Approximation Methods for Gaussian Process Regression

Joaquin Quiñonero-Candela
Applied Games, Microsoft Research Ltd.
7 J J Thomson Avenue, CB3 0FB Cambridge, UK
joaquinc@microsoft.com

Carl Edward Ramussen
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, D-72076 Tübingen, Germany
carl@tuebingen.mpg.de

Christopher K. I. Williams
Institute for Adaptive and Neural Computation
School of Informatics, University of Edinburgh
5 Forrest Hill, Edinburgh EH1 2QL, Scotland, UK
c.k.i.williams@ed.ac.uk

May 21, 2007

Abstract

A wealth of computationally efficient approximation methods for Gaussian process regression have been recently proposed. We give a unifying overview of sparse approximations, following Quiñonero-Candela and Rasmussen (2005), and a brief review of approximate matrix-vector multiplication methods.

1 Introduction

Gaussian processes (GPs) are flexible, simple to implement, fully probabilistic methods suitable for a wide range of problems in regression and classification. A recent overview of GP methods is provided by Rasmussen and Williams (2006). Gaussian processes allow a Bayesian use of kernels for learning, with the following two key advantages:

- GPs provide fully probabilistic predictive distributions, including estimates of the uncertainty of the predictions.

- The evidence framework applied to GPs allows to learn the parameters of the kernel.

However, a serious problem with GP methods is that the naïve implementation requires computation which grows as $O(n^3)$, where n is the number of training cases. A host of approximation techniques have recently been proposed to allow application of GPs to large problems in machine learning. These techniques can broadly be divided into two classes: 1) those based on *sparse* methods, which approximate the full posterior by expressions involving matrices of lower rank $m < n$, and 2) those relying on approximate matrix-vector multiplication (MVM) conjugate gradient (CG) methods.

The structure of this paper is as follows: In section 2 we provide an overview of Gaussian process regression. In section 3 we review sparse approximations under the unifying view, recently proposed by Quiñonero-Candela and Rasmussen (2005), based on inducing inputs and conditionally independent approximations to the prior. Section 4 describes work on approximate MVM methods for GP regression. The problem of selecting the inducing inputs is discussed in section 5. We address methods for setting hyperparameters in the kernel in section 6. The main focus of the chapter is on GP regression, but in section 7 we briefly describe how these methods can be extended to the classification problem. Conclusions are drawn in section 8.

2 Gaussian Process Regression

Probabilistic regression is usually formulated as follows: given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$ of n pairs of (vectorial) inputs $\mathbf{x}_i$ and noisy (real, scalar) outputs y_i, compute the predictive distribution of the function values f_* (or noisy y_*) at test locations $\mathbf{x}_*$. In the simplest case (which we deal with here) we assume that the noise is additive, independent and Gaussian, such that the relationship between the (latent) function $f(\mathbf{x})$ and the observed noisy targets y are given by

$$y_i \;=\; f(\mathbf{x}_i) + \varepsilon_i \;, \qquad \text{where} \qquad \varepsilon_i \;\sim\; \mathcal{N}(0,\, \sigma^2_{\text{noise}}) \;, \tag{1}$$

where σ^2_{noise} is the variance of the noise, and we use the notation $\mathcal{N}(\mathbf{a},\, A)$ for the Gaussian distribution with mean $\mathbf{a}$ and covariance A.

Definition 1. *A Gaussian process (GP) is a collection of random variables, any finite number of which have consistent*[1] *joint Gaussian distributions.*

Gaussian process (GP) regression is a Bayesian approach which assumes a GP prior[2] over functions, i.e. that a priori the function values behave according to

$$p(\mathbf{f}|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \;=\; \mathcal{N}(\mathbf{0},\, K) \;, \tag{2}$$

[1] The random variables obey the usual rules of marginalization, etc.

[2] For notational simplicity we exclusively use zero-mean priors.

where $\mathbf{f} = [f_1, f_2, \ldots, f_n]^\top$ is a vector of latent function values, $f_i = f(\mathbf{x}_i)$ and K is a covariance matrix, whose entries are given by the *covariance function*, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Valid covariance functions give rise to positive semidefinite covariance matrices. In general, positive semidefinite kernels are valid covariance functions. The covariance function encodes our assumptions about the the function we wish to learn, by defining a notion of similarity between two function values, as a function of the corresponding two inputs. A very common covariance function is the Gaussian, or squared exponential:

$$K_{ij} \;=\; k(\mathbf{x}_i, \mathbf{x}_j) \;=\; v^2 \exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\lambda^2} \Big) \,, \tag{3}$$

where v^2 controls the prior variance, and λ is an isotropic lengthscale parameter that controls the rate of decay of the covariance, i.e. determines how far away $\mathbf{x}_i$ must be from $\mathbf{x}_j$ for f_i to be unrelated to f_j. We term the parameters that define the covariance functions *hyperparameters*. Learning of the hyperparameters based on data is discussed in section 6.

Note that the GP treats the latent function values f_i as random variables, indexed by the corresponding input. In the following, for simplicity we will always neglect the explicit conditioning on the inputs; the GP model and all expressions are always conditional on the corresponding inputs. The GP model is concerned only with the conditional of the outputs given the inputs; we do not model anything about the distribution of the inputs themselves.

As we will see in later sections, some approximations are strictly equivalent to GPs, while others are not. That is, the implied prior may still be multivariate Gaussian, but the covariance function may be different for training and test cases.

Definition 2. *A Gaussian process is called* degenerate *iff the covariance function has a finite number of non-zero eigenvalues.*

Degenerate GPs (such as e.g. with polynomial covariance function) correspond to *finite* linear (-in-the-parameters) models, whereas non-degenerate GPs (such as e.g. the squared exponential or RBF covariance function) do not. The prior for a finite m-dimensional linear model only considers a universe of at most m linearly independent functions; this may often be too restrictive when $n \gg m$. Note however, that non-degeneracy on its own doesn't guarantee the existence of the "right kind" of flexibility for a given particular modelling task. For a more detailed background on GP models, see for example Rasmussen and Williams (2006).

Inference in the GP model is simple: we put a joint GP prior on training and test latent values, $\mathbf{f}$ and $\mathbf{f}_*$[3], and combine it with the likelihood $p(\mathbf{y}|\mathbf{f})$ using Bayes' rule, to obtain the joint posterior

$$p(\mathbf{f}, \mathbf{f}_*|\mathbf{y}) \;=\; \frac{p(\mathbf{f}, \mathbf{f}_*) p(\mathbf{y}|\mathbf{f})}{p(\mathbf{y})} \,, \tag{4}$$

[3]We will mostly consider a vector of test cases $\mathbf{f}_*$ (rather than a single f_*) evaluated at a set of test inputs $\mathbf{X}_*$.

where $\mathbf{y} = [y_1, \ldots, y_n]$ is the vector of training targets. The final step needed to produce the desired posterior predictive distribution is to marginalize out the unwanted training set latent variables:

$$p(\mathbf{f}_*|\mathbf{y}) \;=\; \int p(\mathbf{f}, \mathbf{f}_*|\mathbf{y}) \mathrm{d}\mathbf{f} \;=\; \frac{1}{p(\mathbf{y})} \int p(\mathbf{y}|\mathbf{f})\, p(\mathbf{f}, \mathbf{f}_*)\, \mathrm{d}\mathbf{f}\;, \tag{5}$$

or in words: the predictive distribution is the marginal of the renormalized joint prior times the likelihood. The joint GP prior and the independent likelihood are both Gaussian

$$p(\mathbf{f}, \mathbf{f}_*) \;=\; \mathcal{N}\Big(\mathbf{0}, \begin{bmatrix} K_{\mathbf{f},\mathbf{f}} & K_{*,\mathbf{f}} \\ K_{\mathbf{f},*} & K_{*,*} \end{bmatrix}\Big), \quad \text{and} \quad p(\mathbf{y}|\mathbf{f}) \;=\; \mathcal{N}(\mathbf{f},\, \sigma^2_{\text{noise}}\mathbf{I})\;, \tag{6}$$

where K is subscript by the variables between which the covariance is computed (and we use the asterisk $*$ as shorthand for $\mathbf{f}_*$) and $\mathbf{I}$ is the identity matrix. Since both factors in the integral are Gaussian, the integral can be evaluated in closed form to give the Gaussian predictive distribution

$$p(\mathbf{f}_*|\mathbf{y}) \;=\; \mathcal{N}\big(K_{*,\mathbf{f}}\,(K_{\mathbf{f},\mathbf{f}}+\sigma^2_{\text{noise}}\mathbf{I})^{-1}\mathbf{y},\; K_{*,*}-K_{*,\mathbf{f}}\,(K_{\mathbf{f},\mathbf{f}}+\sigma^2_{\text{noise}}\mathbf{I})^{-1}K_{\mathbf{f},*}\big)\;. \tag{7}$$

The problem with the above expression is that it requires inversion of a matrix of size $n \times n$ which requires $O(n^3)$ operations, where n is the number of training cases. Thus, the simple exact implementation can handle problems with at most a few thousand training cases on today's desktop machines.

