Fast Forward Selection to Speed Up Sparse Gaussian Process Regression

Matthias Seeger
University of Edinburgh
5 Forrest Hill,
Edinburgh, EH1 2QL
seeger@dai.ed.ac.uk

Christopher K. I. Williams
University of Edinburgh
5 Forrest Hill,
Edinburgh, EH1 2QL
c.k.i.williams@ed.ac.uk

Neil D. Lawrence
University of Sheffield
211 Portobello Street,
Sheffield, S1 4DP
neil@dcs.shef.ac.uk

Abstract

We present a method for the sparse greedy approximation of Bayesian Gaussian process regression, featuring a novel heuristic for very fast forward selection. Our method is essentially as fast as an equivalent one which selects the "support" patterns at random, yet it can outperform random selection on hard curve fitting tasks. More importantly, it leads to a sufficiently stable approximation of the log marginal likelihood of the training data, which can be optimised to adjust a large number of hyperparameters automatically. We demonstrate the model selection capabilities of the algorithm in a range of experiments. In line with the development of our method, we present a simple view on sparse approximations for GP models and their underlying assumptions and show relations to other methods.

1 Introduction

Gaussian process (GP) models are powerful non-parametric tools for Bayesian supervised learning. In comparison with architectures such as multi-layer perceptrons, they are conceptually simpler to understand and handle in practice. In contrast to other kernel machines such as support vector machines (e.g., Vapnik, 1995), GPs are *probabilistic* models, which means that they yield "error bars" on predictions and allow standard Bayesian approaches to model selection to be used. A more widespread use has probably been hindered by their unfavourable scaling: $O(n^3)$ time, $O(n^2)$ memory for training, and at least $O(n)$ time for prediction on a test point, where n is the number of training points. A range of *sparse* GP inference approximations have addressed this problem (see Smola and Bartlett, 2001, Williams and Seeger, 2001, Csató and Opper, 2002, Tresp, 2000, Tipping, 2001). While these schemes are quite different, a common theme is the concentration of efforts on a subset of size d of the full dataset, referred to as *active set* I in this paper. This allows for predictions in time $O(d)$ or $O(d^2)$, and typically leads to a more favourable training scaling as well, such as $O(n\,d^2)$ time, $O(n\,d)$ memory. Perhaps the most crucial step in these sparse methods is the selection of I, for which strategies like exhaustive search are of course out of the question. One may simply select I at random, yet several experimental studies (e.g., Lawrence et al., 2002) show that this can lead to poor results. Greedy strategies which include new points one at a time into I, selecting them as maximizers of some heuristic amongst all points not in I, are promising tractable alternatives.

The method proposed here is closely related to the sparse batch ADATAP method of Csató et al. (2002). The latter resembles an *on-line* scheme, in that the training data is processed in random order, and updates of I and the parameters are done sequentially. While this method does not rely on selection heuristics, it typically requires several passes over the data and exhibits some dynamics w.r.t. I, which may lead to instabilities. We focus on greedy forward selection instead, advocating the use of *cheap* selection heuristics (in contrast to other greedy proposals such as, Smola and Bartlett, 2001). These allow us to do full greedy searches for all inclusions into I, yet ending up with a scheme which is essentially as fast as using random selections. We also show how hyperparameter adaption can be done by maximizing an approximation to the marginal likelihood of the data.

The structure of the paper is as follows. In the remainder of this section we introduce the Gaussian process regression (GPR) setting. In Section 2, we describe our sparse approximation to GPR, show how an active set can be selected efficiently and how hyperparameters can be adjusted by maximizing the marginal likelihood. Relations to other sparse approximations are clarified in Section 3. Experimental results are presented in Section 4.

Vectors $\boldsymbol{g} = (g_i)_i$ and matrices $\boldsymbol{G} = (g_{i,j})_{i,j}$ are boldface[1]. If I, J are sets of indices, we denote the corresponding sub-matrix of $\boldsymbol{G} \in \mathbb{R}^{p,q}$ by $\boldsymbol{G}_{I,J}$. We abbreviate $\boldsymbol{G}_{I,1\dots q}$ to $\boldsymbol{G}_{I,\cdot}$, $\boldsymbol{G}_{I,\{j\}}$ to $\boldsymbol{G}_{I,j}$, $\boldsymbol{G}_{I,I}$ to $\boldsymbol{G}_I$, $\boldsymbol{G}_{I,\{1,\dots,q\}\setminus J}$ by $\boldsymbol{G}_{I,\setminus J}$, *etc.* The Gaussian density with mean $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Sigma}$ is denoted by $N(\cdot|\boldsymbol{m}, \boldsymbol{\Sigma})$.

1.1 Gaussian Process Regression

In this paper we focus on regression estimation. Given a sample $S = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$, $y_i \in \mathbb{R}$, drawn independently and identically distributed (i.i.d.) from an unknown distribution, our goal is to estimate the conditional distribution $P(y|\boldsymbol{x})$ for typical $\boldsymbol{x}$. To model this situation, we introduce a latent variable $u \in \mathbb{R}$ and make the assumption that $P(y|u, \boldsymbol{x}) = P(y|u)$. Thus, y is a noisy realisation of u, and we model the noise as $P(y|u) = N(y|u, \sigma^2)$, where σ^2 is a hyperparameter. The relationship $\boldsymbol{x} \to u$ is a random process $u(\cdot)$, which in a *Gaussian process (GP)* model is given a GP prior with mean function 0 and covariance kernel $K(\cdot,\cdot)$, where the latter is typically defined in terms of further hyperparameters. This process prior can be understood (and visualised) by noting that its joint evaluation at a finite number of input points is a zero-mean Gaussian variable with the Gram matrix induced by K as covariance matrix. We denote the sequence of latent outputs at the training points by $\boldsymbol{u} = (u(\boldsymbol{x}_1), \dots, u(\boldsymbol{x}_n))^T \in \mathbb{R}^n$ and the corresponding covariance or *kernel* matrix by $\boldsymbol{K} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j} \in \mathbb{R}^{n,n}$, i.e., $P(\boldsymbol{u}) = N(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K})$.

