximization in the sense that primary parameters are integrated out, but hyperparameters are maximized over. In our case, the *marginal likelihood* is

$$P(\boldsymbol{y}|\boldsymbol{\alpha}) = \int P(\boldsymbol{y}|\boldsymbol{v}) P(\boldsymbol{v}|\boldsymbol{\alpha})\, d\boldsymbol{v}.$$

In the *maximum likelihood II (ML-II)* method we choose the hyperparameters $\hat{\boldsymbol{\alpha}}$ as (local) maximizer of $P(\boldsymbol{y}|\boldsymbol{\alpha})$ or $P(\boldsymbol{y}, \boldsymbol{\alpha})$. The latter requires the specification of a hyperprior $P(\boldsymbol{\alpha})$ and implies that ML-II can also be seen as *maximum a posteriori (MAP)* technique in that the posterior $P(\boldsymbol{\alpha}|\boldsymbol{y})$ is maximized.

The computation of $\log P(\boldsymbol{y}|\boldsymbol{\alpha})$ is as hard as doing conditional inference (*i.e.* computing marginals of the conditional posterior $P(\boldsymbol{v}|\boldsymbol{y}, \boldsymbol{\alpha})$). In terms of statistical physics, $\log P(\boldsymbol{y}|\boldsymbol{\alpha})$ is the *log partition function* of the model. Interestingly a general *variational* framework

allows us to use a method for approximate inference in order to lower bound the log partition function. We can then maximize the lower bound in order to determine $\hat{\boldsymbol{\alpha}}$. Note that $\log P(\boldsymbol{y}|\boldsymbol{\alpha})$ is a convex function of $\boldsymbol{v} \mapsto \log P(\boldsymbol{y}, \boldsymbol{v}|\boldsymbol{\alpha})$. By Legendre-Fenchel duality [1, 2] we have

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

for any distribution $Q(\boldsymbol{v})$. The maximizer $Q(\boldsymbol{v})$ for the lower bound is the posterior $P(\boldsymbol{v}|\boldsymbol{y}, \boldsymbol{\alpha})$ (in general duality terms, $\boldsymbol{v} \mapsto \log P(\boldsymbol{y}, \boldsymbol{v}|\boldsymbol{\alpha})$ and $P(\boldsymbol{v}|\boldsymbol{y}, \boldsymbol{\alpha})$ are dually coupled), but any other posterior approximation gives a valid lower bound. In this paper, we use the particular $Q(\boldsymbol{v})$ described in Section 2 as variational distribution.

By a slight abuse of notation let $\boldsymbol{v}_I$ denote the vector of all components $v_{i,c}$, $c = 1, \ldots, C$, $i \in I_c$. In order to compute the relative entropy term, we note that $Q(\boldsymbol{v} \setminus \boldsymbol{v}_I|\boldsymbol{v}_I) = P(\boldsymbol{v} \setminus \boldsymbol{v}_I|\boldsymbol{v}_I)$, therefore

$$\mathrm{D}[Q(\boldsymbol{v}) \,\|\, P(\boldsymbol{v})] = \mathrm{D}[Q(\boldsymbol{v}_I) \,\|\, P(\boldsymbol{v}_I)].$$

Recall that the efficient representation we use for $Q(\boldsymbol{v})$ allows access to the marginals $Q(\boldsymbol{v}_c)$, $c = 1, \ldots, C$ only, so especially $Q(\boldsymbol{v}_I)$ is not available. If $\boldsymbol{v}_{I,c} = (v_{i,c})_{i \in I_c}$, we make use of the additional bounding step

$$\mathrm{D}[Q(\boldsymbol{v}_I) \,\|\, P(\boldsymbol{v}_I)] \leq \sum_c \mathrm{D}[Q(\boldsymbol{v}_{I,c}) \,\|\, P(\boldsymbol{v}_{I,c})].$$

Thus, the learning criterion to be minimized is

$$\mathcal{G} = \sum_{c=1}^{C} \left(\sum_{i=1}^{n} \mathrm{E}_Q[-\log P(y_{i,c}|v_{i,c})] + \mathrm{D}\left[Q(\boldsymbol{v}_{I,c}) \,\|\, P(\boldsymbol{v}_{I,c})\right] \right).$$

It turns out that given the representation of the $Q(\boldsymbol{v}_c)$, the criterion and its gradient w.r.t. $\boldsymbol{\alpha}$ can be computed efficiently if the dependence of $\{I_c, \boldsymbol{b}^{(c)}, \boldsymbol{D}^{(c)}\}$ on $\boldsymbol{\alpha}$ is neglected. Collect the latter variables in the *inner state* $\boldsymbol{s}$. The computation of $\nabla_{\boldsymbol{\alpha}} \mathcal{G}$ is involved, it is sketched in Appendix D. Importantly, the time complexity is the same as for conditional inference, and the computation is actually significantly cheaper because software for large matrix computations can be employed. Furthermore, if the buffers required for conditional inference are overwritten, the additional memory requirements are subdominant.

We use a simple double-loop optimization strategy. In the inner loop, we fix the inner state $\boldsymbol{s}$ and perform gradient-based minimization of $\mathcal{G}$ using a Quasi-Newton method. In the outer loop, $\boldsymbol{s}$ is re-selected as described in Section 2. There is no obvious stopping criterion for the outer loop, so we run for a fixed number of outer loop iterations, or until no more improvement is recorded in the inner loop.

We close this section with a number of comments. First, note that our practice differs from most other variational methods in which *all* of Q is kept fixed for an update of $\boldsymbol{\alpha}$. Q depends on both the inner state $\boldsymbol{s}$ and α, and the latter dependence is strong. If we fixed all of Q during the inner loop, the criterion could not be improved much. In a variant of our double-loop method we could split $\boldsymbol{s}$ into the active set indices and the site parameters and recompute the latter more often. However, it is important to note that in the case of Gaussian likelihoods $P(y|v)$ for the regression model, the site parameters are fixed anyway, so that $\boldsymbol{s}$ contains the active set indices only. In this important special case, both proposals are the same.

Note that, as opposed to many standard variational methods, the optimization strategy is not a pure descent method. Re-selection of $\boldsymbol{s}$ can actually lead to an increase in $\mathcal{G}$. The problem is that "closeness" to the true posterior $P(\boldsymbol{v}|\boldsymbol{y}, \boldsymbol{\alpha})$ is measured in different ways in the variational bound and in our sparse approximation. In other words, the active sets I_c are not selected with the minimization of $\mathcal{G}$ in mind. Futhermore, the optimization is not convex in any sense. The overall process involves the selection of subsets which really is a discrete optimization. Since our goal is to improve upon a "practically intractable" yet just cubic scaling, invoking any of the heavy machinery to deal with combinatorial problems would certainly not be sensible. But even for fixed $\boldsymbol{s}$, the inner loop problem is not convex in general. Convexity is only obtained in rather unnatural parameterizations of the kernel matrices whose relevance in practice is unclear. It would be very interesting to find convex relaxations of our problem which retain the statistical features to some quantifiable extent.

A Notation

In this section we describe the notation used in this paper.

A.1 Matrices and Vectors

Vectors $\boldsymbol{a} \in \mathbb{R}^n$ and matrices $\boldsymbol{A} \in \mathbb{R}^{n,m}$ are set in boldface. We write $\boldsymbol{a} = (a_i)_i = (a_1, \ldots, a_n)^T$ and $\boldsymbol{A} = (a_{i,j})_{i,j}$ for the components. Ordered set subscripts are used to extract corresponding parts of vectors or matrices: for $I \subset \{1, \ldots, n\},\ J \subset \{1, \ldots, m\}$ we have $\boldsymbol{A}_{I,J} = (a_{i,j})_{i \in I, j \in J}$ and $\boldsymbol{a}_I = (a_i)_{i \in I}$. We write short "$\cdot$" for the whole range and i (instead of $\{i\}$) for single element sets, *i.e.* $\boldsymbol{a}_i = a_i,\ \boldsymbol{A}_{\cdot,j} = (a_{i,j})_i,\ \boldsymbol{A}_{\cdot,\cdot} = \boldsymbol{A}$, *etc.* We also abbreviate $\boldsymbol{B}_{I,I}$ to $\boldsymbol{B}_I$.