3 Sparse Approximations Based on Inducing Variables

To overcome the computational limitations of GP regression, numerous authors have recently suggested a wealth of *sparse* approximations. Common to all these approximation schemes is that only a subset of the latent variables are treated exactly, and the remaining variables are given some approximate, but computationally cheaper treatment. However, the published algorithms have widely different motivations, emphasis and exposition, so it is difficult to get an overview of how they relate to each other, and which can be expected to give rise to the best algorithms.

A useful discussion of some of the approximation methods is given in chapter 8 of Rasmussen and Williams (2006). In this Section we go beyond this, and provide a unifying view of sparse approximations for GP regression, following Quiñonero-Candela and Rasmussen (2005). For each algorithm we analyze the posterior, and compute the *effective prior* which it is using. Thus, we reinterpret the algorithms as "exact inference with an approximate prior", rather than the existing (ubiquitous) interpretation "approximate inference with the exact prior". This approach has the advantage of directly expressing the approximations in terms of prior assumptions about the function, which makes

the consequences of the approximations much easier to understand. While this view of the approximations is not the only one possible (Seeger, 2003, categorizes sparse GP approximations according to likelihood approximations), it has the advantage of putting all existing probabilistic sparse approximations under one umbrella, thus enabling direct comparison and revealing the relation between them.

We now seek to modify the joint prior $p(\mathbf{f}_*, \mathbf{f})$ from (6) in ways which will reduce the computational requirements from (7). Let us first rewrite that prior by introducing an additional set of m latent variables $\mathbf{u} = [u_1, \ldots, u_m]^\top$, which we call the *inducing variables*. These latent variables are values of the Gaussian process (as also $\mathbf{f}$ and $\mathbf{f}_*$), corresponding to a set of input locations $\mathbf{X}_\mathbf{u}$, which we call the *inducing inputs*. Whereas the additional latent variables $\mathbf{u}$ are always marginalized out in the predictive distribution, the choice of inducing inputs *does* leave an imprint on the final solution. The inducing variables will turn out to be generalizations of variables which other authors have referred to variously as "support points", "active set" or "pseudo-inputs". Particular sparse algorithms choose the inducing variables in various different ways; some algorithms chose the inducing inputs to be a subset of the training set, others not, as we will discuss in Section 5. For now consider any arbitrary inducing variables.

Due to the *consistency* of Gaussian processes, we know that we can recover $p(\mathbf{f}_*, \mathbf{f})$ by simply integrating (marginalizing) out $\mathbf{u}$ from the joint GP prior $p(\mathbf{f}_*, \mathbf{f}, \mathbf{u})$

$$\begin{aligned} p(\mathbf{f}_*, \mathbf{f}) \;&=\; \int p(\mathbf{f}_*, \mathbf{f}, \mathbf{u})\, \mathrm{d}\mathbf{u} \;=\; \int p(\mathbf{f}_*, \mathbf{f}|\mathbf{u})\, p(\mathbf{u})\, \mathrm{d}\mathbf{u}, \\ \text{where} \quad p(\mathbf{u}) \;&=\; \mathcal{N}(\mathbf{0},\, K_{\mathbf{u},\mathbf{u}})\,. \end{aligned} \tag{8}$$

This is an exact expression. Now, we introduce the fundamental approximation which gives rise to almost all sparse approximations. We approximate the joint prior by assuming that $\mathbf{f}_*$ and $\mathbf{f}$ are *conditionally independent given* $\mathbf{u}$, see Figure 1, such that

$$p(\mathbf{f}_*, \mathbf{f}) \;\simeq\; q(\mathbf{f}_*, \mathbf{f}) \;=\; \int q(\mathbf{f}_*|\mathbf{u})\, q(\mathbf{f}|\mathbf{u})\, p(\mathbf{u})\, \mathrm{d}\mathbf{u}\,. \tag{9}$$

The name *inducing* variable is motivated by the fact that $\mathbf{f}$ and $\mathbf{f}_*$ can only communicate though $\mathbf{u}$, and $\mathbf{u}$ therefore *induces* the dependencies between training and test cases. As we shall detail in the following sections, the different computationally efficient algorithms proposed in the literature correspond to different *additional assumptions* about the two approximate *inducing* conditionals $q(\mathbf{f}|\mathbf{u})$, $q(\mathbf{f}_*|\mathbf{u})$ of the integral in (9). It will be useful for future reference to specify here the exact expressions for the two conditionals

$$\text{\textit{training} conditional:} \qquad p(\mathbf{f}|\mathbf{u}) \;=\; \mathcal{N}(K_{\mathbf{f},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u},\, K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}})\,, \tag{10a}$$

$$\text{\textit{test} conditional:} \qquad p(\mathbf{f}_*|\mathbf{u}) \;=\mathcal{N}(K_{*,\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u},\, K_{*,*} - Q_{*,*})\,, \tag{10b}$$

where we have introduced the shorthand notation[4] $Q_{\mathbf{a},\mathbf{b}} \triangleq K_{\mathbf{a},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}K_{\mathbf{u},\mathbf{b}}$. We

[4]Note, that $Q_{\mathbf{a},\mathbf{b}}$ depends on $\mathbf{u}$ although this is not explicit in the notation.

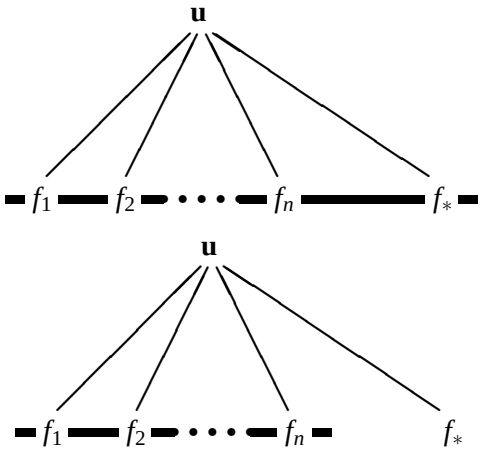

Figure 1: Graphical model of the relation between the inducing variables $\mathbf{u}$, the training latent functions values $\mathbf{f} = [f_1, \ldots, f_n]^\top$ and the test function value f_*. The thick horizontal line represents a set of fully connected nodes. The observations $y_1, \ldots, y_n, y_*$ (not shown) would dangle individually from the corresponding latent values, by way of the exact (factored) likelihood (6). **Upper graph:** the fully connected graph corresponds to the case where no approximation is made to the full joint Gaussian process distribution between these variables. The inducing variables $\mathbf{u}$ are superfluous in this case, since all latent function values can communicate with all others. **Lower graph:** assumption of *conditional independence* between training and test function values given $\mathbf{u}$. This gives rise to the separation between training and test conditionals from (9). Notice that having cut the communication path between training and test latent function values, information from $\mathbf{f}$ can only be transmitted to f_* via the inducing variables $\mathbf{u}$.

can readily identify the expressions in (10) as special (noise free) cases of the standard predictive equation (7) with $\mathbf{u}$ playing the role of (noise free) observations. Note that the (positive semidefinite) covariance matrices in (10) have the form $K - Q$ with the following interpretation: the prior covariance K minus a (non-negative definite) matrix Q quantifying how much information $\mathbf{u}$ provides about the variables in question ($\mathbf{f}$ or $\mathbf{f}_*$). We emphasize that all the sparse methods discussed in this chapter correspond simply to different approximations to the conditionals in (10), and throughout we use the exact likelihood and inducing prior

$$p(\mathbf{y}|\mathbf{f}) \;=\; \mathcal{N}(\mathbf{f},\; \sigma^2_{\text{noise}}\mathbf{I})\,, \quad \text{and} \quad p(\mathbf{u}) \;= \mathcal{N}(\mathbf{0},\; K_{\mathbf{u},\mathbf{u}})\,. \tag{11}$$

The sparse approximations we present in the following subsections are systematically sorted by decreasing crudeness of the additional approximations to the training and test conditionals. We will present the subset of data (SoD),

subset of regressors (SoR), deterministic training conditional (DTC), and the partially and fully independent training conditional (PITC, and FITC) approximations.