If we condition on the hyperparameters, the posterior process $P(u_*|\boldsymbol{x}_*, S)$ can be determined analytically. First,

$$\begin{aligned} P(\boldsymbol{u}|S) &\propto P(\boldsymbol{u}) N(\boldsymbol{y}|\boldsymbol{u}, \sigma^2 \boldsymbol{I}) \\ &\propto N(\boldsymbol{u}|\boldsymbol{K}(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}, \sigma^2\boldsymbol{K}(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}), \end{aligned}$$

furthermore $P(u_*|\boldsymbol{x}_*, S) = \int P(u_*|\boldsymbol{u}) P(\boldsymbol{u}|S)\, d\boldsymbol{u} = N(u_*|\mu_*, \sigma_*^2)$, where

$$\begin{aligned} \mu_* &= \boldsymbol{y}^T(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_*, \quad \boldsymbol{k}_* = (K(\boldsymbol{x}_*, \boldsymbol{x}_i))_i, \\ \sigma_*^2 &= K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^T(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_*. \end{aligned}$$

The hyperparameters can be adjusted by maximising the marginal likelihood $P(\boldsymbol{y}) = N(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K} + \sigma^2\boldsymbol{I})$, or if priors are placed on the hyperparameters one can sample from the hyperposterior using MCMC methods. However, in general these methods have time and memory requirements which are clearly superlinear in n, and predictions are at least $O(n)$ per test point.

2 Sparse Gaussian Process Regression

We first introduce in Section 2.1 the likelihood approximation leading to the sparse Gaussian process regression (GPR) method we are interested in. In Section 2.2 we discuss our method for selecting the active set I. In Section 2.3 we motivate model selection by optimisation of the approximate log marginal likelihood.

2.1 Sparse Likelihood Approximation

Our sparse GPR approximation can be understood as *likelihood approximation*. Let $I \subset \{1, \dots, n\}$ denote the active set, and write $R = \{1, \dots, n\} \setminus I$ for the indexes of the remaining points. Our approximation is obtained by replacing the likelihood $P(\boldsymbol{y}|\boldsymbol{u})$ by

$$Q(\boldsymbol{y}|\boldsymbol{u}_I) = N(\boldsymbol{y}|\boldsymbol{P}_I^T\boldsymbol{u}_I, \sigma^2\boldsymbol{I}), \ \boldsymbol{P}_I = \boldsymbol{K}_I^{-1}\boldsymbol{K}_{I,\cdot}.$$

Here, $\boldsymbol{P}_I^T$ is a matrix which maps $\boldsymbol{u}_I$ to the prior conditional mean $\mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_I]$. A strong motivation for this approximation is given by Csató (2002), showing that the minimisation of the relative entropy[2] $\mathrm{D}[\cdot \,\|\, P(\boldsymbol{u}|\boldsymbol{y})]$ over *all* distributions of the form $\propto P(\boldsymbol{u})R(\boldsymbol{u}_I)$, R positive and dependent on $\boldsymbol{u}_I$ only, is attained by $R(\boldsymbol{u}_I) = Q(\boldsymbol{y}|\boldsymbol{u}_I)$. Note that the choice corresponds to the following "non-standard" sampling model. In order to generate $\boldsymbol{y}$, we first sample $\boldsymbol{u}_I \sim P(\boldsymbol{u}_I)$. While the full GPR sampling model would continue sampling $\boldsymbol{u}_R \sim P(\boldsymbol{u}_R|\boldsymbol{u}_I)$, we require a likelihood term which does not depend on $\boldsymbol{u}_R$. The conditional distribution $P(\boldsymbol{u}_R|\boldsymbol{u}_I)$ has mean $\hat{\boldsymbol{u}}_R = (\boldsymbol{P}_I^T\boldsymbol{u}_I)_R$, thus in order to approximate the full likelihood $P(\boldsymbol{y}|\boldsymbol{u})$, it seems sensible to use $P(\boldsymbol{y}|\hat{\boldsymbol{u}}) = N(\boldsymbol{y}|\hat{\boldsymbol{u}}, \sigma^2\boldsymbol{I})$, where $\hat{\boldsymbol{u}} = \boldsymbol{P}_I^T\boldsymbol{u}_I$ (note that $\hat{\boldsymbol{u}}_I = \boldsymbol{u}_I$). Thus, $\boldsymbol{y} \sim N(\boldsymbol{y}|\hat{\boldsymbol{u}}, \sigma^2\boldsymbol{I})$. In the following, instead of dealing with the true posterior $P(\cdot|\boldsymbol{y})$, we will use an approximation $Q(\cdot|\boldsymbol{y})$ obtained by replacing the true likelihood $P(\boldsymbol{y}|\boldsymbol{u})$ by $Q(\boldsymbol{y}|\boldsymbol{u}_I) = N(\boldsymbol{y}|\hat{\boldsymbol{u}}, \sigma^2\boldsymbol{I})$. For fixed I of size k, let $\boldsymbol{K}_I = \boldsymbol{L}\boldsymbol{L}^T$ be decomposed in Cholesky factors and let $\boldsymbol{V} = \boldsymbol{L}^{-1}\boldsymbol{K}_{I,\cdot}$, $\boldsymbol{M} = \sigma^2\boldsymbol{I} + \boldsymbol{V}\boldsymbol{V}^T$. The exact posterior distribution for $\boldsymbol{u}_I$ is given by $P(\boldsymbol{u}_I|\boldsymbol{y}) \propto P(\boldsymbol{y}|\boldsymbol{u}_I)P(\boldsymbol{u}_I)$. Replacing $P(\boldsymbol{y}|\boldsymbol{u}_I)$ with $Q(\boldsymbol{y}|\boldsymbol{u}_I)$ and carrying out some manipulations we obtain

$$Q(\boldsymbol{u}_I|\boldsymbol{y}) = N(\boldsymbol{u}_I|\boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{V}\boldsymbol{y}, \sigma^2\boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{L}^T).$$

Below we will frequently omit the conditioning on $\boldsymbol{y}$ and write $Q(\cdot) = Q(\cdot|\boldsymbol{y})$. Let $\boldsymbol{M} = \boldsymbol{L}_M\boldsymbol{L}_M^T$ be the Cholesky decomposition of $\boldsymbol{M}$, and define $\boldsymbol{\beta} = \boldsymbol{L}_M^{-1}\boldsymbol{V}\boldsymbol{y}$. The predictive distribution at a test point

[1]Whenever we use a bold symbol $\boldsymbol{g}$ or $\boldsymbol{G}$ for a vector or matrix, we denote its components by the corresponding normal symbols g_i and $g_{i,j}$.

