Chapter 1

Markov chain Monte Carlo algorithms for Gaussian processes

Michalis K. Titsias and Magnus Rattray and Neil D. Lawrence[1]

1.1 Introduction

Gaussian processes (GPs) have a long history in statistical physics and mathematical probability. Two of the most well-studied stochastic processes, Brownian motion (Einstein, 1905; Wiener, 1923) and the Ornstein-Uhlenbeck process (Uhlenbeck and Ornstein, 1930), are instances of GPs. In the context of regression and statistical learning, GPs have been used extensively in applications that arise in geostatistics and experimental design (O'Hagan, 1978; Wahba, 1990; Cressie, 1993; Stein, 1999). More recently, in the machine learning literature, GPs have been considered as general estimation tools for solving problems such as nonlinear regression and classification (Rasmussen and Williams, 2006). In the context of machine learning, GPs offer a flexible non-parametric Bayesian framework for estimating latent functions from data and they share similarities with neural networks (Neal, 1996) and kernel methods (Schölkopf and Smola, 2002).

In standard GP regression, where the likelihood is Gaussian, the posterior over the latent function (given data and hyperparameters) is described by a new GP that is obtained analytically. In all other cases, where the likelihood function is non-Gaussian, exact inference is intractable and approximate inference methods are needed. Deterministic approximate methods are currently widely used for inference in GP models (Williams and Barber, 1998; Gibbs and MacKay, 2000; Csato and Opper, 2002; Rasmussen and Williams, 2006; Kuss and Rasmussen, 2005; Rue et al., 2009). However, they are somehow limited since they rely on the assumption that the likelihood function factorizes. In addition, these methods usually treat the hyperparameters of the model (the parameters that appear in the likelihood and the kernel function) in a non full Bayesian way by providing only point estimates. When more complex GP models are considered that may have non-factorizing and heavily parametrized likelihood functions, the development of useful deterministic methods is much more difficult. Complex GP models can arise in time-series applications, where the association of the latent function with the observed data can be described, for instance, by a system of ordinary differential equations. An application of this type has been recently considered in systems biology (Alon, 2006) where the latent function is a transcription factor protein that influences through time the mRNA expression level of a set of target genes (Barenco et al., 2006; Rogers et al., 2006; Lawrence et al., 2007). In this chapter, we discuss Markov chain Monte Carlo (MCMC) algorithms for inference in GP models. An advantage

[1]University of Manchester

of MCMC over deterministic approximate inference is that it provides an arbitrarily precise approximation to the posterior distribution in the limit of long runs. Another advantage is that the sampling scheme will often not depend on details of the likelihood function, and is therefore very generally applicable.

In order to benefit from the advantages of MCMC it is necessary to develop efficient sampling strategies. This has proved to be particularly difficult in many GP applications that involve the estimation of a smooth latent function. Given that the latent function is represented by a discrete set of values, the posterior distribution over these function values can be highly correlated. The larger discrete representations of the function are used, the worse the problem of high correlation becomes. Therefore, simple MCMC schemes such as Gibbs sampling can often be very inefficient. In this chapter, we introduce two MCMC algorithms for GP models that can be more effective in sampling from highly correlated posterior GPs. The first algorithm is a block-based Metropolis-Hastings technique, where the latent function variables are partitioned into disjoint groups corresponding to different function regions. The algorithm iteratively samples each function region by conditioning on the remaining part of the function. The construction of the proposal distribution requires the partitioning of the function points into groups. This is achieved by an adaptive process performed in the early stage of MCMC. The block-based Metropolis-Hastings scheme can improve upon the Gibbs sampler, but it is still not so satisfactory in dealing with highly correlated posterior GPs. Therefore, we introduce a more advanced scheme that uses control variables. These variables are auxiliary function points which are chosen to provide an approximate low dimensional summary of the latent function. We consider Metropolis-Hastings updates that firstly propose moves in the low dimensional representation space and then globally sample the function. The design parameters of the control variables, i.e. their input locations, are found by minimizing an objective function which is the expected least squares error of reconstructing the function values from the control variables, where the expectation is under the GP prior. The number of control variables required to construct the proposal distribution is found automatically by an adaptive process performed during the early iterations of the Markov chain. This sampling algorithm has been previously presented in (Titsias et al., 2009).

Furthermore, we review other sampling algorithms that have been applied to GPs models such as schemes based on variable transformation and Hybrid Monte Carlo (Duane et al., 1987). In the context of sampling, we also discuss the problem of inference over large datasets faced by all GP models due to an unfavourable time complexity $O(n^3)$ where n is the number of function values needed in the GP model.

In our experimental study, we firstly demonstrate the MCMC algorithms on regression and classification problems. As our main application, we consider a problem in systems biology where we wish to estimate the concentration function of a transcription factor protein that regulates a set of genes. The relationship between the protein and the target genes is governed by a system of ordinary differential equations in which the concentration of the protein is an unobserved time-continuous function. Given a time-series of observed gene expression mRNA measurements and assuming a GP prior over the protein concentration, we apply Bayesian inference using MCMC. This allows us to infer the protein concentration function together with other unknown kinetic parameters that appear in the differential equations.

The remainder of this chapter is as follows. Section 1.2 gives an introduction to GP models used in statistical learning, while section 1.3 gives a brief overview of deterministic approximate inference algorithms applied to GP models. Section 1.4 describes sampling algorithms and section 1.5 discusses related work. Section

1.6 demonstrates the sampling methods on regression and classification problems, while section 1.7 gives a detailed description of the application to the regulation of gene transcription. Section 1.8 deals with sampling methods for large GP models. The chapter concludes with a discussion in section 1.9.

1.2 Gaussian process models

A Gaussian process is a stochastic process, that is a set of random variables $\{f(\mathbf{x})|\mathbf{x} \in \mathcal{X}\}$, where $\mathcal{X}$ is an index set, for which any finite subset follows a Gaussian distribution. To describe a GP, we only need to specify the mean function $m(\mathbf{x})$ and a covariance or kernel function $k(\mathbf{x}, \mathbf{x}')$:

$$m(\mathbf{x}) = \mathsf{E}(f(\mathbf{x})), \tag{1.1}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathsf{E}((f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))), \tag{1.2}$$

where $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. GPs naturally arise in the study of time-continuous stochastic processes (Doob, 1953; Wang and Uhlenbeck, 1945). In the context of statistical learning, the practical use of GPs stems from the fact that they provide flexible ways of specifying prior distributions over real-valued functions that can be used in a Bayesian estimation framework. In this section, we give a brief introduction to GPs models in the context of statistical learning. For extensive treatments see, for example, Rasmussen and Williams (2006).