The matrices $\boldsymbol{I}_{\cdot,I}$ and $\boldsymbol{I}_{I,\cdot}$ (with $\cdot = \{1, \ldots, n\}$) should be regarded as distribution and selection operators respectively. $\boldsymbol{I}_{\cdot,I}\boldsymbol{b}$ for $\boldsymbol{b} \in \mathbb{R}^d$ is the vector in $\mathbb{R}^n$ with b_k at position i_k and 0 elsewhere, if $I = \{i_1, \ldots, i_d\}$. Furthermore, $\boldsymbol{I}_{I,\cdot}\boldsymbol{a} = (a_{i_1}, \ldots, a_{i_d})^T$. Note that for matrices we have $\boldsymbol{I}_{K,\cdot}\boldsymbol{A}\boldsymbol{I}_{\cdot,I} = \boldsymbol{A}_{K,I}$.

In this paper we need to use matrices and vectors with double indexes (i, c) or (i, p). In order to be able to present our derivations in a reasonable fluid manner, we choose a "lightweight" notation which may seem unfamiliar and ambiguous to the reader at first. In general we treat double indexes as flat with i (the index over datapoints) meant to be the inner one, changing faster than than the index c over classes or p over latent processes. For example, $\boldsymbol{u} = (u_{i,p})_{i,p} = (u_{1,1}, \ldots, u_{n,1}, \ldots, u_{n,P})^T$. We also write $\boldsymbol{u}_i = (u_{i,p})_p$ and $\boldsymbol{u}_p = (u_{i,p})_i$, in this context we will exclusively use i, j as indexes over datapoints, c as index over classes, and p as index over latent processes.

Subset indexing is "overloaded" to the double index case as follows. If I, $J \subset \{1, \ldots, n\}$ and $\boldsymbol{A} \in \mathbb{R}^{nP,nP}$, then $\boldsymbol{A}_{I,J}$ is short for $\boldsymbol{A}_{I \times \{1,\ldots,P\}, J \times \{1,\ldots,P\}}$. Thus, selection is done only on the index over datapoints. However, if an index p or c is used, selection is meant to be w.r.t. latent process or class. For example, if $\boldsymbol{A} \in \mathbb{R}^{nC,nP}$, then $\boldsymbol{A}_{c,p} = (a_{(i,c),(j,p)})_{i,j} \in \mathbb{R}^{n,n}$. The same holds for obvious variants such as p', c', p_1, c_1, *etc.*

A.2 Other Notation

We need to manipulate Gaussians in complicated ways and make use of the convenient notation introduced in [8]. Let

$$N^U(\boldsymbol{x}|\boldsymbol{r},\boldsymbol{V}) = \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{x} + \boldsymbol{r}^T\boldsymbol{x}\right)$$

denote an *unnnormalized Gaussian*. Here, $\boldsymbol{V}$ must be symmetric but need not be positive definite. If so, we have

$$N^U(\boldsymbol{x}|\boldsymbol{r},\boldsymbol{V}) = N\left(\boldsymbol{x}|\boldsymbol{V}^{-1}\boldsymbol{r},\boldsymbol{V}^{-1}\right)\left|2\pi\boldsymbol{V}^{-1}\right|^{1/2}\exp\left(\frac{1}{2}\boldsymbol{r}^T\boldsymbol{V}^{-1}\boldsymbol{r}\right).$$

B Derivations for Representation and Updates

In this section we collect derivations for the representation and the scheme to update it after inclusions. The subsections are referenced from the main text and are not self-contained.

B.1 The Message $m_{\boldsymbol{v}_c\to\boldsymbol{u}}$

The message $m_{\boldsymbol{v}_c\to\boldsymbol{u}}$ is defined in Eq. 4. If $\boldsymbol{\mu} = (\boldsymbol{\phi}_c^T\otimes\boldsymbol{I})\boldsymbol{u}$, then

$$\int \Psi_v(\boldsymbol{v}_c)\Psi_{u\to v}(\boldsymbol{v}_c,\boldsymbol{u})\,d\boldsymbol{v}_c \propto \exp\left(\frac{1}{2}\boldsymbol{r}^T\boldsymbol{\Sigma}\boldsymbol{r} - \frac{1}{2}\boldsymbol{\mu}^T\tilde{\boldsymbol{K}}^{(c)-1}\boldsymbol{\mu}\right),$$

where $\boldsymbol{r} = \boldsymbol{I}_{\cdot,I}\boldsymbol{b}^{(c)} + \tilde{\boldsymbol{K}}^{(c)-1}\boldsymbol{\mu}$ and $\boldsymbol{\Sigma} = (\tilde{\boldsymbol{K}}^{(c)-1} + \boldsymbol{I}_{\cdot,I}\boldsymbol{D}^{(c)}\boldsymbol{I}_{I,\cdot})^{-1} = \tilde{\boldsymbol{K}}^{(c)} - \boldsymbol{M}^{(1,c)}\boldsymbol{M}^{(1,c)T}$ (a collection of useful formulae for manipulating Gaussians can be found in [8], Sect. A.4.3). Plugging these in, some algebra gives

$$m_{\boldsymbol{v}_c\to\boldsymbol{u}} \propto \exp\left(\boldsymbol{\beta}^{(1,c)T}\boldsymbol{\gamma} - \frac{1}{2}\boldsymbol{\gamma}^T\boldsymbol{\gamma}\right), \quad \boldsymbol{\gamma} = \boldsymbol{L}^{(1,c)-1}\boldsymbol{D}^{(c)1/2}\boldsymbol{\mu}_I = \boldsymbol{L}^{(1,c)-1}\boldsymbol{D}^{(c)1/2}(\boldsymbol{\phi}_c^T\otimes\boldsymbol{I})\boldsymbol{I}_{I,\cdot}\boldsymbol{u}.$$

With the definition of $\boldsymbol{P}^{(c)}$ (Eq. 7) we have $\boldsymbol{\gamma} = \boldsymbol{P}^{(c)T}\boldsymbol{u}_I$ (recall that we use a different component ordering for $\boldsymbol{u}_I$).

B.2 First Update of $\boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}$

We have

$$\left(\boldsymbol{A}^{(2,c)} + \boldsymbol{V}\boldsymbol{V}^T\right)^{-1} = \boldsymbol{A}^{(2,c)-1} - \boldsymbol{M}\boldsymbol{M}^T, \quad \boldsymbol{M} = \boldsymbol{L}^{(2,c)-T}\boldsymbol{W}\boldsymbol{L}^{-T}, \quad \boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{I} + \boldsymbol{W}^T\boldsymbol{W}.$$

Thus,

$$\left(\boldsymbol{M}^{(2,c)}\boldsymbol{M}^{(2,c)T}\right)' = \boldsymbol{M}^{(2,c)}\boldsymbol{M}^{(2,c)T} - \tilde{\boldsymbol{M}}\tilde{\boldsymbol{M}}^T,$$
$$\tilde{\boldsymbol{M}} = (\boldsymbol{\phi}_c^T\otimes\boldsymbol{I})\boldsymbol{K}_{\cdot,I}\boldsymbol{\Pi}\boldsymbol{M} = \boldsymbol{M}^{(2,c)}\boldsymbol{W}\boldsymbol{L}^{-T} \in \mathbb{R}^{n,C-1}$$

which shows how to update $\boldsymbol{\Sigma}^{(c)}$. $\tilde{\boldsymbol{M}}$ is required explicitly in the subsequent $\mathcal{R}_3(c)$ update, the cost is $O(n\,C\,P\,d)$. Furthermore,

$$\begin{aligned}(\boldsymbol{\mu}^{(c)})' &= (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{K}_{\cdot,I}\boldsymbol{\Pi}\left(\boldsymbol{A}^{(2,c)-1} - \boldsymbol{M}\boldsymbol{M}^T\right)\hat{\boldsymbol{K}}_{I,I'}\left(\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)}\right)' \\ &= \boldsymbol{\mu}^{(c)} + \boldsymbol{\mu}_*^{(c)}, \quad \boldsymbol{\mu}_*^{(c)} = \boldsymbol{M}^{(2,c)}\boldsymbol{w} - \tilde{\boldsymbol{M}}\boldsymbol{M}^T\hat{\boldsymbol{K}}_{I,I'}\left(\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)}\right)'\end{aligned}$$

and

$$\boldsymbol{M}^T\hat{\boldsymbol{K}}_{I,I'}\left(\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)}\right)' = \boldsymbol{L}^{-1}\boldsymbol{W}^T\left(\boldsymbol{\beta}^{(2,c)} + \boldsymbol{w}\right).$$