[2]Also known as *Kullback-Leibler divergence.*

$\boldsymbol{x}_*$ is given by $Q(u_*|\boldsymbol{y}) = N(u_*|\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$ with

$$\mu(\boldsymbol{x}_*) = \boldsymbol{l}_{M,*}^T \boldsymbol{\beta},$$
$$\sigma^2(\boldsymbol{x}_*) = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \|\boldsymbol{l}_*\|^2 + \sigma^2 \|\boldsymbol{l}_{M,*}\|^2,$$

where $\boldsymbol{l}_* = \boldsymbol{L}^{-1}(K(\boldsymbol{x}_*, \boldsymbol{x}_i))_{i \in I}$, $\boldsymbol{l}_{M,*} = \boldsymbol{L}_M^{-1} \boldsymbol{l}_*$. If only $\mu(\boldsymbol{x}_*)$ is required, $\boldsymbol{L}^{-T} \boldsymbol{L}_M^{-T} \boldsymbol{\beta}$ should be precomputed, after which the mean can be computed in $O(d)$ per test pattern.

The likelihood approximation defined above has been considered by a number of authors (e.g., Smola and Bartlett, 2001, Csató et al., 2002, Luo and Wahba, 1997), although with different motivations. Lawrence et al. (2002) use a simpler approximation, leading to a very efficient method for GP classification.

2.2 Algorithm for Subset Selection

We have to approach three questions:

- How to include a new point into the active set?
- How to construct a selection score for the greedy selection of inclusion candidates, which can be evaluated in $O(1)$ per point?
- How to compute a sparse approximation to the log marginal likelihood and its gradient to do Bayesian model selection?

We discuss the first two issues here and the third one in Subsection 2.3. With some hindsight, we define $\boldsymbol{p} = \operatorname{diag} \boldsymbol{V}^T \boldsymbol{V}$, $\boldsymbol{q} = \operatorname{diag} \boldsymbol{V}^T \boldsymbol{M}^{-1} \boldsymbol{V}$. We also maintain the mean $\boldsymbol{\mu}$ of the posterior approximation $Q(\boldsymbol{u})$. In order to include $i \notin I$ into I, we have to update $\boldsymbol{L} \to \boldsymbol{L}' \in \mathbb{R}^{d+1,d+1}$, $\boldsymbol{V} \to \boldsymbol{V}' \in \mathbb{R}^{d+1,n}$, $\boldsymbol{p} \to \boldsymbol{p}'$, $\boldsymbol{L}_M \to \boldsymbol{L}_M' \in \mathbb{R}^{d+1,d+1}$, $\boldsymbol{\beta} \to \boldsymbol{\beta}' \in \mathbb{R}^{d+1}$ and $\boldsymbol{\mu} \to \boldsymbol{\mu}'$, as shown in Appendix A.1. The time for an inclusion is dominated by three $n \cdot d$ operations and the computation of $\boldsymbol{K}_{\cdot,i}$, and we require one $n \cdot d$ buffer for $\boldsymbol{V}$.

In Lawrence et al. (2002), the use of greedy selection heuristics is motivated, of a kind originally proposed for *active learning*[3], a related yet somewhat harder problem in which one has to select (amongst a pool of unlabeled data) points whose labels may prove most valueable w.r.t. learning progress (e.g., Seung et al., 1998). The "informativeness" of an input point $\boldsymbol{x}_i$ may be scored by the *information gain*

$$\mathrm{D}[Q^{new}(\boldsymbol{u}) \,\|\, Q(\boldsymbol{u})], \tag{1}$$

where Q^{new} is the posterior approximation *after* inclusion of i into I. To understand this score, imagine encoding samples from $Q^{new}(\boldsymbol{u})$. If we switched our coding scheme from being based on Q to using the true distribution, the score value would be the amount of nats per sample point we could save. However, in order to compute this score we require time $O(dn)$ and the evaluation of the kernel matrix column i, a scaling which we consider prohibitive in the context of sparse approximation methods. Note that other scores could be considered as well, such as the entropy difference $\mathrm{H}[Q] - \mathrm{H}[Q^{new}]$, the expected log likelihood $\mathrm{E}_Q[\log P(y_i|u_i)]$ or the value of $\log Q^{new}(\boldsymbol{u})$ at its mode, yet all of them suffer from the same unfavourable scaling. If $I' = I \cup \{i\}$ and $R' = R \setminus \{i\}$, we can obtain an *approximation* to the information gain by simply replacing $Q(\boldsymbol{y}|\boldsymbol{u}_{I'})$ in the definition of Q^{new} by

$$N(\boldsymbol{y}_{\setminus i} | (\boldsymbol{V}_{\cdot,\setminus i})^T \boldsymbol{L}^{-1} \boldsymbol{u}_I, \sigma^2 \boldsymbol{I}) N(y_i | u_i, \sigma^2) \tag{2}$$

(recall that $\boldsymbol{V}_{\cdot,\setminus i}$ is obtained from $\boldsymbol{V}$ by removing the i-th column). One can view the $\hat{u}_i = (\boldsymbol{P}_I^T \boldsymbol{u}_I)_i$ as "pseudo-variables", in which case we simply replace the "pseudo-site" $N(y_i|\hat{u}_i, \sigma^2)$ in the likelihood by the true site $N(y_i|u_i, \sigma^2)$ (see also Csató et al., 2002). By doing so, we ignore that upon proper inclusion, u_i would be coupled with the targets of *all* remaining points in the likelihood, not just with y_i, yet it is exactly this coupling which causes the high evaluation costs. It is clear how this method can be generalised to allow couplings between u_i and *some* of the targets, although we do not pursue this here for simplicity[4]. Now, the information gain (1) can be approximated as

$$\Delta_i = -\log \frac{\sigma}{l_i} - \frac{1}{2} \Big(\log \xi_i + \xi_i (1 - \kappa_i) \sigma^{-2} (y_i - \mu_i)^2 - \kappa_i + 2 \Big), \quad \kappa_i = \xi_i \left(1 + 2(\sigma/l_i)^2\right), \tag{3}$$

where $l_i = (K(\boldsymbol{x}_i, \boldsymbol{x}_i) - p_i)^{1/2}$, $\xi_i = ((\sigma/l_i)^2 + 1 - q_i)^{-1}$. The rather lengthy derivation can be found in (Seeger, 2002). Note that Δ_i can be evaluated in $O(1)$, given p_i, q_i, μ_i, and requires the evaluation of $K(\boldsymbol{x}_i, \boldsymbol{x}_i)$ only. Also note that the replacement of $Q(\boldsymbol{y}|\boldsymbol{u}_{I'})$ against (2) is used only to compute the Δ_i, $i \in R$. Once an inclusion candidate has been chosen, it is included into I by properly updating the likelihood approximation to $Q(\boldsymbol{y}|\boldsymbol{u}_{I'})$ (see Appendix A.1).