3.1 The Subset of Data (SoD) Approximation

Before we get started with the more sophisticated approximations, we mention as a baseline method the simplest possible sparse approximation (which doesn't fall inside our general scheme): use only a subset of the data (SoD). The computational complexity is reduced to $O(m^3)$, where $m < n$. We would not generally expect SoD to be a competitive method, since it would seem impossible (even with fairly redundant data and a good choice of the subset) to get a realistic picture of the uncertainties when only a part of the training data is even considered. We include it here mostly as a baseline against which to compare better sparse approximations.

We will illustrate the various sparse approximations on a toy dataset, as illustrated in Figure 4. There are $n = 100$ datapoints in 1-D, but for the sparse methods we have randomly selected $m = 10$ of these data points. The target function is given by $sin(x)/x$ with additive, independent, identically distributed Gaussian noise of amplitude 0.2. The training inputs lie on a uniform grid in $[-10, 10]$. For the predictions we have used a squared exponential (SE) covariance function (equation 3), with slightly too short a lengthscale $\lambda = 1.2$, chosen so as to emphasize the different behaviours, and with amplitude $v^2 = 0.7$. Figure 4 top left shows the predictive mean and two standard deviation error bars obtained with the full dataset. The 500 test inputs lie on a uniform grid in $[-14, 14]$.

In Figure 4 top right, we see how the SoD method produces wide predictive distributions when training on a randomly selected subset of 10 cases. A fair comparison to other methods would take into account that the computational complexity is independent of n as opposed to other more advanced methods. These extra computational resources could be spent in a number of ways, e.g. larger m, or an active (rather than random) selection of the m points. In this Chapter we will concentrate on understanding the theoretical foundations of the various approximations rather than investigating the necessary heuristics needed to turn the approximation schemes into practical algorithms.

3.2 The Subset of Regressors (SoR)

SoR models are finite linear-in-the-parameters models with a particular prior on the weights. For any input $\mathbf{x}_*$, the corresponding function value f_* is given by:

$$f_* = \sum_{i=1}^{m} k(\mathbf{x}_*, \mathbf{x}_\mathbf{u}^i) w_\mathbf{u}^i = K_{*,\mathbf{u}} \mathbf{w}_\mathbf{u} , \quad \text{with} \quad p(\mathbf{w}_\mathbf{u}) = \mathcal{N}(\mathbf{0}, K_{\mathbf{u},\mathbf{u}}^{-1}) , \quad (12)$$

where there is one weight associated with each inducing input in $\mathbf{X}_\mathbf{u}$. Note that the covariance matrix for the prior on the weights is the *inverse* of that on $\mathbf{u}$,

such that we recover the exact GP prior on $\mathbf{u}$, which is Gaussian with zero mean and covariance

$$\mathbf{u} \;=\; K_{\mathbf{u},\mathbf{u}}\,\mathbf{w}_{\mathbf{u}} \;\;\Rightarrow\;\; \langle \mathbf{u}\,\mathbf{u}^\top \rangle \;=\; K_{\mathbf{u},\mathbf{u}} \langle \mathbf{w}_{\mathbf{u}}\,\mathbf{w}_{\mathbf{u}}^\top \rangle K_{\mathbf{u},\mathbf{u}} \;=\; K_{\mathbf{u},\mathbf{u}}\,. \tag{13}$$

Using the effective prior on $\mathbf{u}$ and the fact that $\mathbf{w}_{\mathbf{u}} = K_{\mathbf{u},\mathbf{u}}^{-1}\,\mathbf{u}$ we can redefine the SoR model in an equivalent, more intuitive way:

$$\mathbf{f}_* \;=\; K_{*,\mathbf{u}}\,K_{\mathbf{u},\mathbf{u}}^{-1}\,\mathbf{u}\,, \quad \text{with} \quad \mathbf{u} \;\sim\; \mathcal{N}(\mathbf{0},\, K_{\mathbf{u},\mathbf{u}})\,. \tag{14}$$

The Subset of Regressors (SoR) algorithm was proposed in Wahba (1990, chapter 7), and in Poggio and Girosi (1990, eq. 25) via the regularization framework. It was adapted by Smola and Bartlett (2001) to propose a sparse greedy approximation to Gaussian process regression.

We are now ready to integrate the SoR model in our unifying framework. Given that there is a *deterministic* relation between any $\mathbf{f}_*$ and $\mathbf{u}$, the approximate conditional distributions in the integral in (9) are given by:

$$q_{\mathrm{SoR}}(\mathbf{f}|\mathbf{u}) \;=\; \mathcal{N}(K_{\mathbf{f},\mathbf{u}}\,K_{\mathbf{u},\mathbf{u}}^{-1}\,\mathbf{u},\, \mathbf{0})\,, \quad \text{and} \quad q_{\mathrm{SoR}}(\mathbf{f}_*|\mathbf{u}) \;=\; \mathcal{N}(K_{*,\mathbf{u}}\,K_{\mathbf{u},\mathbf{u}}^{-1}\,\mathbf{u},\, \mathbf{0})\,, \tag{15}$$

with zero conditional covariance, compare to (10). The effective prior implied by the SoR approximation is easily obtained from (9), giving

$$q_{\mathrm{SoR}}(\mathbf{f}, \mathbf{f}_*) \;=\; \mathcal{N}\Big(\mathbf{0},\, \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & Q_{*,*} \end{bmatrix}\Big)\,, \tag{16}$$

where we recall $Q_{\mathbf{a},\mathbf{b}} \triangleq K_{\mathbf{a},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} K_{\mathbf{u},\mathbf{b}}$. A more descriptive name for this method, would be the Deterministic Inducing Conditional (DIC) approximation. We see that this approximate prior is degenerate. There are only m degrees of freedom in the model, which implies that only m linearly independent functions can be drawn from the prior. The $m+1$-th one is a linear combination of the previous ones. For example, in a very low noise regime, the posterior could be severely constrained by only m training cases.

The degeneracy of the prior can cause unreasonable predictive distributions. For covariance functions that decay to zero for a pair of faraway inputs, it is immediate to see that $Q_{*,*}$ will go to zero when the test inputs $\mathbf{X}_*$ are far away from the inducing inputs $\mathbf{X}_{\mathbf{u}}$. As a result, $\mathbf{f}_*$ will have no prior variance under the approximate prior (16), and therefore the predictive variances will tend to zero as well far from the inducing inputs. This is unreasonable, because the area around the inducing inputs is where we are gathering the most information about the training data: we would like to be most uncertain about our predictions when far away from the inducing inputs. For covariance functions that do not decay to zero, the approximate prior over functions is still very restrictive, and given enough data only a very limited family of functions will be plausible under the posterior, leading to overconfident predictive variances. This is a general problem of finite linear models with small numbers of weights

(for more details see Rasmussen and Quiñonero-Candela, 2005). Figure 4, middle left panel, illustrates the unreasonable predictive uncertainties of the SoR approximation on a toy dataset.[5]

The predictive distribution is obtained by using the SoR approximate prior (16) instead of the true prior in (5). For each algorithm we give two forms of the predictive distribution, one which is easy to interpret, and the other which is economical to compute with:

$$q_{\mathrm{SoR}}(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}\big(Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \sigma^2_{\mathrm{noise}}\mathbf{I})^{-1}\mathbf{y}, \\ Q_{*,*} - Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \sigma^2_{\mathrm{noise}}\mathbf{I})^{-1}Q_{\mathbf{f},*}\big)\,, \tag{17a}$$

$$= \mathcal{N}\big(\sigma^{-2}K_{*,\mathbf{u}}\Sigma\, K_{\mathbf{u},\mathbf{f}}\,\mathbf{y},\; K_{*,\mathbf{u}}\Sigma K_{\mathbf{u},*}\big)\,, \tag{17b}$$

where we have defined $\Sigma = (\sigma^{-2}K_{\mathbf{u},\mathbf{f}}K_{\mathbf{f},\mathbf{u}} + K_{\mathbf{u},\mathbf{u}})^{-1}$. Equation (17a) is readily recognized as the regular prediction equation (7), except that the covariance K has everywhere been replaced by Q, which was already suggested by (16). This corresponds to replacing the covariance function k with $k_{\mathrm{SoR}}(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{u})K^{-1}_{\mathbf{u},\mathbf{u}}k(\mathbf{u}, \mathbf{x}_j)$. The new covariance function has rank (at most) m. Thus we have the following

Remark 3. *The SoR approximation is equivalent to exact inference in the degenerate Gaussian process with covariance function* $k_{\mathrm{SoR}}(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{u})K^{-1}_{\mathbf{u},\mathbf{u}}k(\mathbf{u}, \mathbf{x}_j)$.