Suppose we wish to estimate a real-valued function $f(\mathbf{x})$. We assume that $\mathbf{x} \in \mathbb{R}^D$ and D is the dimensionality of the input space. We consider a GP model as the prior over the latent function $f(\mathbf{x})$, where for simplicity the mean function $m(\mathbf{x})$ is set to be equal to zero. This prior imposes stronger preferences for certain types of functions compared to others which are less probable. For instance, the prior may favour smooth or stationary functions, or functions with certain lengthscales. All this is reflected in the choice of the kernel $k(\mathbf{x}, \mathbf{x}')$, which essentially captures our prior beliefs about the function we wish to estimate. The kernel $k(\mathbf{x}, \mathbf{x}')$ must be positive definite and can be chosen to fall within a parametric family so as the values of the hyperparameters $\boldsymbol{\theta}$ further specify a member in this family. A common choice is the squared-exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}')\right\}, \tag{1.3}$$

where σ_f^2 is the kernel variance parameter and Σ is a positive definite matrix. Special cases of this kernel are often used in practise. For instance, Σ can be chosen to be diagonal, $\Sigma = \text{diag}[\ell_1^2, \ldots, \ell_D^2]$, where each diagonal element is the lengthscale parameter for a given input dimension. This can be useful in high-dimensional input spaces, where by estimating the lengthscales we can learn to ignore irrelevant input dimensions that are uncorrelated with the output signal (Rasmussen and Williams, 2006; Neal, 1996). The above type of kernel function defines a GP model that generates very smooth (infinitely many times differentiable) functions. This can be particularly useful for general purpose learning problems such as those that arise in machine learning applications. Other type of kernel function such as the Matérn class are often used (Abrahamsen, 1997; Stein, 1999; Rasmussen and Williams, 2006). There are also operations such addition, multiplication and convolution that allow us to create new valid kernels from old ones.

Having chosen a GP prior over the latent function we would like to combine this with observed data, through a Bayesian formalism, and obtain a posterior over this

function. When the data consist of noisy realizations of the latent function and the noise is Gaussian, the above framework has an analytical solution. In particular, let $(X, \mathbf{y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be a set of data where $\mathbf{x}_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}$. Each y_i is produced by adding Gaussian noise to the latent function at input $\mathbf{x}_i$:

$$y_i = f_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2),$$

where $f_i = f(\mathbf{x}_i)$. This defines a Gaussian likelihood model $p(\mathbf{y}|\mathbf{f}) = N(\mathbf{y}|\mathbf{f}, \sigma^2 I)$, where $\mathbf{f} = (f_1, \ldots, f_n)$. The marginalization property of GPs allows simplification of the prior over the latent function which initially is an infinite dimensional object. After marginalization of all function points not associated with the data, we obtain a n-dimensional Gaussian distribution, $p(\mathbf{f}) = N(\mathbf{f}|\mathbf{0}, K_{f,f})$, where $\mathbf{0}$ denotes the n-dimensional zero vector and $K_{f,f}$ is the $n \times n$ covariance matrix obtained by evaluating the kernel function on the observed inputs. Overall, the joint probability model takes the form

$$p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}). \tag{1.4}$$

Notice that this model is non-parametric as the dimension of the (parameter) $\mathbf{f}$ grows linearly with the number of data points. By applying Bayes' rule we can obtain the posterior over $\mathbf{f}$:

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f}}, \tag{1.5}$$

which can be used to obtain the prediction of any quantity of interest. For instance, the function values $\mathbf{f}_*$ at any set of unseen inputs X_* are computed according to:

$$p(\mathbf{f}_*|\mathbf{y}) = \int p(\mathbf{f}_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f}, \tag{1.6}$$

where $p(\mathbf{f}_*|\mathbf{f})$ is the conditional GP prior given by

$$p(\mathbf{f}_*|\mathbf{f}) = N(\mathbf{f}_*|K_{f_*,f}K_{f,f}^{-1}\mathbf{f}, K_{f_*,f_*} - K_{f_*,f}K_{f,f}^{-1}K_{f*,f}^\top). \tag{1.7}$$

Here, the covariance matrix K_{f_*,f_*} is obtained by evaluating the kernel function on the inputs X_* and the cross-covariance matrix $K_{f_*,f}$ is obtained by evaluating for X_* and X. The prediction of the values $\mathbf{y}_*$ of the output signal corresponding to the latent points $\mathbf{f}_*$ is given by $p(\mathbf{y}_*|\mathbf{y}) = \int p(\mathbf{y}_*|\mathbf{f}_*)p(\mathbf{f}_*|\mathbf{y})d\mathbf{f}_*$. In the regression case, where the likelihood is Gaussian, all the above computations are analytically tractable and give rise to Gaussian distributions. Furthermore, the posterior over the latent function can be expressed as a new GP with an updated mean and kernel function. Thus, the counterparts of eq. (1.1) and (1.2) for the posterior GP are given by

$$m_{\mathbf{y}}(\mathbf{x}) = k(\mathbf{x}, X)(\sigma^2 I + K_{f,f})^{-1}\mathbf{y}, \tag{1.8}$$

$$k_{\mathbf{y}}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, X)(\sigma^2 I + K_{f,f})^{-1}k(X, \mathbf{x}'). \tag{1.9}$$

where $k(\mathbf{x}, X)$ is a n-dimensional row vector of kernel function values between $\mathbf{x}$ and X, while $k(X, \mathbf{x}) = k(\mathbf{x}, X)^\top$. The above functions fully specify our posterior GP and we can use them directly to compute any quantity of interest. For instance, the mean and the covariance matrix of the predictive Gaussian $p(\mathbf{f}_*|\mathbf{y})$ in eq. (1.6) is simply obtained by evaluating the above at the inputs X_*.

The posterior GP depends on the values of the kernel parameters $\boldsymbol{\theta}$ as well as the likelihood parameters. To make our notation explicit, we write the likelihood as $p(\mathbf{y}|\mathbf{f}, \boldsymbol{\alpha})$, with $\boldsymbol{\alpha}$ being the parameters of the likelihood[2], and the GP prior as $p(\mathbf{f}|\boldsymbol{\theta})$. The quantities $(\boldsymbol{\alpha}, \boldsymbol{\theta})$ are the hyperparameters of the GP model which have to be specified in order to obtain a close fit to the observed data. A common practise in machine learning is to follow an empirical Bayes approach and choose these parameters by maximizing the marginal likelihood:

$$p(\mathbf{y}|\boldsymbol{\alpha}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f}, \boldsymbol{\alpha}) p(\mathbf{f}|\boldsymbol{\theta}) d\mathbf{f}.$$

When the likelihood is Gaussian this quantity is just a Gaussian distribution which can be maximized over $(\boldsymbol{\alpha}, \boldsymbol{\theta})$ by applying a continuous optimization method. A full Bayesian treatment of the hyperparameters requires the introduction of corresponding prior distributions and an estimation procedure based on MCMC; see section 1.4.5 for further discussion of this issue.