Using the facts $\boldsymbol{w} = \boldsymbol{W}\boldsymbol{\beta}_*^{(\backslash c)}$ and $\boldsymbol{I} + \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{L}\boldsymbol{L}^T$, some algebra gives

$$\boldsymbol{\mu}_*^{(c)} = \tilde{\boldsymbol{M}}\boldsymbol{L}^{-1}\left(\boldsymbol{\beta}_* - \boldsymbol{W}^T\boldsymbol{\beta}^{(2,c)}\right).$$

B.3 Second Part of $\mathcal{R}_3(c)$ Update

Recall the $\boldsymbol{\Sigma}^{(c)}$ update from Eq. 15. We need to adjust the $\mathcal{R}_3(c)$ representation to incorporate this change and the one of $\boldsymbol{\mu}^{(c)}$. This is done in two steps, first a positive *chollrup* for $\boldsymbol{M}_*$ and the $\boldsymbol{\mu}^{(c)}$ change, then a negative *chollrup* for $(\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{M}_{**}$. Recall our convention that after each of these steps, the new variables $\boldsymbol{x}'$ overwrite the old ones $\boldsymbol{x}$.

For the positive update we need $\boldsymbol{Q} = \boldsymbol{L}^{(3,c)-1}\boldsymbol{D}^{(c)\,1/2}(\boldsymbol{M}_*)_{I_c,\cdot}$. As for $\boldsymbol{M}^{(3,c)}$, we first replace $\boldsymbol{\Sigma}_{\cdot,I_c}^{(c)}$ which amounts to

$$\boldsymbol{M}^{(3,c)} \to \boldsymbol{M}^{(3,c)} + \boldsymbol{M}_*\boldsymbol{Q}^T,$$

then drag it along the *chollrup* in order to replace $\boldsymbol{L}^{(3,c)-T}$ by $(\boldsymbol{L}^{(3,c)\prime})^{-T}$. As for $\boldsymbol{\beta}^{(3,c)}$, we update $\boldsymbol{\mu}_{I_c}^{(c)}$ which amounts to

$$\boldsymbol{\beta}^{(3,c)} \to \boldsymbol{\beta}^{(3,c)} - \boldsymbol{L}^{(3,c)-1}\boldsymbol{D}^{(c)\,1/2}(\boldsymbol{M}_*)_{I_c,\cdot}\boldsymbol{\beta}_*,$$

then drag it along.

For the negative *chollrup* we need $\boldsymbol{Q} = \boldsymbol{L}^{(3,c)-1}\boldsymbol{D}^{(c)\,1/2}(\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})(\boldsymbol{M}_{**})_{I_c,\cdot}$ and replace

$$\boldsymbol{M}^{(3,c)} \to \boldsymbol{M}^{(3,c)} - (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{M}_{**}\boldsymbol{Q}^T$$

before dragging it along. $\boldsymbol{\beta}^{(3,c)}$ is simply dragged along. Both updates are of rank P, so the individual cost is $O(n\,P\,d_c)$ (which is also the cost of computing $\boldsymbol{M}_*\boldsymbol{Q}^T$ and $(\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{M}_{**}\boldsymbol{Q}^T$).

C Details for ADF Projection

In this section we provide details for the ADF projection discussed in Section 2.6. Recall that we need to compute mean and variance of the tilted distribution $\propto P(y|v)N(v|h,a)$. Let

$$Z = \mathrm{E}[P(y|v)], \quad \alpha = \frac{d}{dh}\log Z, \quad \nu = -\frac{d^2}{dh^2}\log Z,$$

where $\mathrm{E}[\cdot]$ is over $N(v|h,a)$. It is easy to see that mean $\hat{h}$ and variance $\hat{a}$ of the tilted distribution are given by

$$\hat{h} = h + \alpha a, \quad \hat{a} = a(1 - a\nu).$$

Furthermore, the information gain criterion described in Section 2.6 can be computed as

$$\Delta_{j,c}^{info} = -\frac{1}{2}\left(-\log(1 - a\nu) + 1 - a\nu - 1 + \alpha^2 a\right).$$

If the ADF projection is used for an inclusion, the new site parameters can be determined from

$$N(v|\hat{h},\hat{a}) \propto N(v|h,a)\exp\left(-dv^2/2 + bv\right),$$

they are given as

$$d = \frac{\nu}{1 - a\nu}, \quad b = \frac{h\nu + \alpha}{1 - a\nu}.$$

It is interesting to note that if $P(y|v)$ is (strictly) log-concave, then one can show that $\log Z$ is (strictly) concave in h, so that $\nu \geq 0$ ($\nu > 0$). For such a likelihood, d will always be set to a nonnegative number. In our implementation we reject inclusions if the corresponding ADF update leads to a very small d.[7]

For a general smooth likelihood $P(y|v)$, α and ν can be computed by Gauss-Hermite quadrature. For the probit likelihood (Eq. 1), the computations are analytic. We have

$$Z = \mathrm{E}_{v\sim N(h,a),n\sim N(0,1)}\left[\mathrm{I}_{\{n\leq y(v+\sigma)\}}\right] = \Phi(z), \quad z = y(h+\sigma)s,\ s = \frac{1}{\sqrt{1+a}}.$$

Then,

$$\alpha = \frac{N(z)}{\Phi(z)}ys = \exp\left(\log N(z) - \log\Phi(z)\right)ys$$

and

$$\nu = -ys\frac{N(z)}{\Phi(z)}(-zys - \alpha) = \alpha\left(\alpha + \frac{h+\sigma}{1+a}\right).$$

For the Gaussian likelihood $P(y|u) = N(y|u,\sigma^2)$, we have $Z = N(y|h, a+\sigma^2)$, so that

$$\alpha = \frac{y-h}{a+\sigma^2}, \quad \nu = \frac{1}{a+\sigma^2}.$$

D Learning Criterion and Gradient

In this section we provide details for the computation of the learning criterion $\mathcal{G}$ developed in Section 3 and its gradient. The complete derivation is lengthy and some details are ommitted here.

Let $\mathcal{G} = \mathcal{G}_1 + \mathcal{G}_2$ with

$$\mathcal{G}_1 = \sum_{c,i}\mathrm{E}_Q[-\log P(y_{i,c}|v_{i,c})], \quad \mathcal{G}_2 = \sum_c \mathcal{G}_2(c), \quad \mathcal{G}_2(c) = \mathrm{D}\left[Q(\boldsymbol{v}_{I,c}) \,\|\, P(\boldsymbol{v}_{I,c})\right].$$

[7] Our selection criterion actually favours updates which lead to larger d.

Recall that $\boldsymbol{v}_{I,c} = (v_{i,c})_{i \in I_c}$. We have $\mathcal{G}_1 = -\mathbf{1}^T \boldsymbol{z}$ with

$$z_{i,c} = \mathrm{E}_Q[\log P(y_{i,c}|v_{i,c})], \quad Q(v_{i,c}) = N(h_{i,c}, a_{i,c}),\ \boldsymbol{h}^{(c)} = (h_{i,c})_i,\ \boldsymbol{a}^{(c)} = (a_{i,c})_i.$$

Recall that $\boldsymbol{h}^{(c)}$, $\boldsymbol{a}^{(c)}$ denote marginal means and variances of $Q(\boldsymbol{v}_c)$. If

$$c_{i,c} = -a_{i,c}^{-1/2} \mathrm{E}_Q\left[(\log P(y_{i,c}|v_{i,c}))\, \tilde{v}_{i,c}\right], \quad g_{i,c} = \frac{1}{2a_{i,c}} \left(z_{i,c} - \mathrm{E}_Q\left[(\log P(y_{i,c}|v_{i,c}))\, \tilde{v}_{i,c}^2\right]\right),$$
$$\tilde{v}_{i,c} = a_{i,c}^{-1/2} \left(v_{i,c} - h_{i,c}\right),$$

then

$$\begin{aligned} dz_{i,c} &= -g_{i,c} da_{i,c} - c_{i,c} dh_{i,c}, \text{ i.e.} \\ d\mathcal{G}_1 &= \boldsymbol{g}^T (d\boldsymbol{a}) + \boldsymbol{c}^T (d\boldsymbol{h}) = \sum_c \left(\boldsymbol{g}_c^T \left(d\boldsymbol{a}^{(c)}\right) + \boldsymbol{c}_c^T \left(d\boldsymbol{h}^{(c)}\right)\right). \end{aligned} \tag{16}$$