The equivalent (17b) is computationally cheaper, and with (12) in mind, Σ is the covariance of the posterior on the weights $\mathbf{w}_{\mathbf{u}}$. Note that as opposed to the subset of data method, all training cases are taken into account. The computational complexity is $O(nm^2)$ initially, and $O(m)$ and $O(m^2)$ per test case for the predictive mean and variance respectively.

3.3 The Deterministic Training Conditional (DTC) Approximation

Taking up ideas contained in the work of Csató and Opper (2002), Seeger et al. (2003) recently proposed another sparse approximation to Gaussian process regression, which does not suffer from the nonsensical predictive uncertainties of the SoR approximation, but that interestingly leads to exactly the same predictive mean. Seeger et al. (2003), who called the method Projected Latent Variables (PLV), presented the method as relying on a *likelihood* approximation, based on the projection $\mathbf{f} = K_{\mathbf{f},\mathbf{u}}\,K^{-1}_{\mathbf{u},\mathbf{u}}\,\mathbf{u}$:

$$p(\mathbf{y}|\mathbf{f}) \;\simeq\; q(\mathbf{y}|\mathbf{u}) \;=\; \mathcal{N}(K_{\mathbf{f},\mathbf{u}}\,K^{-1}_{\mathbf{u},\mathbf{u}}\,\mathbf{u},\; \sigma^2_{\mathrm{noise}}\mathbf{I})\,. \tag{18}$$

The method has also been called the Projected Process Approximation (PPA) by Rasmussen and Williams (2006, Chapter 8). One way of obtaining an equivalent

[5] Wary of this fact, Smola and Bartlett (2001) propose using the predictive variances of the SoD method, or a more accurate but computationally costly alternative (more details are given in Quiñonero-Candela, 2004, Chapter 3).

model is to retain the usual likelihood, but to impose a deterministic training conditional and the exact test conditional from (10b)

$$q_{\mathrm{DTC}}(\mathbf{f}|\mathbf{u}) \;=\; \mathcal{N}(K_{\mathbf{f},\mathbf{u}}\,K_{\mathbf{u},\mathbf{u}}^{-1}\,\mathbf{u},\mathbf{0}), \quad \text{and} \quad q_{\mathrm{DTC}}(\mathbf{f}_*|\mathbf{u}) \;=\; p(\mathbf{f}_*|\mathbf{u})\,. \quad (19)$$

This reformulation has the advantage of allowing us to stick to our view of exact inference (with exact likelihood) with approximate priors. Indeed, under this model the conditional distribution of $\mathbf{f}$ given $\mathbf{u}$ is identical to that of the SoR, given in the left of (15). In this framework a systematic name for this approximation is the Deterministic Training Conditional (DTC).

The fundamental difference with SoR is that DTC uses the exact test conditional (10b) instead of the deterministic relation between $\mathbf{f}_*$ and $\mathbf{u}$ of SoR. The joint prior implied by DTC is given by:

$$q_{\mathrm{DTC}}(\mathbf{f},\mathbf{f}_*) \;=\; \mathcal{N}\Big(\mathbf{0},\, \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}\Big)\,, \quad (20)$$

which is surprisingly similar to the effective prior implied by the SoR approximation (16). The difference is that under the DTC approximation $\mathbf{f}_*$ has a prior variance of its own, given by $K_{*,*}$. This prior variance reverses the behaviour of the predictive uncertainties, and turns them into sensible ones, see Figure 4, middle right, for an illustration.

The predictive distribution is now given by:

$$\begin{aligned} q_{\mathrm{DTC}}(\mathbf{f}_*|\mathbf{y}) \;&=\; \mathcal{N}\big(Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}}+\sigma^2_{\mathrm{noise}}\mathbf{I})^{-1}\mathbf{y}, \\ &\qquad\qquad K_{*,*}-Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}}+\sigma^2_{\mathrm{noise}}\mathbf{I})^{-1}Q_{\mathbf{f},*}\big)\,, \quad (21a) \\ &=\; \mathcal{N}\big(\sigma^{-2}K_{*,\mathbf{u}}\Sigma\,K_{\mathbf{u},\mathbf{f}}\,\mathbf{y},\; K_{*,*}-Q_{*,*}+K_{*,\mathbf{u}}\Sigma K_{*,\mathbf{u}}^{\top}\big)\,, \quad (21b) \end{aligned}$$

where again we have defined $\Sigma = (\sigma^{-2}K_{\mathbf{u},\mathbf{f}}K_{\mathbf{f},\mathbf{u}} + K_{\mathbf{u},\mathbf{u}})^{-1}$ as in (17). The predictive mean for the DTC is identical to that of the SoR approximation (17), but the predictive variance replaces the $Q_{*,*}$ from SoR with $K_{*,*}$ (which is larger, since $K_{*,*} - Q_{*,*}$ is positive semidefinite). This added term is the predictive variance of the posterior of f_* conditioned on $\mathbf{u}$. It grows to the prior variance $K_{*,*}$ as $\mathbf{x}_*$ moves far from the inducing inputs in $\mathbf{X}_\mathbf{u}$.

Remark 4. *The only difference between the predictive distribution of DTC and SoR is the variance. The predictive variance of DTC is never smaller than that of SoR.*

Note, that since the covariances for training cases and test cases are computed differently, see (20), it follows that

Remark 5. *The DTC approximation does not correspond exactly to a Gaussian process,*

as the covariance between latent values depends on whether they are considered training or test cases, violating consistency, see Definition 1. The computational complexity has the same order as for SoR.

3.4 Partially Independent Training Conditional Approximations

The sparse approximations we have seen so far, SoR and DTC, both impose a deterministic relation between the training and inducing latent variables, resulting in inducing training conditionals where the covariance matrix has been set to zero, (15) and (19). A less crude approximation to the training conditional is to preserve a block-diagonal of the true covariance matrix, given by (10a), and set the remaining entries to zero. The corresponding graphical model is shown in Figure 2; notice that the variables in $\mathbf{f}$ are divided into k groups. Each group corresponds to a block in the block-diagonal matrix. This structure is equivalent to assuming conditional independence only for part of the training function values (those with covariance set to zero). We call this the partially independent training conditional (PITC) approximation:

$$\begin{aligned} q_{\text{PITC}}(\mathbf{f}|\mathbf{u}) &= \mathcal{N}\big(K_{\mathbf{f},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u},\ \text{blockdiag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}]\big)\ , \\ \text{and} \quad q_{\text{PITC}}(f_*|\mathbf{u}) &= p(f_*|\mathbf{u})\ . \end{aligned} \tag{22}$$

where blockdiag$[A]$ is a block-diagonal matrix (where the blocking structure is not explicitly stated). For now, we consider one single test input given by the exact test conditional (10b). As we discuss later in this section, the joint test conditional can also be approximated in various ways. The effective prior implied by the PITC approximation is given by

$$q_{\text{PITC}}(\mathbf{f}, f_*) = \mathcal{N}\Big(\mathbf{0},\ \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \text{blockdiag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}\Big)\ , \tag{23}$$

Note, that the sole difference between the DTC and PITC is that in the top left corner of the implied prior covariance, PITC replaces the approximate covariances of DTC by the exact ones on the block-diagonal. The predictive distribution is

$$q_{\text{FITC}}(f_*|\mathbf{y}) = \mathcal{N}\big(Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \Lambda)^{-1}\mathbf{y},\ K_{*,*} - Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \Lambda)^{-1}Q_{\mathbf{f},*}\big) \tag{24a}$$

$$= \mathcal{N}\big(K_{*,\mathbf{u}}\Sigma K_{\mathbf{u},\mathbf{f}}\Lambda^{-1}\mathbf{y},\ K_{*,*} - Q_{*,*} + K_{*,\mathbf{u}}\Sigma K_{\mathbf{u},*}\big)\ , \tag{24b}$$

where we have defined $\Sigma = (K_{\mathbf{u},\mathbf{u}} + K_{\mathbf{u},\mathbf{f}}\Lambda^{-1}K_{\mathbf{f},\mathbf{u}})^{-1}$ and $\Lambda = \text{blockdiag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}} + \sigma^2_{\text{noise}}\mathbf{I}]$. An identical expression was obtained by Schwaighofer and Tresp (2003, Sect. 3), developed from the original Bayesian Committee Machine (BCM) of Tresp (2000). The BCM was originally proposed as a transductive learner (i.e. where the *test* inputs have to be known before training), and the inducing inputs $\mathbf{X}_\mathbf{u}$ were chosen to be the test inputs.