Our scheme for first-level inference conditioned on the hyperparameters includes points one at a time into

[3] Active learning is applied to support vector machines in Tong and Koller (2001), however in a quite restricted classification context and without underlying probabilistic model.

[4] For example, it may be acceptable to evaluate row $\boldsymbol{K}_{i,\cdot}$, or a part of it, after which we could retain couplings corresponding to the u_j with the highest prior correlation with u_i.

I, selecting the next inclusion candidate among all remaining points in R as the maximizer of the approximate information gain (3). The final size of the active set can either be fixed in advance, or we can use a stopping criterion based on monitoring the average squared error curve of the successive predictors on the remaining points in R or on a holdout set. The former requires the knowledge of the marginals $Q(u_j) = N(u_j|\mu_j, \sigma_j^2)$, $j \in R$, where $\sigma_j^2 = 1 - p_j + \sigma^2 q_j$.

2.3 Adjusting the Hyperparameters

Our likelihood approximation implies the following approximation to the marginal likelihood:

$$Q(\boldsymbol{y}) = N(\boldsymbol{y}|\boldsymbol{0}, \sigma^2\boldsymbol{I} + \boldsymbol{V}^T\boldsymbol{V}). \tag{4}$$

This can be compared directly to the marginal likelihood for the full GP model which is $P(\boldsymbol{y}) = N(\boldsymbol{y}|\boldsymbol{0}, \sigma^2\boldsymbol{I} + \boldsymbol{K})$ (see Section 1.1), by noting that $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{K}_{\cdot,I}\boldsymbol{K}_I^{-1}\boldsymbol{K}_{\cdot,I}^T$. If $\phi = -\log Q(\boldsymbol{y})$, then the (approximate) negative log marginal posterior for hyperparameters $\boldsymbol{\theta}$ is $\phi(\boldsymbol{\theta}) - \log P(\boldsymbol{\theta})$, and we can try to choose $\boldsymbol{\theta}$ as (local) minimiser of this criterion (in our case, $\boldsymbol{\theta}$ includes σ^2 and the kernel parameters). Here, $P(\boldsymbol{\theta})$ is a hyperprior which can be used to confine $\boldsymbol{\theta}$ to a certain range, to enforce sparsity amongst components of $\boldsymbol{\theta}$, etc. To this end, we have to compute ϕ and its gradient w.r.t. hyperparameters, as stated in Appendix A.2. We then feed these into a custom gradient-based optimizer such as Quasi-Newton. A persistent difficulty in practice is that the criterion ϕ and its gradient will fluctuate with changing I. It is sensible to keep I fixed during line searches and only reselect it when computing gradients for new search directions. Because of these fluctuations, we cannot expect a smooth convergence in general, and assessing when to stop is a tricky issue not yet resolved completely.

3 Related Work

The likelihood approximation we employ here has been used by several authors (e.g., Csató et al., 2002, Smola and Bartlett, 2001). In fact, Csató et al. (2002) consider GP models with arbitrary non-Gaussian noise distributions, employing sparse Gaussian approximations to the intractable posterior $P(\boldsymbol{u}|\boldsymbol{y})$. The approximate information gain (and other approximate criteria) can be derived for this framework in much the same way as shown here, and experiments with the resulting general greedy scheme are subject to future work, although it is substantially more expensive[5] than the special case we deal with here. Smola and Bartlett (2001) and Luo and Wahba (1997) use greedy forward selection schemes just as we do, although our framework is somewhat different. The major difference to our scheme is, however, that they employ very expensive criteria, having evaluation costs of $O(n\,d)$, while we use an $O(1)$ criterion. In fact, in these schemes, the scoring of a point is about as expensive as its inclusion into I. While we can score all remaining points for every inclusion in $O(n)$, Smola and Bartlett (2001) have to resort to randomizations. After transforming to $\boldsymbol{\alpha} = \boldsymbol{K}^{-1}\boldsymbol{u}$, they minimise

$$\begin{aligned}\pi(\boldsymbol{\alpha}) &= \frac{1}{2}\boldsymbol{\alpha}^T(\sigma^2\boldsymbol{K} + \boldsymbol{K}^T\boldsymbol{K})\boldsymbol{\alpha} - \boldsymbol{y}^T\boldsymbol{K}\boldsymbol{\alpha} \\ &= -(\boldsymbol{V}\boldsymbol{y})^T\boldsymbol{L}^{-1}\boldsymbol{u}_I + \frac{1}{2}\boldsymbol{u}_I^T(\boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{L}^T)^{-1}\boldsymbol{u}_I,\end{aligned}$$

where $\boldsymbol{\alpha}$ is sparse in the sense that $\boldsymbol{\alpha}_R = \boldsymbol{0}$. Note that $\pi(\boldsymbol{\alpha})$ is proportional to $-\log P(\boldsymbol{\alpha}|\boldsymbol{y})$. For fixed I, $\pi(\boldsymbol{\alpha})$ is proportional[6] to the negative log posterior $-\log P(\boldsymbol{u}_I|\boldsymbol{y})$ and is minimized by the posterior mean $\boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{V}\boldsymbol{y}$, the minimum value is $\pi_I = \min_{\boldsymbol{\alpha}} \pi(\boldsymbol{\alpha}) = -(1/2)\boldsymbol{\beta}^T\boldsymbol{\beta}$, where the minimum is over $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}_R = \boldsymbol{0}$. Their selection heuristic for a point $i \notin I$ is

$$\Delta_i^{SB} = \pi_I - \pi_{I'} = \frac{1}{2}\beta_{d+1}^2, \quad I' = I \cup \{i\},$$