Due to the multi-part representation and the nontrivial message flow, the gradient computation is very challenging. First, we need to find the general form of $d\mathcal{G}$ in terms of accumulator matrices. If we ignore the parameters $(\sigma_c)_c$ of the likelihood for the moment, we can infer from the representation that

$$\begin{aligned} d\mathcal{G} = &\sum_c \boldsymbol{z}_c^{(1)T} \left(d \operatorname{diag} \tilde{\boldsymbol{K}}^{(c)}\right) + \sum_p \boldsymbol{z}_p^{(2)T} \left(d \operatorname{diag} \boldsymbol{K}^{(p)}\right) + \operatorname{tr} \boldsymbol{Z}^{(1)T} (d\boldsymbol{\Phi}) \\ &+ \sum_c \operatorname{tr} \boldsymbol{Z}_c^{(2)T} \left(d\tilde{\boldsymbol{K}}^{(c)}_{\cdot,I_c}\right) + \sum_p \operatorname{tr} \boldsymbol{Z}_p^{(3)T} \left(d\boldsymbol{K}^{(p)}_{\cdot,I}\right) + \sum_{c,p} \operatorname{tr} \boldsymbol{Z}_{c,p}^{(4)T} \left(d\boldsymbol{K}^{(p)}_{\cdot,I_c \setminus I}\right) \\ &+ \sum_c \operatorname{tr} \boldsymbol{Z}_c^{(5)T} \left(d\tilde{\boldsymbol{K}}^{(c)}_{I_c}\right). \end{aligned} \tag{17}$$

The computation starts with a loop over $c = 1, \ldots, C$ in which the accumulator matrices are computed. Here, the dominant matrices $\boldsymbol{Z}_c^{(2)}$ overwrite the buffers used for $\boldsymbol{M}^{(3,c)}$. We maintain $\boldsymbol{Z}_c^{(5)}$ separately, because in every iteration c, *all* of the $\boldsymbol{Z}_{c'}^{(5)}$ are modified, and we cannot do these modifications on the $\boldsymbol{Z}_{c'}^{(2)}$ because they only come available once $\boldsymbol{M}^{(3,c')}$ is not required anymore. The gradient is computed in subsequent loops over c and p. In the following we sketch how the accumulators are updated during a fixed iteration c of the first loop. Operations marked with $(*)$ are done only once, say for $c = 1$. If p appears, the operation is done for every $p = 1, \ldots, P$ (if nothing else is said).

From IVM learning in the single process case, we know how to formulate $d\boldsymbol{h}^{(c)}$, $d\boldsymbol{a}^{(c)}$ in terms of $d\boldsymbol{\Sigma}^{(c)}_{\cdot,I_c}$, $d \operatorname{diag} \boldsymbol{\Sigma}^{(c)}$ and $d\boldsymbol{\mu}^{(c)}$. Thus $d\mathcal{G}_1$ can be written as linear expression in these terms, and the same is true for a part of $d\mathcal{G}_2$ as we show now. Our first goal is to write

$$d\mathcal{G}_1 + d\mathcal{G}_{2,part} = \sum_c \left(\boldsymbol{g}_c^T \left(d \operatorname{diag} \boldsymbol{\Sigma}^{(c)}\right) + \operatorname{tr} \tilde{\boldsymbol{Z}}_c^T \left(d\boldsymbol{\Sigma}^{(c)}_{\cdot,I_c}\right) + \boldsymbol{f}^{(c)T} \left(d\boldsymbol{\mu}^{(c)}\right)\right). \tag{18}$$

with $\tilde{\boldsymbol{Z}}_c$, $\boldsymbol{f}^{(c)}$ to be determined. We then compute the remaining part of $d\mathcal{G}_2$, and finally deal with the principal part (Eq. 18). Let

$$\begin{aligned} &\boldsymbol{\gamma}^{(c)} = \boldsymbol{D}^{(c)\,1/2} \boldsymbol{L}^{(3,c)-T} \boldsymbol{\beta}^{(3,c)}, \quad \tilde{\boldsymbol{M}}^{(3,c)} = \boldsymbol{M}^{(3,c)} \boldsymbol{L}^{(3,c)-1} \boldsymbol{D}^{(c)\,1/2}, \\ &\boldsymbol{f}^{(c)} = \boldsymbol{c}_c - \boldsymbol{I}_{\cdot,I_c} \tilde{\boldsymbol{M}}^{(3,c)T} \boldsymbol{c}_c \end{aligned} \tag{19}$$

where we overwrite $\boldsymbol{M}^{(3,c)}$ by $\tilde{\boldsymbol{M}}^{(3,c)}$. We have

$$d\boldsymbol{M}^{(3,c)}\boldsymbol{M}^{(3,c)T} = \boldsymbol{H}\left(d\boldsymbol{\Sigma}^{(c)}_{\cdot,I_c}\right)\tilde{\boldsymbol{M}}^{(3,c)\,T} + \tilde{\boldsymbol{M}}^{(3,c)}\left(d\boldsymbol{\Sigma}^{(c)}_{\cdot,I_c}\right)^T,$$
$$d\boldsymbol{M}^{(3,c)}\boldsymbol{\beta}^{(3,c)} = \boldsymbol{H}\left(d\boldsymbol{\Sigma}^{(c)}_{\cdot,I_c}\right)\boldsymbol{\gamma}^{(c)} + (\boldsymbol{H}-\boldsymbol{I})\left(d\boldsymbol{\mu}^{(c)}\right), \quad \boldsymbol{H} = \boldsymbol{I} - \tilde{\boldsymbol{M}}^{(3,c)}\boldsymbol{I}_{I_c,\cdot},$$

therefore

$$d\boldsymbol{h}^{(c)} = \boldsymbol{H}\left(d\boldsymbol{\mu}^{(c)}\right) + \boldsymbol{H}\left(d\boldsymbol{\Sigma}^{(c)}_{\cdot,I_c}\right)\boldsymbol{\gamma}^{(c)},$$
$$d\boldsymbol{a}^{(c)} = \left(d\,\mathrm{diag}\,\boldsymbol{\Sigma}^{(c)}\right) - \mathrm{diag}\,(\boldsymbol{I}+\boldsymbol{H})\left(d\boldsymbol{\Sigma}^{(c)}_{\cdot,I_c}\right)\tilde{\boldsymbol{M}}^{(3,c)\,T}.$$

Noting that $\boldsymbol{f}^{(c)} = \boldsymbol{H}^T\boldsymbol{c}_c$ and $\boldsymbol{g}_c^T\,\mathrm{diag}\,\boldsymbol{B} = \mathrm{tr}(\mathrm{diag}\,\boldsymbol{g}_c)\boldsymbol{B}$, we have

$$\tilde{\boldsymbol{Z}}_c = -2(\mathrm{diag}\,\boldsymbol{g}_c)\tilde{\boldsymbol{M}}^{(3,c)} + \boldsymbol{I}_{\cdot,I_c}\tilde{\boldsymbol{M}}^{(3,c)T}(\mathrm{diag}\,\boldsymbol{g}_c)\tilde{\boldsymbol{M}}^{(3,c)} + \boldsymbol{f}^{(c)}\boldsymbol{\gamma}^{(c)T} \tag{20}$$

which overwrites $\tilde{\boldsymbol{M}}^{(3,c)}$. The cost is $O(n\,d_c^2)$ for each c. As for $\mathcal{G}_2(c)$, let

$$\boldsymbol{T}^{(c)} = \tilde{\boldsymbol{K}}^{(c)}_{I_c} + (\boldsymbol{\phi}_c^T\otimes\boldsymbol{I})\boldsymbol{K}_{I_c}(\boldsymbol{\phi}_c\otimes\boldsymbol{I}) \in \mathbb{R}^{d_c,d_c}$$

and denote

$$\boldsymbol{A}^{(c)} = \mathrm{E}_Q[\boldsymbol{v}_{I,c}] = \boldsymbol{\Sigma}^{(c)}_{I_c} - \boldsymbol{M}^{(3,c)}_{I_c,\cdot}\boldsymbol{M}^{(3,c)T}_{I_c,\cdot}.$$