However, it is important to realize that the BCM proposes two orthogonal ideas: first, the block-diagonal structure of the partially independent training conditional, and second setting the inducing inputs to be the test inputs. These two ideas can be used independently and in Section 5 we propose using the first without the second.

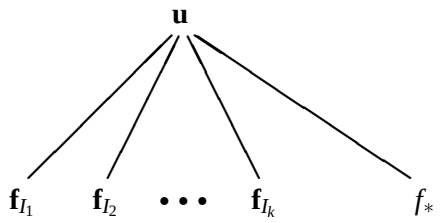

Figure 2: Graphical representation of the PITC approximation. The set of latent function values $\mathbf{f}_{I_i}$ indexed by the the set of indices I_i is fully connected. The PITC is more general than the FITC, (see graph in Fig. 3) in that conditional independence is between k *groups* of training latent function values. This corresponds to the block-diagonal approximation to the true training conditional given in (22).

The computational complexity of the PITC approximation depends on the blocking structure imposed in (22). A reasonable choice, also recommended by Tresp (2000) may be to choose $k = n/m$ blocks, each of size $m \times m$. The computational complexity is thus $O(nm^2)$. Since in the PITC model, the covariance is computed differently for training and test cases we have

Remark 6. *The PITC approximation does not correspond exactly to a Gaussian process.*

This is because computing covariances requires knowing whether points are from the training- or test-set, (23). To obtain a Gaussian process from the PITC, one would need to extend the partial conditional independence assumption to the joint conditional $p(\mathbf{f}, \mathbf{f}_*|\mathbf{u})$, which would require abandoning our primal assumption that the training and the test function values are conditionally independent, given the inducing variables.

Fully Independent Training Conditional Recently Snelson and Ghahramani (2006) proposed another likelihood approximation to speed up Gaussian process regression, which they called Sparse Pseudo-input Gaussian processes (SPGP). While the DTC is based on the likelihood approximation given by (18), the SPGP proposes a more sophisticated likelihood approximation with a richer covariance

$$p(\mathbf{y}|\mathbf{f}) \;\simeq\; q(\mathbf{y}|\mathbf{u}) \;=\; \mathcal{N}(K_{\mathbf{f},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u},\; \mathrm{diag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}] + \sigma^2_{\text{noise}}\mathbf{I}) \;, \qquad (25)$$

where $\mathrm{diag}[A]$ is a diagonal matrix whose elements match the diagonal of A. As was pointed out by Csató (2005), the SPGP is an extreme case of the PITC approximation, where the training conditional is taken to be fully independent. We call it the Fully Independent Training Conditional (FITC) approximation. The corresponding graphical model is given in Figure 3. The effective prior

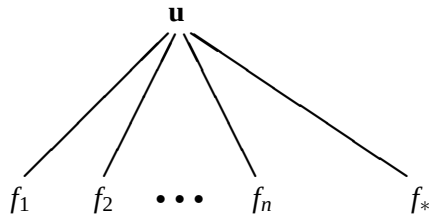

Figure 3: Graphical model for the FITC approximation. Compared to those in Figure 1, all edges between latent function values have been removed: the latent function values are conditionally fully independent given the inducing variables $\mathbf{u}$. Although strictly speaking the SoR and DTC approximations could also be represented by this graph, note that both further assume a deterministic relation between $\mathbf{f}$ and $\mathbf{u}$. The FITC is an extreme case of PITC, with $k = n$ unitary groups (blocks), see Figure 2

implied by the FITC is given by

$$q_{\mathrm{FITC}}(\mathbf{f}, f_*) \;=\; \mathcal{N}\Big(\mathbf{0},\; \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \mathrm{diag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}\Big)\,, \tag{26}$$

and we see that in contrast to PITC, FITC only replaces the approximate covariances of DTC by the exact ones on the diagonal, i.e. the approximate prior variances are replaced by the true prior variances.

The predictive distribution of the FITC is identical to that of PITC (24), except for the alternative definition of $\Lambda = \mathrm{diag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}} + \sigma^2_{\mathrm{noise}}\mathbf{I}]$. The computational complexity is identical to that of SoR and DTC.

The question poses itself again for FITC, of how to treat the test conditional. For predictions at a single test input, this is obvious. For joint predictions, there are two options, either 1) use the exact full test conditional from (10b), or 2) extend the additional factorizing assumption to the test conditional. Although Snelson and Ghahramani (2006) don't explicitly discuss joint predictions, it would seem that they probably intend the second option. Whereas the additional independence assumption for the test cases is not really necessary for computational reasons, it does affect the nature of the approximation. Under option 1) the training and test covariance are computed differently, and thus this does not correspond to our strict definition of a GP model, but

Remark 7. *Iff the assumption of full independence is extended to the test conditional, the FITC approximation is equivalent to exact inference in a non-degenerate Gaussian process with covariance function* $k_{\mathrm{FIC}}(\mathbf{x}_i, \mathbf{x}_j) = k_{\mathrm{SoR}}(\mathbf{x}_i, \mathbf{x}_j) + \delta_{i,j}[k(\mathbf{x}_i, \mathbf{x}_j) - k_{\mathrm{SoR}}(\mathbf{x}_i, \mathbf{x}_j)]$,

where $\delta_{i,j}$ is Kronecker's delta. A logical name for the method where the conditionals (training and test) are always forced to be fully independent would

be the Fully Independent Conditional (FIC) approximation. The effective prior implied by FIC is:

$$q_{\mathrm{FIC}}(\mathbf{f}, \mathbf{f}_*) \;=\; \mathcal{N}\Big(\mathbf{0},\; \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \mathrm{diag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & Q_{*,*} - \mathrm{diag}[Q_{*,*} - K_{*,*}] \end{bmatrix}\Big) \,. \tag{27}$$

In Figure 4, bottom, we show the behaviour of the predictive distribution of FITC, and PITC (with 10 uniform blocks).

Summary of sparse approximations In the table below we give a summary of the way approximations are built. All these methods have been detailed in the previous subsections. The initial cost and that of the mean and variance per test case are respectively n^3, n and n^2 for the exact GP, and nm^2, m and m^2 for all other listed methods. The "GP?" column indicates whether the approximation is equivalent to a GP. For FITC see Remark 7. Recall that we have defined $Q_{\mathbf{a},\mathbf{b}} \triangleq K_{\mathbf{a},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} K_{\mathbf{u},\mathbf{b}}$.

Method	$q(\mathbf{f}_*\|\mathbf{u})$	$q(\mathbf{f}\|\mathbf{u})$	joint prior covariance	GP?
GP	exact	exact	$\begin{bmatrix} K_{\mathbf{f},\mathbf{f}} & K_{\mathbf{f},*} \\ K_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}$	$\surd$
SoR	determ.	determ.	$\begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & Q_{*,*} \end{bmatrix}$	$\surd$
DTC	exact	determ.	$\begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}$	
FITC	(exact)	fully indep.	$\begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \mathrm{diag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}$	$(\surd)$
PITC	exact	partially indep.	$\begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \mathrm{blokdiag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}$	