1.3 Non-Gaussian likelihoods and deterministic methods

The above framework, while flexible and conceptually simple, it is computationally tractable only when the likelihood function $p(\mathbf{y}|\mathbf{f}, \boldsymbol{\alpha})$ is Gaussian. When the likelihood is non-Gaussian, computations become intractable and quantities such as the posterior $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$ and the marginal likelihood $p(\mathbf{y}|\boldsymbol{\alpha}, \boldsymbol{\theta})$ are not available in closed form. Clearly, the posterior process over the latent function $f(\mathbf{x})$ is not a GP any more. In such cases we need to consider approximate inference methods. Before describing MCMC methods in section 1.4, we give a brief overview of deterministic approximate inference methods and highlight some of their limitations.

Deterministic methods are widely used for approximate inference in GP models, especially in the machine learning community. Three different algorithms used are the Laplace approximation (Williams and Barber, 1998), the expectation-propagation algorithm (Minka, 2001; Csato and Opper, 2002; Lawrence et al., 2002; Kuss and Rasmussen, 2005; Seeger, 2003) and the variational Gaussian approximation (Opper and Archambeau, 2009). For instance, in binary GP classification, the expectation-propagation algorithm seems to be accurate (Kuss and Rasmussen, 2005). Deterministic methods are also recently discussed in the statistics literature in the context of Gaussian Markov random fields (Rue et al., 2009). All of these methods rely heavily on GP models that have a factorizing likelihood function, i.e. $p(\mathbf{y}|\mathbf{f}, \boldsymbol{\alpha}) = \prod_{i=1}^{n} p(y_i|f_i)$, where each likelihood factor $p(y_i|f_i)$ depends on a single function value f_i, and there is no sharing of function points across factors. Based on these assumptions, the conditional posterior is written in the form

$$p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y}) \propto \exp\left\{ \sum_{i=1}^{N} \log p(y_i|f_i) - \frac{1}{2} \mathbf{f}^T K_{f,f}^{-1} \mathbf{f} \right\}. \tag{1.10}$$

All alternative methods approximate this posterior by a Gaussian distribution. They differ in the way such a Gaussian is obtained. For instance, the Laplace method replaces each factor $\log p(y_i|f_i)$ with a quadratic approximation, based on a Taylor series, and applies continuous optimization to locate the mode of $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$. The expectation-propagation algorithm and the variational method

[2] For the regression case $\boldsymbol{\alpha}$ consists only of σ^2.

also use iterative procedures, while being somehow more advanced as they minimize some divergence between a Gaussian approximation and the exact posterior. These methods will often be reasonably accurate especially when the conditional posterior $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$ is uni-modal. Note, however, that the marginal posterior $p(\mathbf{f}|\mathbf{y}) = \int p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y}) p(\boldsymbol{\alpha}, \boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\alpha} d\boldsymbol{\theta}$ will generally be multi-modal even for the standard regression case. The hyperparameters $(\boldsymbol{\alpha}, \boldsymbol{\theta})$ are typically estimated based on empirical Bayes, where point estimates are obtained by maximizing an approximation to the marginal likelihood $p(\mathbf{y}|\boldsymbol{\alpha}, \boldsymbol{\theta})$. More recently a deterministic method, the nested Laplace approximation (Rue et al., 2009), considers a full Bayesian methodology where the hyperparameters are integrated out by applying numerical integration. However, this method can handle only a small number of hyperparameters (less than six).

In complex GP models, with non-factorizing likelihood functions, it is not clear how to apply the current deterministic methods[3]. Such a complex form of likelihood arises in the application described in section 1.7 that concerns inference of transcription factors in gene regulation. This problem involves a dynamical model derived by solving a systems of ODEs. Furthermore, in this model the number of likelihood parameters $\boldsymbol{\alpha}$ can be large (84 in one example given in section 1.7) and it is of great importance to estimate confidence intervals for those parameters through a full Bayesian methodology. Note that the method described by Rue et al. (2009) that considers full Bayesian inference is not applicable in this case, not only because it assumes a factorizing likelihood but also because it assumes a small number of hyperparameters.

Instead of using deterministic inference algorithms, we can consider stochastic methods based on MCMC. Efficient MCMC methods can reliably deal with complex GP models, having non-factorizing likelihoods, and unlike deterministic methods they benefit from a arbitrarily precise approximation to the true posterior in the limit of long runs. In the next section we discuss MCMC algorithms.

1.4 Sampling algorithms for Gaussian Process models

A major concern with the development of MCMC algorithms in GP models is how to efficiently sample from the posterior conditional $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$. This posterior involves a high-dimensional random variable, consisting of function values that can be highly correlated with one another.

In this section, we describe several sampling schemes that can simulate from $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$ given that the hyperparameters obtain some arbitrary, but fixed, values. In order for our presentation to be instructive, we start with simple schemes such as Gibbs sampling (section 1.4.1) and move to more advanced schemes using block-based Metropolis-Hastings (section 1.4.2) and control variables (section 1.4.3). All these methods can easily be generalized to incorporate steps that can also simulate from $(\boldsymbol{\alpha}, \boldsymbol{\theta})$ as discussed in section 1.4.5. To simplify our notation in the next three sections we omit reference to the hyperparameters.

1.4.1 Gibbs sampling and independent Metropolis-Hastings

The MCMC algorithm we consider is the general Metropolis-Hastings (MH) algorithm (Robert and Casella, 2004; Gelman et al., 2004). Suppose we wish to sample

[3]This is true for the expectation-propagation, variational Gaussian approximation and nested Laplace method which seem to depend on the assumption of having a factorizing likelihood. The Laplace approximation is, of course, generally applicable.

from the posterior in eq. (1.5). The MH algorithm forms a Markov chain. We initialize $\mathbf{f}^{(0)}$ and we consider a proposal distribution $Q(\mathbf{f}^{(t+1)}|\mathbf{f}^{(t)})$ that allows us to draw a new state given the current state. The new state is accepted with probability $\min(1, A)$ where

$$A = \frac{p(\mathbf{y}|\mathbf{f}^{(t+1)})p(\mathbf{f}^{(t+1)})}{p(\mathbf{y}|\mathbf{f}^{(t)})p(\mathbf{f}^{(t)})} \frac{Q(\mathbf{f}^{(t)}|\mathbf{f}^{(t+1)})}{Q(\mathbf{f}^{(t+1)}|\mathbf{f}^{(t)})}. \tag{1.11}$$

To apply this generic algorithm, we need to choose the proposal distribution Q. For GP models, finding a good proposal distribution is challenging since $\mathbf{f}$ is high dimensional and the posterior distribution can be highly correlated. Despite that, there is a lot of structure in a GP model, specifically in the prior $p(\mathbf{f})$, that can greatly facilitate the selection of a good proposal distribution.