We have that $P(\boldsymbol{v}_{I,c}) = N(\boldsymbol{0},\boldsymbol{T}^{(c)})$ and $Q(\boldsymbol{v}_{I,c}) = N(\boldsymbol{h}^{(c)}_{I_c},\boldsymbol{A}^{(c)})$. If $\boldsymbol{R}^{(c)} = \boldsymbol{T}^{(c)-1}\boldsymbol{A}^{(c)}$, then we have

$$\mathcal{G}_2(c) = \frac{1}{2}\left(-\log|\boldsymbol{R}^{(c)}| + \mathrm{tr}\,\boldsymbol{R}^{(c)} - d_c + \boldsymbol{h}^{(c)T}_{I_c}\boldsymbol{T}^{(c)-1}\boldsymbol{h}^{(c)}_{I_c}\right).$$

We make use of the Cholesky decomposition of $\boldsymbol{T}^{(c)}$ to compute expressions involving $\boldsymbol{T}^{(c)-1}$. We have

$$d\log|\boldsymbol{R}^{(c)}| = \mathrm{tr}\,\boldsymbol{A}^{(c)-1}\left(d\boldsymbol{A}^{(c)}\right) - \mathrm{tr}\,\boldsymbol{T}^{(c)-1}\left(d\boldsymbol{T}^{(c)}\right),$$
$$d\,\mathrm{tr}\,\boldsymbol{R}^{(c)} = -\mathrm{tr}\,\boldsymbol{T}^{(c)-1}\boldsymbol{A}^{(c)}\boldsymbol{T}^{(c)-1}\left(d\boldsymbol{T}^{(c)}\right) + \mathrm{tr}\,\boldsymbol{T}^{(c)-1}\left(d\boldsymbol{A}^{(c)}\right)$$

and

$$d\boldsymbol{T}^{(c)} = d\tilde{\boldsymbol{K}}^{(c)}_{I_c} + \sum_p \phi_{c,p}^2\left(d\boldsymbol{K}^{(p)}_{I_c}\right) + 2\sum_p \phi_{c,p}\boldsymbol{K}^{(p)}_{I_c}(d\phi_{c,p}).$$

And with $\boldsymbol{t}^{(c)} = \boldsymbol{T}^{(c)-1}\boldsymbol{h}^{(c)}_{I_c}$ we have

$$d\boldsymbol{h}^{(c)T}_{I_c}\boldsymbol{T}^{(c)-1}\boldsymbol{h}^{(c)}_{I_c} = 2\boldsymbol{t}^{(c)T}\left(d\boldsymbol{h}^{(c)}_{I_c}\right) - \boldsymbol{t}^{(c)T}\left(d\boldsymbol{T}^{(c)}\right)\boldsymbol{t}^{(c)},$$

therefore

$$d\mathcal{G}_2(c) = \mathrm{tr}\,\boldsymbol{J}^{(1,c)T}\left(d\boldsymbol{A}^{(c)}\right) + \mathrm{tr}\,\boldsymbol{J}^{(2,c)T}\left(d\boldsymbol{T}^{(c)}\right) + \boldsymbol{t}^{(c)T}\left(d\boldsymbol{h}^{(c)}_{I_c}\right),$$
$$\boldsymbol{J}^{(1,c)} = \frac{1}{2}\left(\boldsymbol{T}^{(c)-1} - \boldsymbol{A}^{(c)-1}\right), \quad \boldsymbol{J}^{(2,c)} = \frac{1}{2}\left(\boldsymbol{T}^{(c)-1} - \boldsymbol{T}^{(c)-1}\boldsymbol{A}^{(c)}\boldsymbol{T}^{(c)-1} - \boldsymbol{t}^{(c)}\boldsymbol{t}^{(c)T}\right).$$

Also,

$$d\boldsymbol{A}^{(c)} = \left(d\boldsymbol{\Sigma}^{(c)}_{I_c}\right) - 2\,\mathrm{sym}\,\tilde{\boldsymbol{M}}^{(3,c)}_{I_c,\cdot}\left(d\boldsymbol{\Sigma}^{(c)}_{I_c}\right) + \tilde{\boldsymbol{M}}^{(3,c)}_{I_c,\cdot}\left(d\boldsymbol{\Sigma}^{(c)}_{I_c}\right)\tilde{\boldsymbol{M}}^{(3,c)T}_{I_c,\cdot}.$$

We can incorporate the $d\boldsymbol{h}_{I_c}^{(c)}$ part by replacing $\boldsymbol{c}_c$ in Eq. 16 by $\boldsymbol{c}_c + \boldsymbol{I}_{\cdot,I_c}\boldsymbol{t}^{(c)}$ which affects the computation of $\boldsymbol{f}^{(c)}$ (Eq. 19) and $\tilde{\boldsymbol{Z}}_c$ (Eq. 20). The $d\boldsymbol{A}^{(c)}$ is then incorporated by

$$\tilde{\boldsymbol{Z}}_c \leftarrow \tilde{\boldsymbol{Z}}_c + \boldsymbol{I}_{\cdot,I_c}\left(\tilde{\boldsymbol{M}}_{I_c,\cdot}^{(3,c)} - \boldsymbol{I}\right)^T \boldsymbol{J}^{(1,c)}\left(\tilde{\boldsymbol{M}}_{I_c,\cdot}^{(3,c)} - \boldsymbol{I}\right).$$

The $d\boldsymbol{T}^{(c)}$ part results in the following direct gradient contributions:

$$\begin{aligned}
&\boldsymbol{Z}_c^{(5)} += \boldsymbol{J}^{(2,c)}, \quad \boldsymbol{Z}_p^{(3)} += \phi_{c,p}^2 \boldsymbol{I}_{\cdot,I_c}\boldsymbol{J}_{\cdot,1\ldots d}^{(2,c)}, \quad \boldsymbol{Z}_{c,p}^{(4)} += \phi_{c,p}^2 \boldsymbol{I}_{\cdot,I_c}\boldsymbol{J}_{\cdot,d+1\ldots d_c}^{(2,c)}, \\
&\boldsymbol{Z}^{(1)} += \left(2\phi_{c,p}\,\mathrm{tr}\,\boldsymbol{J}^{(2,c)}\boldsymbol{K}_{I_c}^{(p)}\right)_{c,p}.
\end{aligned} \tag{21}$$

Denote the parts in Eq. 18 by $d\mathcal{G}_{1,1}(c)$, $d\mathcal{G}_{1,2}(c)$, $d\mathcal{G}_{1,3}(c)$ respectively. Let

$$\boldsymbol{\Sigma}^{(c)} = \tilde{\boldsymbol{\Sigma}}^{(c)} + \boldsymbol{M}^{(2,c)}\boldsymbol{M}^{(2,c)T}, \quad \tilde{\boldsymbol{\Sigma}}^{(c)} = \tilde{\boldsymbol{K}}^{(c)} + (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\left(\boldsymbol{K} - \boldsymbol{M}^{(4)}\boldsymbol{M}^{(4)T}\right)(\boldsymbol{\phi}_c \otimes \boldsymbol{I}).$$