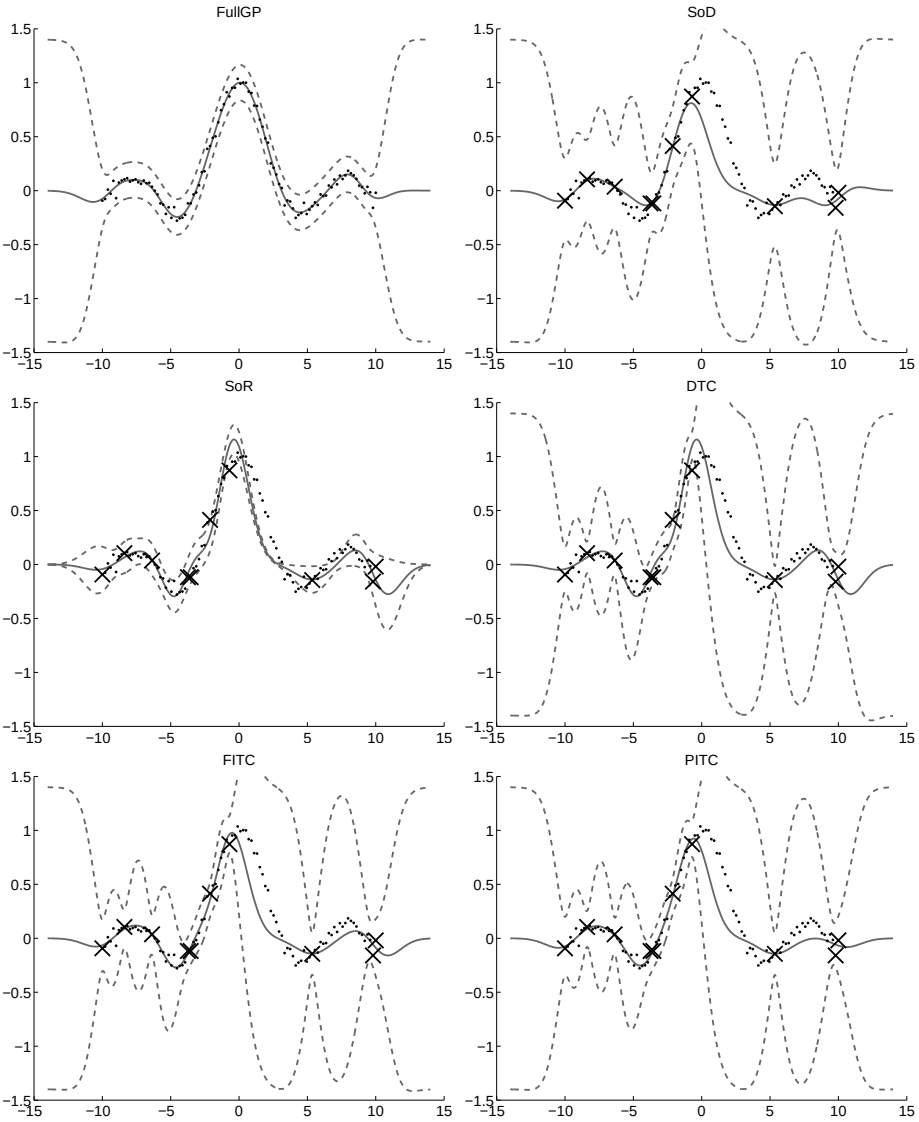

Figure 4: Toy data. All methods use the squared exponential covariance function, and a slightly too short lengthscale, chosen on purpose to emphasize the different behaviour of the predictive uncertainties. The dots are the training points, the crosses are the targets corresponding to the inducing inputs, randomly selected from the training set. The solid line is the mean of the predictive distribution, and the dotted lines show the 95% confidence interval of the predictions. We include a full GP (top left) for reference.

4 Fast Matrix Vector Multiplication (MVM) Approximations

One straightforward method to speed up GP regression is to note that linear system $(K_{\mathbf{f},\mathbf{f}} + \sigma^2_{\text{noise}}\,\mathbf{I})\,\boldsymbol{\alpha} = \mathbf{y}$ can be solved by an iterative method, for example conjugate gradients (CG). (See Golub and Van Loan (1989, sec. 10.2) for further details on the CG method.) Each iteration of the CG method requires a matrix-vector multiplication (MVM) which takes $O(n^2)$ time. Conjugate gradients gives the exact solution (ignoring round-off errors) if run for n iterations, but it will give an approximate solution if terminated earlier, say after k iterations, with time complexity $O(kn^2)$. The CG method has been suggested by Wahba et al. (1995) (in the context of numerical weather prediction) and by Gibbs and MacKay (1997) (in the context of general GP regression).

However, the scaling of the CG method (at least $O(n^2)$) is too slow to really be useful for large problems. Recently a number of researchers have noted that if the MVM step can be approximated efficiently then the CG method can be speeded up. Each CG iteration involves an MVM of the form $\mathbf{a} = (K_{\mathbf{f},\mathbf{f}} + \sigma^2_{\text{noise}}\mathbf{I})\mathbf{v}$, which requires n inner products $a_i = \sum_{j=1}^{n} k(\mathbf{x}_i, \mathbf{x}_j)v_j$. Each a_i is a weighted sum of kernels between source points $\{\mathbf{x}_j\}$ and target point $\mathbf{x}_i$. Such sums can be approximated using multi-resolution space-partitioning data structures. For example Shen et al. (2006) suggest a simple approximation using a k-d tree, and while Gray (2004) and Freitas et al. (2006) make use of dual trees. The Improved Fast Gauss Transform (IFGT) (Yang et al., 2005) is another method for approximating the weighted sums. It is specific to the Gaussian kernel and uses a more sophisticated approximation based on Taylor expansions. Generally experimental results show that these fast MVM methods are most effective when the input space is low dimensional.

Notice that these methods for accelerating the weighted sum of kernel functions can also be used at test time, and may be particularly helpful when n is large.

5 Selecting the Inducing Variables

We have until now assumed that the inducing inputs $\mathbf{X}_\mathbf{u}$ were given. Traditionally, sparse models have very often been built upon a carefully chosen subset of the training inputs. This concept is probably best exemplified in the popular support vector machine (SVM) (Cortes and Vapnik, 1995). The recent work by Keerthi et al. (2006) seeks to further sparsify the SVM. In sparse Gaussian processes, it has also been suggested to select the inducing inputs $\mathbf{X}_\mathbf{u}$ from among the training inputs. Since this involves a prohibitive combinatorial optimization, greedy optimization approaches have been suggested using various selection criteria like online learning (Csató and Opper, 2002), greedy posterior maximization (Smola and Bartlett, 2001), maximum information gain (Lawrence et al., 2003; Seeger et al., 2003), matching pursuit (Keerthi and Chu, 2006), and others. As discussed in Section 3.4, selecting the inducing inputs

from among the test inputs has also been considered in transductive settings by Tresp (2000). Recently, Snelson and Ghahramani (2006) have proposed to relax the constraint that the inducing variables must be a subset of training/test cases, turning the discrete selection problem into one of continuous optimization. One may hope that finding a good solution is easier in the continuous than the discrete case, although finding the global optimum is intractable in both cases. And it is possible that the less restrictive choice can lead to better performance in very sparse models.

Which criterion should be used to set the inducing inputs? Departing from a fully Bayesian treatment which would involve defining priors on $\mathbf{X_u}$, one could maximize the marginal likelihood (also called the evidence) with respect to $\mathbf{X_u}$, an approach also followed by Snelson and Ghahramani (2006). Each of the approximate methods proposed involves a different effective prior, and hence its own particular effective marginal likelihood conditioned on the inducing inputs

$$q(\mathbf{y}|\mathbf{X_u}) = \iint p(\mathbf{y}|\mathbf{f})\, q(\mathbf{f}|\mathbf{u})\, p(\mathbf{u}|\mathbf{X_u}) \mathrm{d}\mathbf{u}\, \mathrm{d}\mathbf{f} = \int p(\mathbf{y}|\mathbf{f})\, q(\mathbf{f}|\mathbf{X_u}) \mathrm{d}\mathbf{f}\ , \qquad (28)$$

which of course is independent of the test conditional. We have in the above equation explicitly conditioned on the inducing inputs $\mathbf{X_u}$. Using Gaussian identities, the effective marginal likelihood is very easily obtained by adding a ridge $\sigma^2_{\text{noise}}\mathbf{I}$ (from the likelihood) to the covariance of effective prior on $\mathbf{f}$. Using the appropriate definitions of Λ, the log marginal likelihood becomes

$$\log q(\mathbf{y}|\mathbf{X_u}) \ = \ -\tfrac{1}{2}\log|Q_{\mathbf{f},\mathbf{f}} + \Lambda| - \tfrac{1}{2}\mathbf{y}^\top (Q_{\mathbf{f},\mathbf{f}} + \Lambda)^{-1}\mathbf{y} - \tfrac{n}{2}\log(2\pi)\ , \quad (29)$$

where $\Lambda_{\text{SoR}} = \Lambda_{\text{DTC}} = \sigma^2_{\text{noise}}\mathbf{I}$, $\Lambda_{\text{FITC}} = \text{diag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}] + \sigma^2_{\text{noise}}\mathbf{I}$, and $\Lambda_{\text{PITC}} = \text{blockdiag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}] + \sigma^2_{\text{noise}}\mathbf{I}$. The computational cost of the marginal likelihood is $O(nm^2)$ for all methods, that of its gradient with respect to one element of $\mathbf{X_u}$ is $O(nm)$. This of course implies that the complexity of computing the gradient wrt. to the whole of $\mathbf{X_u}$ is $O(dnm^2)$, where d is the dimension of the input space.