To motivate the algorithms presented in section 1.4.2, and 1.4.3, we firstly discuss two extreme options for specifying the proposal distribution Q. One simple way to choose Q is to set it equal to the GP prior $p(\mathbf{f})$ so that the proposed state is independent of the current one. This gives us an independent MH algorithm (Robert and Casella, 2004). However, sampling from the GP prior is very inefficient since it ignores the posterior structure induced by the data leading to a low acceptance rate. Thus the Markov chain will get stuck in the same state for thousands of iterations. On the other hand, sampling from the prior is appealing because any generated sample satisfies the smoothness requirement imposed by the kernel function. Functions drawn from the posterior GP should satisfy the same smoothness requirement as well. It would be interesting to design proposal distributions that can possess this property but simultaneously allow us to increase the acceptance rate.

The other extreme choice for the proposal, that has been considered by Neal (1997), is to apply Gibbs sampling where we iteratively draw samples from each posterior conditional density $p(f_i|\mathbf{f}_{\setminus i}, \mathbf{y})$ with $\mathbf{f}_{\setminus i} = \mathbf{f} \setminus f_i$. This scheme is feasible only when each conditional is log-concave and the adaptive rejection sampling method (Gilks and Wild, 1992) can be used. This will often be the case for models with a factorizing likelihood, where $p(f_i|\mathbf{f}_{\setminus i}, \mathbf{y}) \propto p(y_i|f_i)p(f_i|\mathbf{f}_{\setminus i})$. Any sample in the Gibbs algorithm is accepted with probability one. However, Gibbs sampling can be extremely slow for densely discretized or sampled functions, as in the regression problem of Figure 1.1, where the posterior distribution over $\mathbf{f}$ becomes highly correlated. To clarify this, note that the variance of the posterior conditional $p(f_i|\mathbf{f}_{\setminus i}, \mathbf{y})$ will typically be smaller than the variance of the conditional GP prior $p(f_i|\mathbf{f}_{\setminus i})$. However, $p(f_i|\mathbf{f}_{\setminus i})$ may already have a tiny variance caused by the conditioning on all remaining latent function values. The more densely sampled a function is (relative to the lengthscale of the kernel function), the more inefficient the Gibbs algorithm becomes since the variance of $p(f_i|\mathbf{f}_{\setminus i})$ tends to zero. For the one-dimensional example in Figure 1.1, Gibbs sampling is practically not useful. We study this issue further in section 1.6.

To obtain an algorithm similar to Gibbs sampling but without requiring the use of adaptive rejection sampling, we can consider as the proposal distribution in the MH algorithm the sequence of the conditional densities $p(f_i|\mathbf{f}_{\setminus i})$. Thus, we replace the posterior conditional $p(f_i|\mathbf{f}_{\setminus i}, \mathbf{y})$ with the prior conditional $p(f_i|\mathbf{f}_{\setminus i})$. We call this algorithm, which has been used in geostatistics (Diggle et al., 1998), the Gibbs-like algorithm. This algorithm can exhibit a high acceptance rate, but it is inefficient to sample from highly correlated functions for reasons discussed above.

A common technique used to improve the slow mixing of the Gibbs-type of algorithms when sampling from a high dimensional posterior distribution is to cluster the variables into separate groups and sample all variables of a group within

a single MH step based on an appropriately defined proposal distribution. Given that different groups of variables are weakly correlated, such a scheme can be more effective. Next we describe the local region sampling algorithm which is a way of implementing this idea for GP models.

1.4.2 Sampling using local regions

We now introduce a simple generalization of the Gibbs-like algorithm that is more appropriate for sampling from smooth functions. The idea here is to divide the domain of the function into regions and sample the entire function within each region.

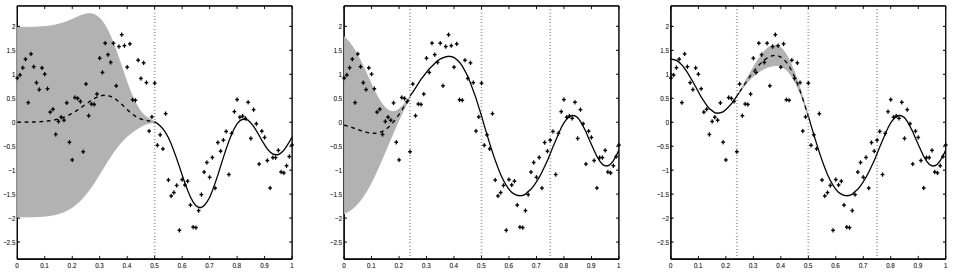
We wish to divide the domain of the function into local regions and sample these local regions iteratively. Let $\mathbf{f}_k$ denote the function points that belong to the local region k, where $k = 1, \ldots, M$ and $\mathbf{f}_1 \cup \ldots \cup \mathbf{f}_M = \mathbf{f}$. New values for the region k are proposed by drawing from the conditional GP prior $p(\mathbf{f}_k^{t+1}|\mathbf{f}_{\setminus k}^{(t)})$, where $\mathbf{f}_{\setminus k} = \mathbf{f} \setminus \mathbf{f}_k$, by conditioning on the remaining function values. $\mathbf{f}_k^{(t+1)}$ is accepted with probability $\min(1, A)$ where

$$A = \frac{p(\mathbf{y}|\mathbf{f}_k^{(t+1)}, \mathbf{f}_{\setminus k}^{(t)})}{p(\mathbf{y}|\mathbf{f}_k^{(t)}, \mathbf{f}_{\setminus k}^{(t)})}. \tag{1.12}$$

Sampling $\mathbf{f}_k$ is iterated between all different regions $k = 1, \ldots, M$. Note that the terms associated with the GP prior cancel out from the acceptance probability since sampling from the conditional prior ensures that any proposed sample is invariant to the GP prior. Given that the initial state $\mathbf{f}^{(0)}$ is a sample from the prior, any proposed function region leads to a possible sample drawn from the GP prior. Notice that sampling from the GP prior and the Gibbs-like algorithm are two extreme cases of the above scheme.

To apply the algorithm, we need to partition the function values $\mathbf{f}$ into groups. This process corresponds to adaption of the proposal distribution and can be carried out during the early iterations of MCMC. An adaptive scheme can start with a small number of clusters, so that the acceptance rate is very low, and then refine the initial clusters in order to increase the acceptance rate. Following the widely used ideas in the theory of adaptive MCMC (Gelman et al., 1996; Roberts et al., 1996; Haario et al., 2001) and Atchade et al. (in this volume) according to which desirable acceptance rates of MH algorithms are around 1/4, we require the algorithm to sample with acceptance rate close to that value.