i.e. the decrease that can be obtained in $\pi(\boldsymbol{\alpha})$ by allowing component i in $\boldsymbol{\alpha}$ to become non-zero. $\boldsymbol{\beta}' = (\boldsymbol{\beta}^T \;\; \beta_{d+1})^T$ is obtained as shown in Appendix A.1. Since this computation is $O(n\,d)$, they can only afford to score a fixed number $k \ll n$ of randomly chosen points from R for each inclusion. Even then, the overall complexity is now dominated by the criterion evaluations: $O(k\,d^2\,n)$, which is about k times the requirements of our method (Smola and Bartlett, 2001, recommend $k = 59$). Furthermore, since the random subsets fluctuate freely over $\{1, \dots, n\}$, one typically ends up evaluating a large fraction of the full kernel matrix $\boldsymbol{K}$, which is problematic if kernel evaluations are very expensive. Our scheme requires the evaluation of $\boldsymbol{K}_{\cdot,I}$ only (if I is the final active set).

Our approximation is also somewhat similar to the Bayesian committee machine (BCM) of Tresp (2000) where the information from n training points is absorbed onto a smaller number of points, although in BCM it is the test points and not a subset of the training points that are used[7]. Note that the approximation is *not* equivalent to a Bayesian subset of regressors (SR) model where we put $u(\boldsymbol{x}_*) = \sum_{i \in I} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_*)$

[5] This is due to the nature of the Gaussian approximations, which require the introduction of site parameters for each of the n training points. These parameters have to be updated regularly between inclusions, causing substantial extra costs.

[6] The normalisation constant depends on the active set I.

[7] This is problematic for applications where the test data is not known at training time.

with a prior $\boldsymbol{\alpha}_I \sim N(\mathbf{0}, \boldsymbol{K}_I^{-1})$. The SR model gives the same predictive mean but produces different predictive variances (see Williams et al., 2002, for details), in fact the predictive process $Q(u_*|\boldsymbol{x}_*, \boldsymbol{y})$ is *not* a Gaussian process (as in our scheme) and can be degenerate. The idea of approximating the likelihood using a linear projection from d variables is not restricted to using a d-subset of $\boldsymbol{u}$ (as has been noted in Smola and Bartlett (2001), Csató and Opper (2002)). If we project into the d-th Krylow subspace as constructed by a linear conjugate gradients solver on the matrix $\boldsymbol{K} + \sigma^2 \boldsymbol{I}$, we obtain an approximation suggested in Gibbs (1997), although this is not a sparse approximation, requires the complete $\boldsymbol{K}$ to be stored in memory and is at least $O(n^2\, d)$.

4 Experiments

For the experiments presented here, we chose the datasets *kin-40k* (10000 training, 30000 test cases, 9 attributes) and *pumadyn-32nm* (7168 training, 1024 test cases, 33 attributes),[8] both artificially created using a robot arm simulator, highly non-linear and low-noise. We used the *squared-exponential kernel* $K(\boldsymbol{x}, \boldsymbol{x}') = C \exp(-(1/(2q))(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{W}(\boldsymbol{x} - \boldsymbol{x}')) + v_b$, $\boldsymbol{x} \in \mathbb{R}^q$, $\boldsymbol{W} = \mathrm{diag}(w_j)_j$, with the hyperpriors employed by Rasmussen (1996). The components $w_j^{-1/2}$ determine the typical length-scales of functions from the prior along the Euclidean coordinate axes in input space, thus by driving $w_j \approx 0$, the corresponding dimension is effectively ignored (*automatic relevance determination (ARD)*, e.g., Rasmussen, 1996). Where not stated otherwise, experiments are repeated ten times, and we quote medians (as well as quartiles in the plots). Predictive accuracy is measured in terms of average squared error, i.e. $(1/2)(y_* - \mu(\boldsymbol{x}_*))^2$ averaged over the test set.

4.1 Learning Curves

Here, we compare full GPR against a range of sparse methods: sparse GPR using our criterion Δ_i (*info-gain*; #1); using purely random selection (*random*; #2); using the criterion Δ_i^{SB} of Smola and Bartlett (2001) (*smo-bart*; #3). Hyperparameters were adjusted by maximising their (marginal) log posterior for full GPR over a random subset of size n_{MS} of the training set. Figure 1 shows learning curves for kin-40k, $n_{MS} = 2000$ (note that each plot contains upper and lower quartiles for full GPR as horizontal lines) and Table 1 gives the corresponding training times.

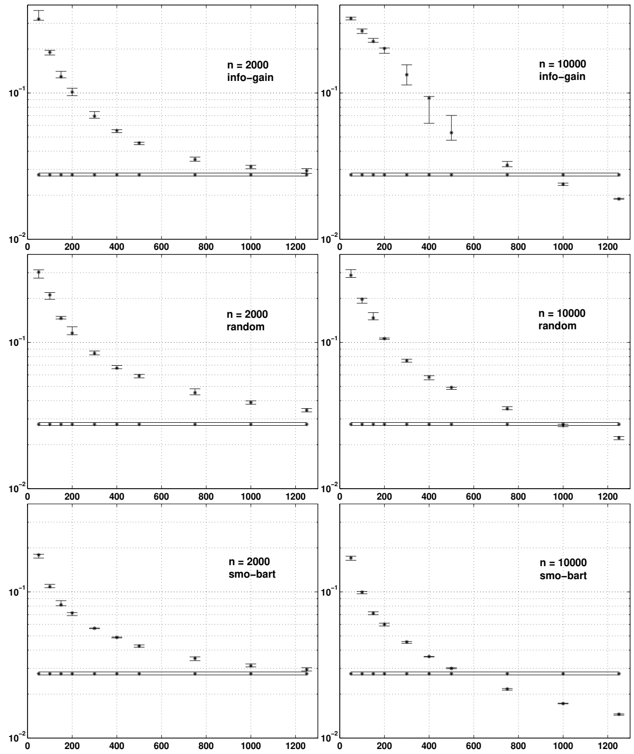

Figure 1: Learning curves for sparse GPR on kin-40k. Left: sparse methods use MS training sets of full GPR, $n = 2000$. Right: sparse methods use full training set, $n = 10000$. X-axis: Active set size d. Y-axis: Average squared error. Horizontal lines: quartiles for full GPR, $n = 2000$.