It has been proposed to maximize the effective posterior instead of the effective marginal likelihood (Smola and Bartlett, 2001). However, this is potentially dangerous and can lead to overfitting. Maximizing the whole evidence instead is sound and comes at an identical computational cost (for a deeper analysis, see Quiñonero-Candela, 2004, Sect. 3.3.5 and Fig. 3.2).

The marginal likelihood has traditionally been used to learn the hyperparameters of GPs, in the non fully Bayesian treatment (see for example Williams and Rasmussen, 1996). For the sparse approximations presented here, once you are learning $\mathbf{X_u}$ it is straightforward to allow for learning hyperparameters (of the covariance function) during the same optimization. For methods that select $\mathbf{X_u}$ from the training data, one typically interleaves optimization of the hyperparameters with with selection of $\mathbf{X_u}$, as proposed for example by Seeger et al. (2003).

Augmentation Since in the previous sections, we haven't assumed anything about $\mathbf{u}$, for each test input $\mathbf{x}_*$ in turn, we can simply *augment* the set of inducing variables by f_*, so that we have one additional inducing variable equal to the current test latent. Let us first investigate the consequences for the test conditional from (10b). Note that the interpretation of the covariance matrix $K_{*,*} - Q_{*,*}$ was "the prior covariance minus the information which $\mathbf{u}$ provides about f_*". It is clear that the augmented $\mathbf{u}$ (with f_*) provides all possible information about f_*, and consequently $Q_{*,*} = K_{*,*}$. An equivalent view on augmentation is that the assumption of conditional independence between f_* and $\mathbf{f}$ is dropped. This is seen trivially, by adding edges between f_* and the f_i. Because f_* enjoys a full, original prior variance under the test conditional, augmentation helps reverse the misbehaviour of the predictive variances of degenerate GP priors, see (Quiñonero-Candela and Rasmussen, 2005, Section 8.1) for the details. Augmented SoR and augmented DTC are identical models (see Quiñonero-Candela and Rasmussen, 2005, Remark 12).

Augmentation was originally proposed by Rasmussen (2002), and applied in detail to the SoR with RBF covariance by Quiñonero-Candela (2004). Later, Rasmussen and Quiñonero-Candela (2005) proposed to use augmentation to "heal" the relevance vector machine (RVM) (Tipping, 2000), which is also equivalent to a degenerate GP with nonsensical predictive variances that shrink to zero far away from the training inputs (Tipping, 2001, Appendix D). Although augmentation was initially proposed for a narrow set of circumstances, it is easily applied to any of the approximations discussed. Of course, augmentation doesn't make any sense for an exact, non-degenerate Gaussian process model (a GP with a covariance function that has a feature-space which is infinite dimensional, i.e. with basis functions *everywhere*).

Prediction with an augmented sparse model comes at a higher computational cost, since now f_* interacts directly with all of $\mathbf{f}$, and not just with $\mathbf{u}$. For each new test point $O(nm)$ operations are required, as opposed to $O(m)$ for the mean, and $O(m^2)$ for the predictive distribution of all the non-augmented methods we have discussed. Whether augmentation is of practical interest (i.e. increases performance at a fixed computational budget) is unclear, since the extra computational effort needed for augmentation could be invested by the non-augmented methods, for example to use more inducing variables.

6 Approximate Evidence and Hyperparameter Learning

Hyperparameter learning is an issue that is sometimes completely ignored in the literature on approximations to GPs. However, learning good values of the hyperparameters is crucial, and it might come at a high computational cost that cannot be ignored. For clarity, let us distinguish between three phases in GP modelling:

hyperparameter learning: the hyperparameters are learned, by for example

maximizing the log marginal likelihood.

pre-computation: all possible pre-computations not involving test inputs are performed (such as inverting the covariance matrix of the training function values, and multiplying it by the vector of targets).

testing: only the computations involving the test inputs are made, which could not have been made previously.

If approximations to GP regression are to provide a computational speedup, then one must, in principle, take into account *all three phases* of GP modelling.

Let us for a moment group hyperparameter learning and pre-computations into a wider *training* phase. In practice, there will be situations where one of the two times, training or testing, is more important than the other. Let us distinguish between two extreme scenarios. One the one hand, there are applications for which training time is unimportant, the crucial point being that the time required to make predictions, once the system has been trained, is really small. An example could be applications embedded in a portable device, with limited CPU power. At the other extreme are applications where the system needs to be re-trained often and quickly, and where prediction time matters much less than the quality of the predictions. An example could be a case where we know already that we will only need to make very few predictions, hence training time automatically becomes predominant. A particular application might lie somewhere between these two extremes.

A common choice of objective function for setting the hyperparameters is the *marginal likelihood* $p(\mathbf{y}|\mathbf{X})$. For GP regression, this can be computed exactly (by integrating out $\mathbf{f}$ analytically) to obtain

$$\log p(\mathbf{y}|\mathbf{X}) \;=\; -\tfrac{1}{2}\log|K_{\mathbf{f},\mathbf{f}}+\sigma^2_{\text{noise}}\mathbf{I}|-\tfrac{1}{2}\mathbf{y}^\top(K_{\mathbf{f},\mathbf{f}}+\sigma^2_{\text{noise}}\mathbf{I})^{-1}\mathbf{y}-\tfrac{n}{2}\log(2\pi)\;. \quad (30)$$

Also, the derivative of the marginal likelihood with respect to a hyperparameter θ_j is given by

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|\mathbf{X}) \;=\; -\tfrac{1}{2}\mathbf{y}^\top R^{-1}\frac{\partial R}{\partial\theta_j}R^{-1}\mathbf{y} - \tfrac{1}{2}\text{tr}\left(R^{-1}\frac{\partial R}{\partial\theta_j}\right), \quad (31)$$

where $R = (K_{\mathbf{f},\mathbf{f}} + \sigma^2_{\text{noise}}\mathbf{I})$, and $\text{tr}(A)$ denotes the trace of matrix A. These derivatives can then be fed to a gradient-based optimization routine. The marginal likelihood is not the only criterion that can be used for hyperparameter optimization. For example one can use a cross-validation objective; see Rasmussen and Williams (2006, chapter 5) for further details.

The approximate log marginal likelihood for the SoR, DTC, FITC and PITC approximations is given in equation (29), taking time $O(nm^2)$. For the SoD approximation one simply ignores all training datapoints not in the inducing set, giving

$$\begin{aligned}\log q_{\text{SoD}}(\mathbf{y}|\mathbf{X}_\mathbf{u}) \;=\; &-\tfrac{1}{2}\log|K_{\mathbf{u},\mathbf{u}}+\sigma^2_{\text{noise}}\mathbf{I}| - \tfrac{1}{2}\mathbf{y}_\mathbf{u}^\top(K_{\mathbf{u},\mathbf{u}}+\sigma^2_{\text{noise}}\mathbf{I})^{-1}\mathbf{y}_\mathbf{u} \\ &-\tfrac{m}{2}\log(2\pi)\;,\end{aligned} \quad (32)$$

where the inducing inputs $\mathbf{X_u}$ are a subset of the training inputs, and $\mathbf{y_u}$ are the corresponding training targets. The computational cost here is $O(m^3)$, instead of the $O(n^3)$ operations needed to compute the evidence of the full GP model. For all of these approximations one can also calculate derivatives with respect to the hyperparameters analytically, for use in a gradient-based optimizer.

In general we believe that for a given sparse approximation it makes most sense to both optimize the hyperparameters and make predictions under the same approximation (Quiñonero-Candela, 2004, Chapter 3). The hyperparameters that are optimal for the full GP model, if we were able to obtain them, may also well be very different from those optimal for a specific sparse approximation. For example, for the squared exponential covariance function, see (3), the lengthscale optimal for the full GP might be too short for a very sparse approximation. Certain sparse approximations lead to marginal likelihoods that are better behaved than others. Snelson and Ghahramani (2006) report that the marginal likelihood of FITC suffers much less from local maxima than the marginal likelihood common to SoR and DTC.