More specifically, the adaption process is as follows. We obtain an initial partitioning of the vector $\mathbf{f}$ by clustering the inputs X using the k-means algorithm. Then we start the simulation and observe the local acceptance rate r_k associated with the proposal $p(\mathbf{f}_k|\mathbf{f}_{\setminus k})$. Each r_k provides information about the variance of the proposal distribution relative to the local characteristics of the function in region k. A small r_k implies that $p(\mathbf{f}_k|\mathbf{f}_{\setminus k})$ has high variance and most of the generated samples are outside of the support of the posterior GP; see Figure 1.1 for an illustrative example. Hence, when r_k is small, we split the cluster k into two clusters by locally applying the k-means algorithm using all the inputs previously assigned to the initial cluster k. Clusters that have high acceptance rate are unchanged. This hierarchical partitioning process is recursively repeated until all of the current clusters exhibit a local acceptance rate larger than the predefined threshold 1/4. Notice that the above partitioning process can be characterized as supervised in the sense that the information provided by the MH steps is used to decide which



(a) (b) (c)

Figure 1.1: Illustration of the hierarchical clustering process. The panel in (a) shows the variance (displayed with shaded two standard errors bars) of the initial conditional GP prior where we condition on the right side of the function. Since the variance is high the generated local parts of the function will not fit the data often. Dividing the local input region in (a) into two smaller groups (plots (b) and (c)) results a decrease of the variance of the newly formed GP conditional priors and gives an increase in the acceptance rate. However, notice that the variance of the proposal distribution in the boundaries between different function regions is always small. This can affect the efficiency of the sampling algorithm.

clusters need to be further split into smaller groups. Figure 1.1 gives an illustration of the adaptive partitioning process in an one-dimensional regression problem.

Once the adaption of the proposal distribution has ended, we can start sampling from the posterior GP model. The final form of the proposal distribution is a partition of the vector $\mathbf{f}$ into M disjoint groups and the conditional GP prior is the proposal distribution for each group.

As shown in section 1.6, the local region algorithm improves upon the Gibbs sampler. However, this scheme will still be inefficient to sample from highly correlated posteriors since the variance of the proposal distribution can become very small close to the boundaries between neighbouring function regions as illustrated in Figure 1.1. In such cases, there will be variables belonging to different groups which are highly correlated with respect to the GP prior distribution. Of course, these variables will be also highly correlated in terms of the GP posterior. Therefore, the boundaries between function regions can cause the state vector $\mathbf{f}^{(t)}$ to move with a rather small speed when exploring the probability mass, which will lead the Markov chain to mix poorly. Next we describe a sampling algorithm using auxiliary variables, called control points, which attempts to resolve the problems encountered by the local region sampling method and sample more efficiently from highly correlated posterior GPs.

1.4.3 Sampling using control variables

The algorithm described previously is a local sampler that samples each part of the function by conditioning on the remaining part of the function. As discussed previously this can result in a slow exploration of the probability density. To resolve the problem of local sampling we would like to sample the function in a more global sense. Next we discuss an algorithm that achieves this by making use of auxiliary variables.

Let $\mathbf{f}_c$ be a set of M auxiliary function values that are evaluated at inputs X_c and drawn from the GP prior. We call $\mathbf{f}_c$ the control variables and their meaning is analogous to the auxiliary inducing variables used in sparse GP models (Snelson and Ghahramani, 2006; Quiñonero Candela and Rasmussen, 2005). To compute

the posterior $p(\mathbf{f}|\mathbf{y})$ based on control variables we use the expression

$$p(\mathbf{f}|\mathbf{y}) = \int_{\mathbf{f}_c} p(\mathbf{f}|\mathbf{f}_c, \mathbf{y}) p(\mathbf{f}_c|\mathbf{y}) d\mathbf{f}_c. \tag{1.13}$$

Assuming that $\mathbf{f}_c$ is an approximate sufficient statistic for the parameter $\mathbf{f}$, so that $p(\mathbf{f}|\mathbf{f}_c, \mathbf{y}) \simeq p(\mathbf{f}|\mathbf{f}_c)$, we can approximately sample from $p(\mathbf{f}|\mathbf{y})$ in a two-stage manner: firstly sample the control variables from $p(\mathbf{f}_c|\mathbf{y})$ and then generate $\mathbf{f}$ from the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$. This scheme can allow us to introduce a MH algorithm, where we need to specify only a proposal distribution $q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})$, that will mimic sampling from $p(\mathbf{f}_c|\mathbf{y})$, and always sample $\mathbf{f}$ from the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$. The whole proposal distribution takes the form

$$Q(\mathbf{f}^{(t+1)}, \mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)}) = p(\mathbf{f}^{(t+1)}|\mathbf{f}_c^{(t+1)}) q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)}), \tag{1.14}$$

which is used in the MH algorithm in order to sample from the augmented posterior $p(\mathbf{f}, \mathbf{f}_c|\mathbf{y})$. We should emphasize that this proposal distribution does not define an independent Metropolis-Hastings algorithm. However, it satisfies a certain conditional independence relatioship according to which each proposed state $(\mathbf{f}^{(t+1)}, \mathbf{f}_c^{(t+1)})$ depends only on the previous state of the control points $\mathbf{f}_c^{(t)}$ and not on $\mathbf{f}^{(t)}$. Figure 1.2 illustrates the steps of sampling from this proposal distribution. Each proposed sample is accepted with probability $\min(1, A)$ where A is given by

$$A = \frac{p(\mathbf{y}|\mathbf{f}^{(t+1)}) p(\mathbf{f}_c^{(t+1)})}{p(\mathbf{y}|\mathbf{f}^{(t)}) p(\mathbf{f}_c^{(t)})} \cdot \frac{q(\mathbf{f}_c^{(t)}|\mathbf{f}_c^{(t+1)})}{q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})}. \tag{1.15}$$

where the terms involving the conditional GP prior $p(\mathbf{f}|\mathbf{f}_c)$ cancel out. The usefulness of the above sampling scheme stems from the fact that the control variables can form a low-dimensional representation of the function that does not depend much on the size of $\mathbf{f}$, i.e. on how much densely the function has been discretized. The control points will tend to be less correlated with one another since the distance between pairs of them can be large as illustrated in Figure 1.2. The use of the proposal distribution in eq. (1.14) implies that the speed of the Markov chain, i.e. the ability to perform big moves when sampling $\mathbf{f}$, will crucially depend on how the control variables are sampled from $q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})$. The other part of the proposal distribution draws an $\mathbf{f}^{(t+1)}$ that interpolates smoothly between the control points. Thus, while Gibbs-sampling will move more slowly as we keep increasing the size of $\mathbf{f}$, the sampling scheme using control variables will remain equally efficient in performing big moves. In section 1.4.4 we describe how to select the number M of control variables and the inputs X_c using an adaptive MCMC process. In the remainder of this section we discuss how we set the proposal distribution $q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})$.