	Time (secs) for act. set size d						
	100	200	300	400	500	1000	1250
	kin-40k, $n = 2000$						
#1	1.0	2.0	5.0	9.0	13.5	53.5	85.0
#2	1.0	2.0	4.0	7.0	11.0	48.0	76.0
#3	17.0	54.0	110.0	185.0	281.0	1088.0	1714.5
	kin-40k, $n = 10000$						
#1	5.0	13.0	25.5	42.0	62.5	230.0	352.0
#2	3.0	10.0	21.0	35.5	55.0	215.0	338.5
#3	88.0	265.0	530.5	885.0	1327.5	4977.0	7794.5

Table 1: Training time (secs) for methods info-gain (#1), random (#2), smo-bart (#3).

For small sizes $d = |I|$, *smo-bart* outperforms the other methods, while for larger d, *random* seems less effective. We also see (from curves on the right) that sparse methods on the full training set ($n = 10000$) can significantly outperform full GPR on a subset ($n = 2000$), even though the former employ sparser expansions ($d = 1000, 1250$). In this case, the freedom of select-

[8] *kin-40k*: see *www.igi.tugraz.at/aschwaig/data.html*. *pumadyn-32nm*: see *www.cs.toronto/~delve*. We preprocessed both by subtracting off a linear regression fit and normalizing all attributes to unit variance. Thus, a linear regression on these tasks would result in averaged squared errors ≈ 0.5.

ing amongst more points outweights the advantage of using a larger expansion. While *random* and *info-gain* have similar training times, *smo-bart* is about thirty times slower (we used selection sets of size $k = 30$ for *smo-bart*), which probably precludes the latter being used with extensive model selection.

The set *pumadyn-32nm* has a lot of irrelevant attributes which, in the context of GPR, have to be identified in order to achieve good performance. Adapting the hyperparameters for $n_{MS} = 1024$ using full GPR leads to four attributes being singled out, all other $w_j \approx 0$. The median of the average squared test errors is 0.0225.[9] Note that traditional techniques like cross-validation are not applicable in this situation, due to the large number (35) of hyperparameters. Figure 2 shows learning curves for the sparse methods.

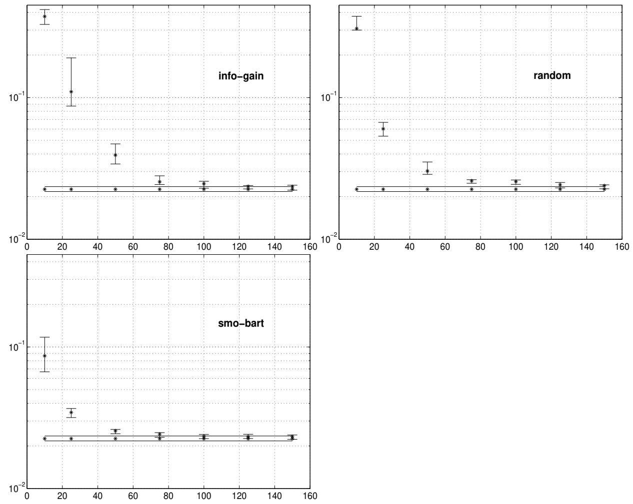

Figure 2: Learning curves for sparse GPR on pumadyn-32nm.

Here, all three methods achieve an accuracy comparable to full GPR after the inclusion of $d = 125$ of the $n = 7168$ training cases, but while the training time median for full GPR is $1102.5s$, *info-gain*, *random* and *smo-bart* require $3s, 2s$ and $66s$ respectively. For unreasonably small values of d, *info-gain* exhibits some fluctuations, leading to a worse performance than *random*.

4.2 Sparse Model Selection

We now try to assess the procedure of maximising the approximation of the (marginal) hyperposterior to adjust σ^2 and the kernel parameters (see section 2.3). In a first experiment, we follow the trajectory in hyperparameter space used by full GPR training (on pumadyn-32nm, $n_{MS} = 1024$) and compute the corresponding approximate criterion values for *info-gain*, *random* for different d. Due to the long running times, *smo-bart* is not considered here. Figure 3 shows the criterion curves for *info-gain* (left) and *random* (right).

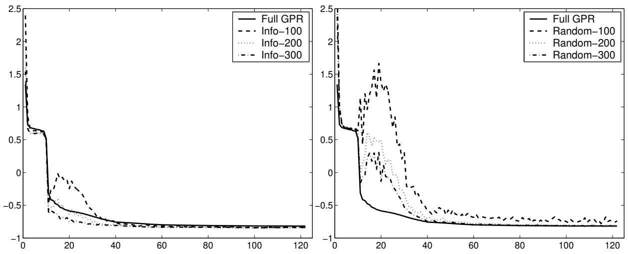

Figure 3: Criterion for full GPR and sparse approximations (evaluated at the same hyperparameters). X-axis: Iterations.

Along critical parts of the trajectory, the approximation given by *random* is fairly poor (for $d = 100, 200$), while the *info-gain* approximation seems acceptable. Note that for both methods, I is re-selected in every iteration, therefore the poor approximation by *random* cannot be attributed to random fluctuations.

In the final experiment, we first trained full GPR on pumadyn-32nm, using $n_{MS} = 2048$. We then ran an identical setup,[10] however using the sparse approximate criterion together with *info-gain*, *random* and a variant of the latter, named *fixed*, in which we select I at random only once and keep it fixed during the optimisation. We used active set sizes $d = 50, 100, 200, 500$. Only two different profiles were observed: either, the "relevant" attributes mentioned above were singled out (ARD) and the squared error was below 0.03, or the method failed with error close to 0.5. In fact, *random* and *fixed* for $d = 50, 100, 200$ failed in all runs, and *info-gain* for $d = 50$ was successful just once. For all other combinations, medians of average squared error, number of iterations and optimisation running time over the successful runs are given in Table 2 (recall that all experiments are repeated ten times).

While model selection based on *info-gain* was reliable even for small sizes $d = 100, 200$, the runs for *random* often converged (to high accuracy) to poor spu-

[9] To demonstrate the significance of ARD, we ran the same experiment using the *radial basis functions (RBF) kernel* (the squared-exponential kernel with $\boldsymbol{W} = w\boldsymbol{I}$), resulting in a squared error of 0.4730. However, if only the four "relevant" attributes are used, full GPR with the RBF kernel attains a squared error of 0.024.