Gibbs and MacKay (1997) discussed how the marginal likelihood and its derivatives can be approximated using CG iterative methods, building on ideas in Skilling (1993). It should be possible to speed up such computations using the fast MVM methods outlined in section 4.

7 Classification

Compared with the regression case, the approximation methods for GP classification need to deal with an additional difficulty: the likelihood is non-Gaussian, which implies that neither the predictive distribution (5), nor the marginal likelihood can be computed analytically. With the approximate prior at hand, the most common approach is to further approximate the resulting posterior, $p(\mathbf{f}|\mathbf{y})$, by a Gaussian. It can be shown that if $p(\mathbf{y}|\mathbf{f})$ is log-concave then the posterior will be unimodal; this is the case for the common logistic and probit response functions. (For a review of likelihood functions for GP classification, see Rasmussen and Williams (2006, Chapter 3).) An evaluation of different, common ways of determining the approximating Gaussian has been made by Kuss and Rasmussen (2005).

For the subset of regressors (SoR) method we have $f_* = \sum_{i=1}^{m} k(\mathbf{x}_*, \mathbf{x}_\mathbf{u}^i) w_\mathbf{u}^i$ with $\mathbf{w_u} \sim \mathcal{N}(\mathbf{0},\ K_{\mathbf{u},\mathbf{u}}^{-1})$ from (12). For log-concave likelihoods, the optimization problem to find the maximum a posteriori (MAP) value of $\mathbf{w_u}$ is convex. Using the MAP value of the weights together with the Hessian at this point, we obtain an approximate Gaussian predictive distribution of f_*, which can be fed through the sigmoid function to yield probabilistic predictions. The question of how to choose the inducing inputs still arises; Lin et al. (2000) select these using a clustering method, while Zhu and Hastie (2002) propose a forward selection strategy.

The subset of data (SoD) method for GP classification was proposed by Lawrence et al. (2003), using an expectation-propagation (EP) approximation

to the posterior (Minka, 2001), and an information gain criterion for greedily selecting the inducing inputs from the training set.

The deterministic training conditional (DTC) approximation has also been used for classification. Csató and Opper (2002) present and "online" method, where the examples are processed sequentially, while Csató et al. (2003) give an EP type algorithm where multiple sweeps across the data are permitted.

The partially independent training conditional (PITC) approximation has also been applied to GP classification. Tresp (2000) generalized the use of the Bayesian committee machine (BCM) to non-Gaussian likelihoods, by applying a Laplace approximation to each of the partially independent blocks.

8 Conclusions

In this chapter we have reviewed a number of methods for dealing with the $O(n^3)$ scaling of a naïve implementation of Gaussian process regression methods. These can be divided into two classes, those based on sparse methods, and those based on fast MVM methods. We have also discussed the selection of inducing variables, hyperparameter learning, and approximation methods for the marginal likelihood.

Probably the question that is most on the mind of the practitioner is "what method should I use on my problem?" Unfortunately this is not an easy question to answer, as it will depend on various aspects of the problem such as the complexity of the underlying target function, the dimensionality of the inputs, the amount of noise on the targets, etc. There has been some empirical work on the comparison of approximate methods, for example in Schwaighofer and Tresp (2003) and Rasmussen and Williams (2006, chapter 8) but more needs to be done. Empirical evaluation is not easy, as there exist very many criteria of comparison, under which different methods might perform best. There is a large choice of the measures of performance: mean squared error, negative log predictive probability, etc., and an abundance of possible measures of speed for the approximations: hyperparameter learning time, pre-computation time, testing time, number of inducing variables, number of CG iterations, etc. One possible approach for empirical evaluation of computationally efficient approximations to GPs is to fix a "computational budget" of available time, and see which approximation achieves the best performance (under some loss) within that time. Empirical comparisons should always include approximations that at first sight might seem to simple or naïve, such as subset of data (SoD), but that might end up performing as well as more elaborate methods.

Acknowledgements

The authors thank Ed Snelson and Iain Murray for recent, very fruitful discussions. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This

publication only reflects the authors' views. CER was supported by the German Research Council (DFG) through grant RA 1030/1.

References

Corinna Cortes and Vladimir Vapnik. Support-vector network. *Machine Learning*, 20(3):273–297, 1995.

Lehel Csató, 2005. Personal communication at the 2005 Sheffield Gaussian Process Round Table, organized by Neil D. Lawrence.

Lehel Csató and Manfred Opper. Sparse online Gaussian processes. *Neural Computation*, 14(3):641–669, 2002.

Lehel Csató, Manfred Opper, and Ole Winther. TAP Gibbs Free Energy, Belief Propagation and Sparsity. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Neural Information Processing Systems 14*, pages 657–663, Cambridge, MA, 2003. MIT Press.

Nando De Freitas, Yang Wang, Maryam Mahdaviani, and Dustin Lang. Fast Krylov methods for N-body learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

Mark N. Gibbs and David J. C. MacKay. Efficient Implementation of Gaussian Processes. Unpublished manuscript. Cavendish Laboratory, Cambridge, UK. `http://www.inference.phy.cam.ac.uk/ mackay/BayesGP.html`, 1997.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989. Second edition.

Alexander Gray. Fast kernel matrix-vector multiplication with application to Gaussian process learning. Technical Report CMU-CS-04-110, School of Computer Science, Carnegie Mellon University, 2004.

S. Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.

S. Sathiya Keerthi and Wei Chu. A matching pursuit approach to sparse GP regression. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

Malte Kuss and Carl Edward Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, pages 1679–1704, 2005.

Neil Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Neural Information Processing Systems 15*, pages 609–616, Cambridge, MA, 2003. MIT Press.

Xiwi Lin, Grace Wahba, Dong Xiang, Fangyu Gao, Ronald Klein, and Barbara Klein. Smoothing Spline ANOVA Models for Large Data Sets With Bernoulli Observations and the Randomized GACV. *Annals of Statistics*, 28:1570–1600, 2000.

Thomas P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

Tomaso Poggio and Frederico Girosi. Networks for Approximation and Learning. *Proceedings of IEEE*, 78:1481–1497, 1990.

Joaquin Quiñonero-Candela. *Learning with Uncertainty – Gaussian Processes and Relevance Vector Machines*. PhD thesis, Technical University of Denmark, Lyngby, Denmark, 2004.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Carl Edward Rasmussen. Reduced rank Gaussian process learning. Technical report, Gatsby Computational Neuroscience Unit, UCL, 2002.

Carl Edward Rasmussen and Joaquin Quiñonero-Candela. Healing the relevance vector machine by augmentation. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning*, pages 689–696, 2005.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Massachusetts, 2006.

Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate Gaussian process regression. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.

Matthias Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, 2003.

Matthias Seeger, Christopher K. I. Williams, and Neil Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In Christopher M. Bishop and Brendan J. Frey, editors, *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2003.

Yirong Shen, Andrew Ng, and Matthias Seeger. Fast Gaussian process regression using KD-trees. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

John Skilling. Bayesian numerical analysis. In W. T. Grandy, Jr. and P. Milonni, editors, *Physics and Probability*. Cambridge University Press, 1993.

Alexander J. Smola and Peter L. Bartlett. Sparse greedy Gaussian process regression. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625, Cambridge, MA, 2001. MIT Press.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

Michael E. Tipping. The relevance vector machine. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658, Cambridge, MA, 2000. MIT Press.

Michael E. Tipping. Sparse Bayesian learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

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

Grace Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

Grace Wahba, Donald R. Johnson, Feng Gao, and Jianjian Gong. Adaptive Tuning of Numerical Weather Prediction Models: Randomized GCV in Three-and Four-Dimensional Data Assimilation. *Monthly Weather Review*, 123:3358–3369, 1995.

Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514–520, Cambridge, MA, 1996. MIT Press.

Changjiang Yang, Ramani Duraiswami, and Larry Davis. Efficient kernel machines using the improved fast Gauss transform. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1561–1568, Cambridge, MA, 2005. MIT Press.

Ji Zhu and Trevor J. Hastie. Kernel Logistic Regression and the Import Vector Machine. In T. G. Diettrich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1081–1088. MIT Press, 2002.