A suitable choice for q is to use a Gaussian distribution with diagonal or full covariance matrix. The covariance matrix can be adapted during the burn-in phase of MCMC, for instance using the algorithm by Haario et al. (2001), in order to tune the acceptance rate. Although this scheme is general, it has practical limitations. Firstly, tuning a full covariance matrix is time consuming and in our case this adaption process must be carried out simultaneously with searching for an appropriate set of control variables. Also, since the terms involving $p(\mathbf{f}_c)$ do not cancel out in the acceptance probability in eq. (1.15), using a diagonal covariance for the q distribution has the risk of proposing control variables that may not satisfy the GP prior smoothness requirement. To avoid these problems, we define q by using the GP prior. According to eq. (1.13) a suitable choice for q must mimic the sampling

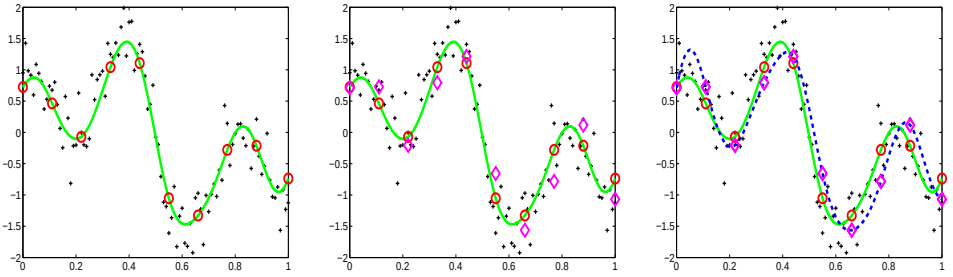


Figure 1.2: Illustration of sampling using control variables. (left) shows the current GP function $\mathbf{f}^{(t)}$ with green, the data and the current location of the control points (red circles). (middle) shows the proposed new positions for the control points (diamonds in magenta). (right) shows the proposed new function values $\mathbf{f}^{(t+1)}$ drawn from the conditional GP prior (blue dotted line).

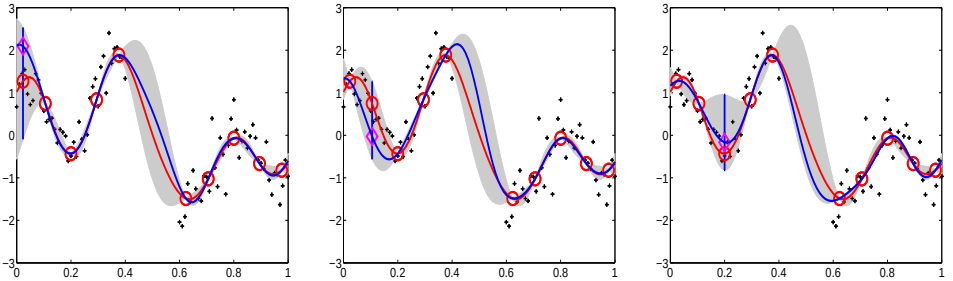


Figure 1.3: Visualization of iterating between control variables. The red solid line is the current $\mathbf{f}^{(t)}$, the blue line is the proposed $\mathbf{f}^{(t+1)}$, the red circles are the current control variables $\mathbf{f}_c^{(t)}$ while the diamond (in magenta) is the proposed control variable $f_{c_i}^{(t+1)}$. The blue solid vertical line represents the distribution $p(f_{c_i}^{(t+1)}|\mathbf{f}_{c_{\setminus i}}^{(t)})$ (with two-standard error bars) and the shaded area shows the effective proposal $p(\mathbf{f}^{t+1}|\mathbf{f}_{c_{\setminus i}}^{(t)})$.

from the posterior $p(\mathbf{f}_c|\mathbf{y})$. Given that the control points are far apart from each other, Gibbs sampling in the control variables space can be efficient. However, iteratively sampling f_{c_i} from the conditional posterior $p(\mathbf{f}_{c_i}|\mathbf{f}_{c_{\setminus i}}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{f}_c)p(f_{c_i}|\mathbf{f}_{c_{\setminus i}})$, where $\mathbf{f}_{c_{\setminus i}} = \mathbf{f}_c \setminus f_{c_i}$ is intractable for non-Gaussian likelihoods[4]. An attractive alternative is to use a Gibbs-like algorithm where each f_{c_i} is drawn from the conditional GP prior $p(\mathbf{f}_{c_i}^{(t+1)}|\mathbf{f}_{c_{\setminus i}}^{(t)})$ and is accepted using the MH step. More specifically, the proposal distribution draws a new $f_{c_i}^{(t+1)}$ for a certain control variable i from $p(f_{c_i}^{(t+1)}|\mathbf{f}_{c_{\setminus i}}^{(t)})$ and generates the function $\mathbf{f}^{(t+1)}$ from $p(\mathbf{f}^{(t+1)}|f_{c_i}^{(t+1)}, \mathbf{f}_{c_{\setminus i}}^{(t)})$. The sample $(f_{c_i}^{(t+1)}, \mathbf{f}^{(t+1)})$ is accepted using the MH step. This scheme of sampling the control variables one-at-a-time and resampling $\mathbf{f}$ is iterated between different control variables. A complete iteration of the algorithm consists of a full scan over all control variables. The acceptance probability A in eq. (1.15) becomes the likelihood ratio and the prior smoothness requirement is always satisfied. The detailed iteration of this sampling method is given in **Algorithm 1** and is illustrated in Figure 1.3.

Although the control variables are sampled one-at-at-time, $\mathbf{f}$ can still be drawn with a considerable variance which does not shrink to zero in certain regions of the

[4]This is because we need to integrate out $\mathbf{f}$ in order to compute $p(\mathbf{y}|\mathbf{f}_c)$.

Algorithm 1 Control-points MCMC

Input: Initial state of control points $\mathbf{f}_c^{(0)}$ and $\mathbf{f}^{(0)}$
repeat
  for $i = 1$ **to** M **do**
    Sample the ith control point: $\mathbf{f}_{c_i}^{(t+1)} \sim p(\mathbf{f}_{c_i}^{(t+1)}|\mathbf{f}_{c_{\setminus i}}^{(t)})$
    Sample $\mathbf{f}^{(t+1)}$: $\mathbf{f}^{(t+1)} \sim p(\mathbf{f}^{t+1}|f_{c_i}^{(t+1)}, \mathbf{f}_{c_{\setminus i}}^{(t)})$
    Accept or reject $(\mathbf{f}^{(t+1)}, f_{c_i}^{(t+1)})$ with the MH probability (likelihood ratio)
  end for
until Convergence of the Markov chain is achieved

input space as happened for the local region sampling algorithm. To clarify this, note that when the control variable f_{c_i} changes, the effective proposal distribution for $\mathbf{f}$ is

$$p(\mathbf{f}^{t+1}|\mathbf{f}_{c_{\setminus i}}^{(t)}) = \int_{f_{c_i}^{(t+1)}} p(\mathbf{f}^{t+1}|f_{c_i}^{(t+1)}, \mathbf{f}_{c_{\setminus i}}^{(t)}) p(f_{c_i}^{(t+1)}|\mathbf{f}_{c_{\setminus i}}^{(t)}) df_{c_i}^{(t+1)}, \tag{1.16}$$

which is the conditional GP prior given all the control points apart from the current point f_{c_i}. This conditional prior can have considerable variance close to f_{c_i} and in all regions that are not close to the remaining control variables. As illustrated in Figure 1.3, the iteration over different control variables allow $\mathbf{f}$ to be drawn with a considerable variance everywhere in the input space whilst respecting the smoothness imposed by the GP prior.