[10] The initial hyperparameter values were chosen as recommended in Rasmussen (1996). They are not close to a useful minimum point for this task (note the sharp initial drop of the criterion values).
The optimiser was stopped once the relative improvement and gradient size fell below small thresholds, or after 200 iterations.

Method	d	# succ. runs	Squared error	# Iter. optim.	MS time (secs)
full GPR	-	10	0.0236	200	22100.5
info-gain	500	10	0.0296	200	9079
info-gain	200	10	0.0259	177.5	1833.5
info-gain	100	9	0.0252	200	836
random	500	8	0.0244	95	4412
fixed	500	9	0.0239	200	9203

Table 2: Comparison of model selection for full and several sparse GPR methods, dataset pumadyn-32nm, $n_{MS} = 2048$.

rious minima of the criterion approximation. Even in the successful runs, flat plateaus are traversed before a new downwards direction is found (see Figure 4, left).[11] Somewhat surprisingly, *info-gain* with $d = 500$ results in larger squared errors than our scheme for smaller d. In Figure 4, we compare criterion curves for full GPR and sparse methods using a run in which all of them were successful. On the right, we plot the sizes of the largest inverse length scales $w_j^{1/2}$ in the hyperparameter vectors chosen by the different methods (recall that the positions of these within $\boldsymbol{W}$ were the same for all methods)

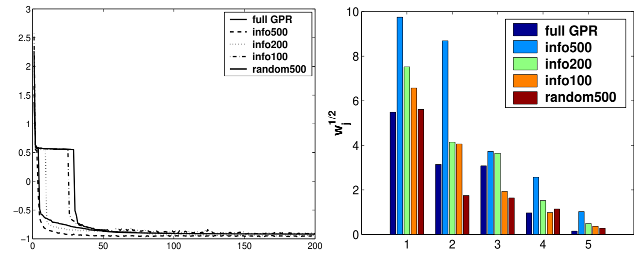

Figure 4: Left: Criterion curves for model selection optimisations (X-axis: Iterations of optimiser); lower solid line: full GPR; upper solid line: *random* ($d = 500$). Right: Largest values of inverse squared length scales w_j in the selected hyperparameter vector.

Note the dangerously flat plateau in the curve for *random* (*fixed* has a very similar curve, not shown here). As for the suboptimal performance of *info-gain* ($d = 500$), one may suspect overfitting[12] behaviour. The curve runs below the one for full GPR, and the concrete $w_j^{1/2}$ values are significantly larger than the ones chosen by the latter. Indeed, training full GPR using the hyperparameters found by *random*, $d = 500$, results in a squared error of 0.0317, suggesting that the hyperparameters are suboptimal. Such problems could probably be alleviated by changing the hyperpriors.

5 Discussion

We have proposed a novel scheme for sparse greedy approximations of GP regression, featuring a very fast forward selection criterion motivated by active learning. The scheme is essentially as fast as selecting the active set I at random. For fixed, well-selected hyperparameter values, small improvements over random selection has been demonstrated for sufficiently large active sets, while our scheme is not robust for very small I.[13] In light of our experiments, a major advantage of our scheme seems that the greedy selection leads to a stable approximation of the log marginal hyperposterior which can be optimised for model selection, even if the size of I is a small fraction of the total training set size only, while for random selection the same setup frequently fails due to strong fluctuations in the approximation. In contrast to the well-known support vector machine, our scheme is a fully probabilistic model from which valid Bayesian confidence intervals can be obtained, and for which a large number of hyperparameters can be adjusted automatically, allowing for example for feature selection via ARD (see section 4).

In future work, we will apply our scheme to harder regression tasks and explore the effectiveness of stopping criteria motivated in Subsection 2.2 in order to find good active set sizes. Furthermore, extensions of our scheme to settings where Bayesian inference for the full GP model is not analytically tractable (such as classification or regression with non-Gaussian noise) will be considered (note that such an extension has been proposed in Csató et al., 2002). Sparse approximations to criteria like the log marginal hyperposterior could also be used to speed up exact algorithms such as hybrid MCMC sampling.

Acknowledgments

MS gratefully acknowledges support through a research studentship from *Microsoft Research Ltd.* We thank Lehel

[11] Somewhat surprisingly, *random* does not behave better than *fixed* w.r.t. spurious converge (for $d = 500$, they both fail in the same run). We would have expected *random* to escape spurious minima more readily.

[12] In general, empirical Bayesian methods such as ours here can lead to overfitting. If approximations are used, it is always desirable that by increasing the complexity of the approximation, we are *guaranteed* to obtain a closer approximation of the true criterion. This is the case for certain variational approximations (if one ignores local minima issues which often plague such methods), but it is *not* true for the sparse GP approximations considered here (for example, the sparse approximation to the log hyperposterior is not a bound, see Figure 3).

[13] This may be due to it focussing on outliers, but further experiments are required to clarify this point.

Csató and Manfred Opper for helpful discussions, and Anton Schwaighofer for providing the *kin-40k* dataset.

References

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

Lehel Csató and Manfred Opper. Sparse online Gaussian processes. *Neural Computation*, 14:641–668, 2002.

Lehel Csató, Manfred Opper, and Ole Winther. TAP Gibbs free energy, belief propagation and sparsity. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 657–663. MIT Press, 2002.

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

Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. To appear in Neural Information Processing Systems 15, 2002.

T. Leen, T. Dietterich, and V. Tresp, editors. *Advances in Neural Information Processing Systems 13*, 2001. MIT Press.

Z. Luo and G. Wahba. Hybrid adaptive splines. *Journal of the American Statistical Association*, 92:107–116, 1997.

C. E. Rasmussen. *Evaluation of Gaussian Processes and Other Methods for Nonlinear Regression.* PhD thesis, University of Toronto, 1996.

Matthias Seeger. Fast forward selection to speed up sparse Gaussian process regression. See `www.dai.ed.ac.uk/~seeger/papers.html`, 2002.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Conference on Computational Learning Theory 5*, pages 287–294. Morgan Kaufmann, 1998.

Alex Smola and Peter Bartlett. Sparse greedy Gaussian process regression. In Leen et al. (2001), pages 619–625.

Michael Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

Volker Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

Christopher K. I. Williams, C. E. Rasmussen, A. Schwaighofer, and V. Tresp. Observations on the Nyström method for Gaussian process prediction. University of Edinburgh. See `www.dai.ed.ac.uk/homes/ckiw/`, 2002.

Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In Leen et al. (2001), pages 682–688.

A Some Details

A.1 Inclusion of a Point into I

Recall the definitions at the beginning of section 2. The update of a Cholesky factor is standard. First, $(\boldsymbol{L}')^T_{d+1,\cdot} = \boldsymbol{L}^{-1}\boldsymbol{K}_{I,i} = \boldsymbol{v}_i = \boldsymbol{V}_{\cdot,i}$, and $l_i = (\boldsymbol{L}')_{d+1,d+1} = (K(\boldsymbol{x}_i,\boldsymbol{x}_i) - p_i)^{1/2}$. Then, $(\boldsymbol{V}')_{1\dots d,\cdot} = \boldsymbol{V}$, and if $\boldsymbol{v} = (\boldsymbol{V}')^T_{d+1,\cdot}$, then $\boldsymbol{v} = l_i^{-1}(\boldsymbol{K}_{\cdot,i} - \boldsymbol{V}^T\boldsymbol{v}_i)$, and $\boldsymbol{p}' = \boldsymbol{p} + (v_j^2)_j$. Next, $(\boldsymbol{L}'_M)_{d+1,1\dots d} = \boldsymbol{l}_M = \boldsymbol{L}_M^{-1}\boldsymbol{V}\boldsymbol{v}$ and $l_{M,i} = (\boldsymbol{L}'_M)_{d+1,d+1} = (\sigma^2 + \boldsymbol{v}^T\boldsymbol{v} - \boldsymbol{l}_M^T\boldsymbol{l}_M)^{1/2}$. Finally, $\boldsymbol{w} = ((\boldsymbol{L}'_M)^{-1}\boldsymbol{V}')^T_{d+1,\cdot} = l_{M,i}^{-1}(\boldsymbol{v} - \boldsymbol{V}^T\boldsymbol{L}_M^{-T}\boldsymbol{l}_M)$ and $\boldsymbol{q}' = \boldsymbol{q} + (w_j^2)_j$.

For the update of $\boldsymbol{\mu}$, note that $\boldsymbol{\mu} = \boldsymbol{V}^T\boldsymbol{L}_M^{-T}\boldsymbol{\beta}$, thus $\boldsymbol{m}' = \boldsymbol{m} + \beta_{d+1}\boldsymbol{w}$, where $\beta_{d+1} = (\boldsymbol{\beta}')_{d+1} = l_{M,i}^{-1}(\boldsymbol{v}^T\boldsymbol{y} - \boldsymbol{\beta}^T\boldsymbol{l}_M)$ is the new component of vector $\boldsymbol{\beta}'$ (note that $\boldsymbol{v}^T\boldsymbol{y} = (\boldsymbol{V}'\boldsymbol{y})_{d+1}$).

A.2 The Negative Log Marginal Likelihood and its Gradient

Recall the marginal likelihood approximation from (4). Here, we show how ϕ and its gradient can be computed, the detailed derivations are given in (Seeger, 2002). We have $(\sigma^2\boldsymbol{I} + \boldsymbol{V}^T\boldsymbol{V})^{-1} = \sigma^{-2}(\boldsymbol{I} - \boldsymbol{V}^T\boldsymbol{M}^{-1}\boldsymbol{V})$ and $\log|\sigma^2\boldsymbol{I} + \boldsymbol{V}^T\boldsymbol{V}| = \log|\boldsymbol{M}| - 2(n-d)\log\sigma^{-1}$. Thus

$$\phi = \frac{1}{2}\left(\log|\boldsymbol{M}| + (n-d)\log\sigma^2 + \frac{1}{\sigma^2}\left(\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{\beta}^T\boldsymbol{\beta}\right)\right)$$

up to an additive constant. The derivative w.r.t. $\log(\sigma^2)$ is given by

$$\frac{\partial\phi}{\partial\log(\sigma^2)} = -\frac{1}{2}\Big(\sigma^2\operatorname{tr}\boldsymbol{M}^{-1} + n - d + \|\boldsymbol{L}_M^{-T}\boldsymbol{\beta}\|^2 - \frac{1}{\sigma^2}\left(\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{\beta}^T\boldsymbol{\beta}\right)\Big).$$

Define $\boldsymbol{A} = \sigma^2\boldsymbol{K}_I + \boldsymbol{K}_{I,\cdot}\boldsymbol{K}_{I,\cdot}^T$ and note that $\boldsymbol{A} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T$, where $\tilde{\boldsymbol{L}} = \boldsymbol{L}\boldsymbol{L}_M$. Let $\boldsymbol{B}_1 = \tilde{\boldsymbol{L}}^{-T}\boldsymbol{L}_M^{-1}\boldsymbol{V} = \boldsymbol{A}^{-1}\boldsymbol{K}_{I,\cdot}$ and $\boldsymbol{b}_1 = \tilde{\boldsymbol{L}}^{-T}\boldsymbol{\beta}$. Then, if θ is one of the kernel parameters, the derivative of ϕ is given by

$$\frac{\partial\phi}{\partial\theta} = -\frac{1}{2}\operatorname{tr}\left(\boldsymbol{K}_I^{-1} - \sigma^2\boldsymbol{A}^{-1}\right)\dot{\boldsymbol{K}}_I + \operatorname{tr}\boldsymbol{B}_1^T\dot{\boldsymbol{K}}_{I,\cdot} - \sigma^{-2}\boldsymbol{b}_1^T\dot{\boldsymbol{K}}_{I,\cdot}(\boldsymbol{y} - \boldsymbol{m}) + \frac{1}{2}\boldsymbol{b}_1^T\dot{\boldsymbol{K}}_I\boldsymbol{b}_1, \tag{5}$$

where $\dot{\boldsymbol{K}}_I = (\partial/\partial\theta)\boldsymbol{K}_I$, etc. The computation of $\boldsymbol{B}_1$ dominates the costs, requiring n backsubstitutions with each of $\boldsymbol{L}_M$ and $\tilde{\boldsymbol{L}}^T$. Note that $\boldsymbol{B}_1$ may overwrite $\boldsymbol{V}$, and that in order to compute (5), it is not necessary to store $\dot{\boldsymbol{K}}_{I,\cdot}$ explicitely.