$\boldsymbol{\mu}^{(c)}$ does not depend on $\tilde{\boldsymbol{\Sigma}}^{(c)}$. Consider only the $\tilde{\boldsymbol{\Sigma}}^{(c)}$ variation for the moment. Let

$$\tilde{\boldsymbol{M}}^{(4)} = \boldsymbol{M}^{(4)}\boldsymbol{L}^{(4)-1} = \mathrm{diag}\left(\boldsymbol{M}_p^{(4)}\boldsymbol{L}_p^{(4)-1}\right)_p$$

which overwrites $\boldsymbol{M}^{(4)}$ once not needed anymore. The cost is $O(n\,P\,d^2)$. From $d\mathcal{G}_{1,1}(c)$ we have

$$\begin{aligned}
&\boldsymbol{z}_c^{(1)} += \boldsymbol{g}_c, \quad \boldsymbol{z}_p^{(2)} += \tilde{\boldsymbol{g}}_p\ (*), \quad \tilde{\boldsymbol{g}}_p = \sum_c \phi_{c,p}^2 \boldsymbol{g}_c, \\
&\boldsymbol{Z}^{(1)} += \left(2\phi_{c,p}\left(\mathrm{diag}\,\boldsymbol{K}^{(p)} - \mathrm{diag}\,\boldsymbol{M}_p^{(4)}\boldsymbol{M}_p^{(4)T}\right)^T \boldsymbol{g}_c\right)_{c,p}, \\
&\boldsymbol{Z}_p^{(3)} += -2(\mathrm{diag}\,\tilde{\boldsymbol{g}}_p)\tilde{\boldsymbol{M}}_p^{(4)} + \boldsymbol{I}_{\cdot,I}\tilde{\boldsymbol{M}}_p^{(4)T}(\mathrm{diag}\,\tilde{\boldsymbol{g}}_p)\tilde{\boldsymbol{M}}_p^{(4)}\ (*),
\end{aligned} \tag{22}$$

and from $d\mathcal{G}_{1,2}(c)$ we have

$$\begin{aligned}
&\boldsymbol{Z}_c^{(2)} += \tilde{\boldsymbol{Z}}_c, \quad \boldsymbol{Z}_{c,p}^{(4)} += \phi_{c,p}^2(\tilde{\boldsymbol{Z}}_c)_{\cdot,d+1\ldots d_c}, \\
&\boldsymbol{Z}^{(1)} += \left(2\phi_{c,p}\,\mathrm{tr}\,\tilde{\boldsymbol{Z}}_c^T\left(\boldsymbol{K}_{\cdot,I_c}^{(p)} - \boldsymbol{M}_p^{(4)}\boldsymbol{M}_{I_c,p}^{(4)\ T}\right)\right)_{c,p}, \\
&\boldsymbol{Z}_p^{(3)} += \phi_{c,p}^2\left((\tilde{\boldsymbol{Z}}_c)_{\cdot,1\ldots d} - \tilde{\boldsymbol{Z}}_c\tilde{\boldsymbol{M}}_{I_c,p}^{(4)} - \boldsymbol{I}_{\cdot,I_c}\tilde{\boldsymbol{Z}}_c^T\tilde{\boldsymbol{M}}_p^{(4)} + \boldsymbol{I}_{\cdot,I}\left(\tilde{\boldsymbol{Z}}_c^T\tilde{\boldsymbol{M}}_p^{(4)}\right)^T\tilde{\boldsymbol{M}}_{I_c,p}^{(4)}\right).
\end{aligned} \tag{23}$$

Here and elsewhere, if (say) $\boldsymbol{M}^{(4)} = (m_{i,(j,p)})_{i,(j,p)}$, then $\boldsymbol{M}_{I_c,p}^{(4)} = (m_{i,(j,p)})_{i\in I_c,j} \in \mathbb{R}^{d_c,d}$. $\boldsymbol{Z}_c^{(2)}$ overwrites $\tilde{\boldsymbol{Z}}_c$. The $\boldsymbol{Z}_c^{(2)}$ accumulator is accessed only here, so it is easiest to compute the $\tilde{\boldsymbol{Z}}_c$ in the c loop and do the $d\tilde{\boldsymbol{\Sigma}}^{(c)}$ updates in one batch after this loop. The total cost of the batch (given $\tilde{\boldsymbol{Z}}_c$) is $O(n\,P\,d\sum_c d_c)$. Note that we need the kernel matrix $\boldsymbol{K}_{\cdot,I_c}^{(p)}$ here. Except in cases where kernel evaluations are very expensive, these matrices should be recomputed here on the fly in order to save memory (part of the kernel matrix is required in Eq. 24 as well).

Next the $d\boldsymbol{M}^{(2,c)}\boldsymbol{M}^{(2,c)T}$ part. Since

$$d(\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{K}_{\cdot,I} = \left(d\boldsymbol{\phi}_c^T \otimes \boldsymbol{I}\right)\boldsymbol{K}_{\cdot,I} + (\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\left(d\boldsymbol{K}_{\cdot,I}\right),$$

we see that for a $d(\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{K}_{\cdot,I}$ part, a contribution $\boldsymbol{Z}_p^{(3)} += \phi_{c,p}\boldsymbol{B}$ is paired with $\boldsymbol{Z}^{(1)} += (\operatorname{tr} \boldsymbol{B}^T \boldsymbol{K}_{\cdot,I}^{(p)})_{c,p}$, so we only need to deal with the $d\boldsymbol{K}_{\cdot,I}$ part explicitly. Let

$$\tilde{\boldsymbol{M}}^{(2,c)} = \boldsymbol{M}^{(2,c)} \boldsymbol{L}^{(2,c)-1} \boldsymbol{\Pi}^T \in \mathbb{R}^{n,Pd}$$

which overwrites $\boldsymbol{M}^{(2,c)}$, the cost is $O(n\,P^2\,d^2)$. Then,

$$d\boldsymbol{M}^{(2,c)} \boldsymbol{M}^{(2,c)T} = \sum_p 2\phi_{c,p} \operatorname{sym}\left(d\boldsymbol{K}_{\cdot,I}^{(p)}\right) \tilde{\boldsymbol{M}}_p^{(2,c)T} - \tilde{\boldsymbol{M}}^{(2,c)} \left(d\boldsymbol{\Pi} \boldsymbol{A}^{(2,c)} \boldsymbol{\Pi}^T\right) \tilde{\boldsymbol{M}}^{(2,c)T}$$

(ignoring $d\phi_c$). We deal with the $d\boldsymbol{A}^{(2,c)}$ part later. Let

$$\begin{aligned} \boldsymbol{S}_c &= \tilde{\boldsymbol{M}}^{(2,c)T} (\operatorname{diag} \boldsymbol{g}_c) \tilde{\boldsymbol{M}}^{(2,c)} = \left(\boldsymbol{S}_c^{(p,p')}\right)_{p,p'} \in \mathbb{R}^{Pd,Pd}, \\ \boldsymbol{F}^{(c)} &= \tilde{\boldsymbol{Z}}_c^T \tilde{\boldsymbol{M}}^{(2,c)} \in \mathbb{R}^{d_c,Pd}, \end{aligned}$$

which comes at cost $O(n\,P^2\,d^2)$ for $\boldsymbol{S}_c$ and $O(n\,P\,d\,d_c)$ for $\boldsymbol{F}^{(c)}$. The contribution of $d\boldsymbol{A}^{(2,c)}$ (through $d\mathcal{G}_{1,1}(c)$, $d\mathcal{G}_{1,2}(c)$) is

$$d\mathcal{G}_{1,1}(c) + d\mathcal{G}_{1,2}(c) = \operatorname{tr}\left(-\boldsymbol{S}_c - \tilde{\boldsymbol{M}}_{I_c,\cdot}^{(2,c)T} \boldsymbol{F}^{(c)}\right) \left(d\boldsymbol{\Pi} \boldsymbol{A}^{(2,c)} \boldsymbol{\Pi}^T\right).$$

Ignoring $d\boldsymbol{A}^{(2,c)}$ we have the contributions

$$\begin{aligned} &\boldsymbol{Z}_p^{(3)} += \phi_{c,p}\boldsymbol{B}, \quad \boldsymbol{B} = 2(\operatorname{diag} \boldsymbol{g}_c)\tilde{\boldsymbol{M}}_p^{(2,c)} + \tilde{\boldsymbol{Z}}_c \tilde{\boldsymbol{M}}_{I_c,p}^{(2,c)} + \boldsymbol{I}_{\cdot,I_c} \boldsymbol{F}_p^{(c)}, \\ &\boldsymbol{Z}^{(1)} += \left(\operatorname{tr} \boldsymbol{B}^T \boldsymbol{K}_{\cdot,I}^{(p)}\right)_{c,p} \end{aligned} \tag{24}$$

at cost $O(n\,P\,d\,d_c)$. The relationship through $\boldsymbol{B}$ may be exploited in an implementation. $\boldsymbol{B}$ occurs as intermediate in the computation of $\boldsymbol{S}_c$.