1.4.4 Selection of the control variables

To apply the previous algorithm we need to select the number, M, of the control points and the associated inputs X_c. X_c must be chosen so that knowledge of $\mathbf{f}_c$ can determine $\mathbf{f}$ with small error. The prediction of $\mathbf{f}$ given $\mathbf{f}_c$ is equal to $K_{f,f_c}K_{f_c,f_c}^{-1}\mathbf{f}_c$ which is the mean of the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$. A suitable way to search over X_c is to minimize the reconstruction error $||\mathbf{f} - K_{f,f_c}K_{f_c,f_c}^{-1}\mathbf{f}_c||^2$ averaged over any possible value of $(\mathbf{f}, \mathbf{f}_c)$:

$$G(X_c) = \int_{\mathbf{f},\mathbf{f}_c} ||\mathbf{f} - K_{f,f_c}K_{f_c,f_c}^{-1}\mathbf{f}_c||^2 p(\mathbf{f}|\mathbf{f}_c)p(\mathbf{f}_c)d\mathbf{f}d\mathbf{f}_c = \mathrm{Tr}(K_{f,f} - K_{f,f_c}K_{f_c,f_c}^{-1}K_{f,f_c}^{\top}).$$

The quantity inside the trace is the covariance of $p(\mathbf{f}|\mathbf{f}_c)$ and thus $G(X_c)$ is the total variance of this distribution. We can minimize $G(X_c)$ w.r.t. X_c using continuous optimization similarly to the approach in (Snelson and Ghahramani, 2006). Note that $G(X_c)$ is nonnegative and when it becomes zero, $p(\mathbf{f}|\mathbf{f}_c)$ becomes a delta function, which means that the control variables fully determine $\mathbf{f}$.

To find the number M of control points we minimize $G(X_c)$ by incrementally adding control variables until the total variance of $p(\mathbf{f}|\mathbf{f}_c)$ becomes smaller than a certain percentage of the total variance of the prior $p(\mathbf{f})$. 5% was the threshold used in all our experiments. Then we start the simulation and we observe the acceptance rate of the Markov chain. According to standard approaches (Robert and Casella, 2004; Gelman et al., 2004), which suggest that desirable acceptance rates of MH algorithms are around 1/4, we require a single step of the algorithm to have an acceptance rate around 1/4. When, for the current set of control inputs X_c, the chain has a low acceptance rate, it means that the variance of $p(\mathbf{f}|\mathbf{f}_c)$ is still too high and we need to add more control points in order to further reduce $G(X_c)$. The

process of observing the acceptance rate and adding control variables is continued until we reach the desired acceptance rate.

When the training inputs X are placed uniformly in the space, and the kernel function is stationary, the minimization of G places X_c in a regular grid. In general, the minimization of G places the control inputs close to the clusters of the input data in such a way that the kernel function is taken into account. This suggests that G can also be used for learning inducing variables in sparse GP models (Snelson and Ghahramani, 2006; Seeger et al., 2003) in a unsupervised fashion, where the observed outputs $\mathbf{y}$ are not involved.

1.4.5 Sampling the hyperparameters

Above we discussed algorithms for sampling from the conditional posterior $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$ given a fixed setting of the hyperparameters $(\boldsymbol{\alpha}, \boldsymbol{\theta})$. These parameters, however, are typically unknown and we need to estimate them by following a full Bayesian approach. In particular, we need to assign priors to those parameters, denoted by $p(\boldsymbol{\alpha})$ and $p(\boldsymbol{\theta})$, and sample their values during MCMC by adding suitable updates into all previous MH algorithms. In these updates, we simulate from the conditional posterior distribution $p(\boldsymbol{\alpha}, \boldsymbol{\theta}|\mathbf{f}, \mathbf{y})$ which factorizes across $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, thus yielding two separate conditionals:

$$p(\boldsymbol{\alpha}|\mathbf{f}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{f}, \boldsymbol{\alpha})p(\boldsymbol{\alpha}), \quad p(\boldsymbol{\theta}|\mathbf{f}) \propto p(\mathbf{f}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{1.17}$$

Sampling now from any of these distributions is carried out by using some proposal distribution, for instance a Gaussian, in the MH algorithm. The kernel hyperparameters often take positive values and they can be sampled in the log space. In the experiments in sections 1.6 and 1.7, we use Gaussian proposal distributions which are adapted during the early iterations of MCMC in order to tune the acceptance rate. Furthermore, in the problem of transcriptional gene regulation (see section 1.7), the likelihood parameters $\boldsymbol{\alpha}$ exhibit additional conditional independencies and thus we can sample them independently in separate blocks. Neal (1997) uses Hybrid Monte Carlo (Duane et al., 1987) to sample the hyperparameters in GP models following his earlier work on Bayesian neural networks (Neal, 1996).

An accepted state for the kernel hyperparameters requires an update of the proposal distribution when sampling $\mathbf{f}$. This holds for all algorithms, described previously, that simulate from the conditional posterior $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta})$. For instance, in the algorithm using control variables and for a newly accepted state of the hyperparameters, denoted by $\boldsymbol{\theta}^{(t)}$, the conditional Gaussian $p(\mathbf{f}|\mathbf{f}_c, \boldsymbol{\theta}^{(t)})$ needs to be computed. This requires the estimation of the mean vector of this Gaussian as well as the Cholesky decomposition of the covariance matrix. Finally, we should point out that sampling the kernel hyperparameters can easily become one of the most expensive updates during MCMC, especially when the dimensions of the vector $\mathbf{f}$ is large.

1.5 Related work and other sampling schemes

The MCMC algorithms described in section 1.4.3 and 1.4.2 use an adaptive process which tunes the proposal distribution in order to fit better the characteristics of the posterior distribution. We can classify these algorithms as instances of adaptive MCMC methods (see Atchade et al. in this volume). However, our schemes are specialized to GP models. The most advanced algorithm we presented, that uses control variables, adapts the proposal distribution by finding a set of control variables which somehow provide an approximate low dimensional representation of the

posterior distribution. This way of adaption is rather different to other adaptive MCMC techniques. Perhaps the nearest technique in the literature is the principal directions method described by Andrieu and Thoms (2008).