Next we look at $d\mathcal{G}_{1,3}(c)$. If

$$\boldsymbol{\varepsilon}^{(c)} = \boldsymbol{\Pi} \boldsymbol{L}^{(2,c)-T} \boldsymbol{\beta}^{(2,c)} \in \mathbb{R}^{Pd},$$

then

$$d\boldsymbol{\mu}^{(c)} = \left(d(\boldsymbol{\phi}_c^T \otimes \boldsymbol{I})\boldsymbol{K}_{\cdot,I}\right) \boldsymbol{\varepsilon}^{(c)} - \tilde{\boldsymbol{M}}^{(2,c)} \left(d\boldsymbol{\Pi} \boldsymbol{A}^{(2,c)} \boldsymbol{\Pi}^T\right) \boldsymbol{\varepsilon}^{(c)} + \tilde{\boldsymbol{M}}^{(2,c)} \left(d\boldsymbol{K}_I \boldsymbol{\Pi} \hat{\boldsymbol{P}}^{(\setminus c)} \hat{\boldsymbol{\beta}}^{(\setminus c)}\right). \tag{25}$$

Again, $d\boldsymbol{A}^{(2,c)}$ is dealt with later. If

$$\boldsymbol{q}^{(c)} = \tilde{\boldsymbol{M}}^{(2,c)T} \boldsymbol{f}^{(c)} \in \mathbb{R}^{Pd},$$

the contribution is

$$d\mathcal{G}_{1,3}(c) = \operatorname{tr}\left(-\boldsymbol{\varepsilon}^{(c)} \boldsymbol{q}^{(c)T}\right) \left(d\boldsymbol{\Pi} \boldsymbol{A}^{(2,c)} \boldsymbol{\Pi}^T\right).$$

For the first part in Eq. 25 we have

$$\boldsymbol{Z}_p^{(3)} += \phi_{c,p}\boldsymbol{B}, \quad \boldsymbol{B} = \boldsymbol{f}^{(c)} \boldsymbol{\varepsilon}_p^{(c)T}, \quad \boldsymbol{Z}^{(1)} += \left(\operatorname{tr} \boldsymbol{B}^T \boldsymbol{K}_{\cdot,I}^{(p)}\right)_{c,p}, \tag{26}$$

where again we use the pairing of $\boldsymbol{Z}_p^{(3)}$ and $\boldsymbol{Z}^{(1)}$ contributions. Note that this contribution can be fused with Eq. 24 by adding the corresponding $\boldsymbol{B}$ matrices. For the third part in Eq. 25 we note that in $\hat{\boldsymbol{P}}^{(\backslash c)}$, the rows are in $\boldsymbol{u}_I$ ordering, so that

$$\boldsymbol{\Pi}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)} = \sum_{c'\neq c}(\boldsymbol{\phi}_{c'}\otimes\boldsymbol{I})\boldsymbol{E}^{(c')}_{1\ldots d,\cdot}\boldsymbol{\beta}^{(1,c')}. \tag{27}$$

Define

$$\boldsymbol{B}^{(c)} = \boldsymbol{E}^{(c)}_{1\ldots d,\cdot}\boldsymbol{E}^{(c)T} \in \mathbb{R}^{d,d_c}, \quad \boldsymbol{\kappa}^{(c)} = \boldsymbol{E}^{(c)}\boldsymbol{\beta}^{(1,c)} \in \mathbb{R}^{d_c}.$$

Then, $d\boldsymbol{K}_I$ gives the contribution

$$\boldsymbol{Z}_p^{(3)} += \boldsymbol{I}_{\cdot,I}\boldsymbol{q}_p^{(c)}\left(\sum_{c'\neq c}\phi_{c',p}\boldsymbol{\kappa}^{(c')}_{1\ldots d}\right)^T. \tag{28}$$

The contribution for $d\boldsymbol{\Pi}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)}$ makes use of Eq. 27:

$$\begin{aligned}
\boldsymbol{Z}^{(1)} &+= \left(\boldsymbol{q}_p^{(c)T}\boldsymbol{K}_I^{(p)}\boldsymbol{\kappa}^{(c')}_{1\ldots d}\right)_{c'\neq c,p},\\
\boldsymbol{Z}^{(5)}_{c'} &+= \left(-\boldsymbol{B}^{(c')T}\left(\sum_p \phi_{c',p}\boldsymbol{K}_I^{(p)}\boldsymbol{q}_p^{(c)}\right)\boldsymbol{\kappa}^{(c')T}\right)\ (c'\neq c).
\end{aligned} \tag{29}$$

Here, only the rows $c'\neq c$ of $\boldsymbol{Z}^{(1)}$ are updated.

It remains to deal with the $d\boldsymbol{A}^{(2,c)}$ part which is

$$d\mathcal{G}_1(c) = \operatorname{tr}\boldsymbol{U}^{(c)}\left(d\boldsymbol{\Pi}\boldsymbol{A}^{(2,c)}\boldsymbol{\Pi}^T\right), \quad \boldsymbol{U}^{(c)} = -\boldsymbol{S}_c - \tilde{\boldsymbol{M}}^{(2,c)T}_{I_c,\cdot}\boldsymbol{F}^{(c)} - \boldsymbol{\varepsilon}^{(c)}\boldsymbol{q}^{(c)T} \in \mathbb{R}^{Pd,Pd}.$$

Here, $\boldsymbol{U}^{(c)}$ can be replaced by $\operatorname{sym}\boldsymbol{U}^{(c)}$ which we do. We have

$$d\boldsymbol{\Pi}\boldsymbol{A}^{(2,c)}\boldsymbol{\Pi}^T = (d\boldsymbol{K}_I) + 2\operatorname{sym}(d\boldsymbol{K}_I)\,\boldsymbol{\Pi}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{P}}^{(\backslash c)T}\boldsymbol{\Pi}^T\boldsymbol{K}_I + \boldsymbol{K}_I\left(d\boldsymbol{\Pi}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{P}}^{(\backslash c)T}\boldsymbol{\Pi}^T\right)\boldsymbol{K}_I.$$

If

$$\boldsymbol{Q}^{(c)} = \boldsymbol{U}^{(c)}\boldsymbol{K}_I\boldsymbol{\Pi}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{P}}^{(\backslash c)T}\boldsymbol{\Pi}^T \in \mathbb{R}^{Pd,Pd},$$

then ignoring $d\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{P}}^{(\backslash c)T}$, the contribution is

$$\boldsymbol{Z}_p^{(3)} += \boldsymbol{I}_{\cdot,I}\left(\boldsymbol{U}_p^{(c)} + 2\boldsymbol{Q}_p^{(c)}\right), \tag{30}$$

where we used that $\boldsymbol{U}^{(c)}$ is symmetric. Recall that $\boldsymbol{U}_p^{(c)}$ denotes the $d\times d$ block at position (p,p) in $\boldsymbol{U}^{(c)}$. The remaining term is

$$\operatorname{tr}\boldsymbol{K}_I\boldsymbol{U}^{(c)}\boldsymbol{K}_I\left(d\boldsymbol{\Pi}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{P}}^{(\backslash c)T}\boldsymbol{\Pi}^T\right),$$

and

$$\begin{aligned}
d\boldsymbol{\Pi}\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{P}}^{(\backslash c)T}\boldsymbol{\Pi}^T = \sum_{c'\neq c}\Big(&2\operatorname{sym}(d\boldsymbol{\phi}_{c'}\otimes\boldsymbol{I})\,\boldsymbol{B}^{(c')}_{\cdot,1\ldots d}(\boldsymbol{\phi}_{c'}^T\otimes\boldsymbol{I})\\
&-(\boldsymbol{\phi}_{c'}\otimes\boldsymbol{I})\boldsymbol{B}^{(c')}\left(d\tilde{\boldsymbol{K}}^{(c')}_{I_{c'}}\right)\boldsymbol{B}^{(c')T}(\boldsymbol{\phi}_{c'}^T\otimes\boldsymbol{I})\Big).
\end{aligned}$$

If

$$\boldsymbol{W}^{(c)} = \boldsymbol{K}_I(\boldsymbol{\phi}_c \otimes \boldsymbol{I})\boldsymbol{B}^{(c)} = \left(\phi_{c,p}\boldsymbol{K}_I^{(p)}\boldsymbol{B}^{(c)}\right)_p \in \mathbb{R}^{Pd,d_c},$$

then the contribution is

$$\boldsymbol{Z}_{c'}^{(5)} += -\boldsymbol{W}^{(c')T}\boldsymbol{U}^{(c)}\boldsymbol{W}^{(c')}, \quad \boldsymbol{Z}^{(1)} += \left(2\,\mathrm{tr}\,\boldsymbol{W}_{\cdot,1\ldots d}^{(c')\,T}\boldsymbol{U}_{\cdot,p}^{(c)}\boldsymbol{K}_I^{(p)}\right)_{c'\neq c,p}. \tag{31}$$

This completes the description of $d\mathcal{G}$ w.r.t. parameters of the covariance functions. The computational cost is estimated under the assumption that $n \gg d_c$ for all c and $P < C$. The dominating operations are the computation of $\tilde{\boldsymbol{M}}^{(3,c)}$ and $\tilde{\boldsymbol{Z}}_c$ at $O(n\sum_c d_c^2)$, the computation of $\tilde{\boldsymbol{M}}^{(2,c)}$ and $\boldsymbol{S}_c$ at $O(n\,C\,P^2\,d^2)$, and the computation of the $\tilde{\boldsymbol{\Sigma}}$ batch and of the $\boldsymbol{F}^{(c)}$ at $O(n\,P\,d\sum_c d_c)$. Thus, the complexity is

$$O\left(n\,\left(\sum_c d_c^2 + P\,d\,\left(C\,P\,d + \sum_c d_c\right)\right)\right)$$

which is the same as cost as computing the representation without selecting the active sets (see Section 2.5.5).

The gradient w.r.t. likelihood parameters depends on the details. For example, if the $P(y_c|v_c)$ have intercept parameters σ_c in the sense that $P(y_c|v_c) = f(y_c, v_c + \sigma_c)$, it is easy to see that

$$d\mathcal{G}_1 = \left(\sum_i \boldsymbol{c}_i\right)^T d\boldsymbol{\sigma}.$$

If $P(y_c|v_c) = N(y_c|v_c, \sigma_c^2)$, then

$$d\mathcal{G}_1 = \sum_{c,i}\frac{1}{2}\left(1 - \frac{a_{i,c} + (y_{i,c} - h_{i,c})^2}{\sigma_c^2}\right)\sigma_c^{-2}d\sigma_c^2. \tag{32}$$

We close the Section providing some further details for the gradient computation in our implementation. First of all, it is actually not sensible to store the accumulators $\boldsymbol{Z}_{c,p}^{(4)} \in \mathbb{R}^{n,d_c-d}$ which would come at a total memory cost of $O(n\,P\sum_c d_c)$. Fortunately, the $\boldsymbol{Z}_{c,p}^{(4)}$ are updated in Eq. 23 and Eq. 21 only, so it is easy to compute the gradient contributions on the fly. In general we separate accumulation from the final traces with kernel derivative matrices for the sake of a simple implementation.

D.1 Derivative w.r.t. σ_c^2 for Regression Model

If the SLFM is used for regression with a Gaussian noise model $P(y_c|v_c) = N(y_c|v_c, \sigma_c^2)$ the site parameters $\boldsymbol{b}^{(c)}$, $\boldsymbol{D}^{(c)}$ depend on $(\sigma_c^2)_c$ and the fixed target values $\boldsymbol{y}$ only. In this case we can refrain from fixing the site parameters during the inner loop optimization, so that only the active set indicators I_c are kept fixed. In this Section we compute the derivative of $\mathcal{G}$ w.r.t. the noise variances.

Denote $l_c = \log\sigma_c^2$. We accumulate $\partial\mathcal{G}/\partial l_c$ in z_c. Note that $\boldsymbol{D}^{(c)} = \sigma_c^{-2}\boldsymbol{I}$ and $\boldsymbol{b}^{(c)} = \sigma_c^{-2}\boldsymbol{y}_{I_c}^{(c)}$, so that

$$d\boldsymbol{D}^{(c)\,1/2} = -\frac{1}{2}\boldsymbol{D}^{(c)\,1/2}(dl_c), \quad d\boldsymbol{D}^{(c)-1/2}\boldsymbol{b}^{(c)} = -\frac{1}{2}\boldsymbol{D}^{(c)-1/2}\boldsymbol{b}^{(c)}(dl_c).$$

In the following we make use of these simple relationships, furthermore of the fact that $\boldsymbol{D}^{(c)} \propto \boldsymbol{I}$ and therefore commutes with all matrices. We also concentrate exclusively on the dl_c terms.

Some algebra gives

$$d\boldsymbol{M}^{(3,c)}\boldsymbol{M}^{(3,c)T} = -\sigma_c^2 \tilde{\boldsymbol{M}}^{(3,c)}\tilde{\boldsymbol{M}}^{(3,c)\,T}(dl_c), \quad d\boldsymbol{M}^{(3,c)}\boldsymbol{\beta}^{(3,c)} = -\sigma_c^2 \tilde{\boldsymbol{M}}^{(3,c)}\boldsymbol{\gamma}^{(c)},$$

so Eq. 16 results in the contribution

$$z_c += \sigma_c^2 \left(\operatorname{tr} \tilde{\boldsymbol{M}}^{(3,c)\,T} (\operatorname{diag} \boldsymbol{g}_c) \tilde{\boldsymbol{M}}^{(3,c)} - \boldsymbol{c}_c^T \tilde{\boldsymbol{M}}^{(3,c)} \boldsymbol{\gamma}^{(c)} \right).$$

Contributions from $d\mathcal{G}_2$ are through $d\boldsymbol{A}^{(c)}$, $d\boldsymbol{h}_{I_c}^{(c)}$:

$$z_c += \sigma_c^2 \left(\operatorname{tr} \tilde{\boldsymbol{M}}_{I_c,\cdot}^{(3,c)\,T} \boldsymbol{J}^{(1,c)} \tilde{\boldsymbol{M}}_{I_c,\cdot}^{(3,c)} - \boldsymbol{t}^{(c)T} \tilde{\boldsymbol{M}}_{I_c,\cdot}^{(3,c)} \boldsymbol{\gamma}^{(c)} \right).$$

The next contribution is through $d\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{\beta}}^{(\backslash c)}$. We have $d\boldsymbol{\kappa}_{1,\ldots,d}^{(c)} = -\sigma_c^2 \boldsymbol{B}^{(c)}\boldsymbol{\kappa}^{(c)}(dl_c)$, resulting in the contribution

$$z_{c'} += \left(-\sigma_{c'}^2 \left(\sum_p \phi_{c',p} \boldsymbol{K}_I^{(p)} \boldsymbol{q}_p^{(c)} \right)^T \boldsymbol{B}^{(c')}\boldsymbol{\kappa}^{(c')} \right), \; c' \neq c.$$

Finally for the $d\hat{\boldsymbol{P}}^{(\backslash c)}\hat{\boldsymbol{P}}^{(\backslash c)T}$ part we note that

$$d\boldsymbol{E}_{1,\ldots,d,\cdot}^{(c)} \boldsymbol{E}_{1,\ldots,d,\cdot}^{(c)\;T} = -\sigma_c^2 \boldsymbol{B}^{(c)}\boldsymbol{B}^{(c)T},$$

resulting in the contribution

$$z_{c'} += \left(-\sigma_{c'}^2 \operatorname{tr} \boldsymbol{W}^{(c')T}\boldsymbol{U}^{(c)}\boldsymbol{W}^{(c')} \right), \; c' \neq c.$$

Note all of the z_c accumulations use terms which have already been computed for the main gradient. The contribution through the likelihood factors (Eq. 32) has to be added to z_c as well.

References

[1] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

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

[3] L. Breiman and J. Friedman. Predicting multivariate responses in multiple linear regression. *Journal of Roy. Stat. Soc. B*, 59(1):3–54, 1997.

[4] N. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, 2nd edition, 1993.

[5] N. D. Lawrence, M. Seeger, and R. 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, 2003.

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

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

[8] 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`.

[9] M. Seeger. Low rank updates for the Cholesky decomposition. Technical report, University of California at Berkeley, 2004. See `www.cs.berkeley.edu/~mseeger`.