Regarding other sampling algorithms for GP models, several other schemes seem possible and some have been considered in applications. A sampling method often considered is based on the transformation of the vector $\mathbf{f}$ of function values (Kuss and Rasmussen, 2005). In particular, since much of the correlation that exists in the posterior conditional distribution $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$ is coming from the GP prior, a way to reduce this correlation is to transform $\mathbf{f}$ so that the GP prior is whitened. If L is the Cholesky decomposition of the covariance matrix $K_{f,f}$ of the GP prior $p(\mathbf{f}|\boldsymbol{\theta})$, then the transformation $\mathbf{z} = L^{-1}\mathbf{f}$ defines a new random vector that is white with respect to the prior. Sampling in the transformed GP model can be easier as the posterior over $\mathbf{z}$ can be less correlated than the posterior over $\mathbf{f}$. However, since $\mathbf{z}$ is a high dimensional random variable, the use of a Gaussian proposal distribution in a random walk MH algorithm can be inefficient. This is mainly because of practical difficulties encountered when tuning a full covariance matrix in very high dimensional spaces. Therefore, a more practical approach often considered (Kuss and Rasmussen, 2005), is to sample $\mathbf{z}$ based on the Hybrid Monte Carlo algorithm (Duane et al., 1987). This method uses gradient information and has shown to be effective in sampling in high dimensional spaces (Neal, 1996).

Another common approach to sample the function latent values is to construct a Gaussian approximation to the posterior conditional $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$ and use this as a proposal distribution in the MH algorithm (Rue and Held, 2005; Christensen et al., 2006; Vanhatalo and Vehtari, 2007). Vanhatalo and Vehtari (2007) further combine this approximation with a transformation of the random variables and a subsequent use of Hybrid Monte Carlo. A Gaussian approximation can be constructed, for instance, by using one of the techniques discussed in section 1.3. This method can be appropriate for specialized problems in which the likelihood function takes a simple factorizing form and the number of the hyperparameters is rather small. Notice that the Gaussian approximation is obtained by fixing the hyperparameters $(\boldsymbol{\alpha}, \boldsymbol{\theta})$ to certain values. However, once new values are sampled for those parameters, the Gaussian approximation can become inaccurate. This is rather more likely to occur when the number of hyperparameters is large and varying their values can significantly affect the shape of the conditional posterior $p(\mathbf{f}|\boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{y})$. To overcome this, we could update the Gaussian approximation in order to accommodate the changes made in the values of the hyperparameters. However, this scheme can be computationally very expensive and additionally we need to make sure that such updates do not affect the convergence of the Markov chain to the correct posterior distribution.

Finally, another simple approach for sampling in a GP model is to use the underrelaxation proposal distribution (Adams et al., 2009; Neal, 1998) according to which the proposed new state $\mathbf{f}^{(t+1)}$ is produced by

$$\mathbf{f}^{(t+1)} = \pi \mathbf{f}^{(t)} + \sqrt{1 - \pi^2}\mathbf{u},$$

where $\mathbf{u}$ is a sample drawn from the GP prior $p(\mathbf{f})$ and $\pi \in [0, 1]$. This procedure leaves the GP prior invariant, so that the MH acceptance probability depends only on the likelihood ratio. The parameter π can be adapted in order to tune the acceptance rate. In a typical problem this adaption process will set π rather very close to 1 so that $\mathbf{f}^{(t+1)}$ will tend to be slightly different than $\mathbf{f}^{(t)}$. A value of π that is very close to 1 can result in a slow mixing behavior especially when the posterior distribution has multiple modes. Nevertheless, we believe that this underrelaxation

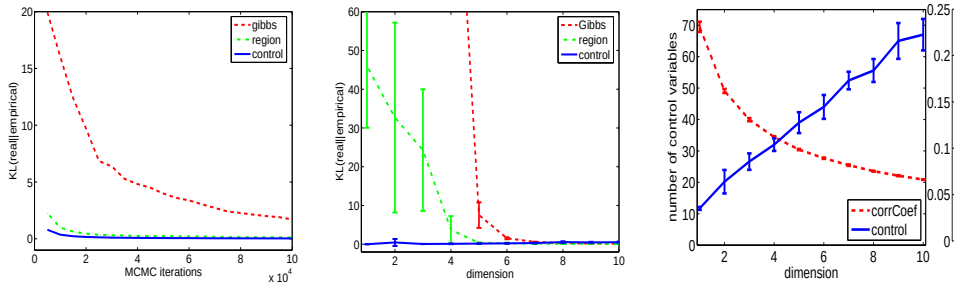


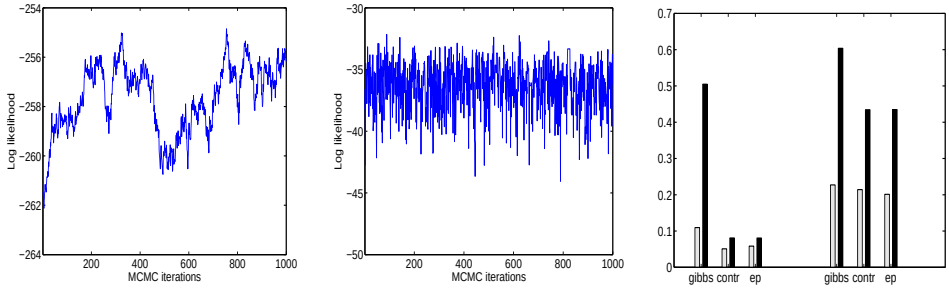
Figure 1.4: (a) shows the evolution of the KL divergence (against the number of MCMC iterations) between the true posterior and the empirically estimated posteriors for a 5-dimensional regression dataset. (b) shows the mean values with one-standard error bars of the KL divergence (against the input dimension) between the true posterior and the empirically estimated posteriors. (c) plots the number of control variables used together with the average correlation coefficient of the GP prior.

scheme can become more promising if combined with the algorithm using control variables where it can provide alternative ways of sampling the control variables. Such a combination is under investigation and it will be presented in a future work.

1.6 Demonstration on regression and classification

In this section, we demonstrate the sampling algorithms on regression and classification problems. In the first experiment we compare Gibbs sampling (*Gibbs*), sampling using local regions (*region*) and sampling using control variables (*control*) in standard regression problems of varied input dimensions. The performance of the algorithms can be accurately assessed by computing the KL divergences between the exact Gaussian posterior $p(\mathbf{f}|\mathbf{y})$ and the Gaussians obtained by MCMC. We fix the number of training points to $N = 200$ and we vary the input dimension d from 1 to 10. The training inputs X were chosen randomly inside the unit hypercube $[0, 1]^d$. This can allow us to study the behavior of the algorithms with respect to the amount of correlation in the posterior GP which is proportional to how densely the function is sampled. The larger the dimension, the sparser the function is sampled. The outputs $\mathbf{y}$ were chosen by randomly producing a GP function using the squared-exponential kernel $\sigma_f^2 \exp(-\frac{||\mathbf{x}_m - \mathbf{x}_n||^2}{2\ell^2})$, where $(\sigma_f^2, \ell^2) = (1, 100)$ and then adding noise with variance $\sigma^2 = 0.09$. The burn-in period was 10^4 iterations[5]. For a certain dimension d the algorithms were initialized to the same state obtained by randomly drawing from the GP prior. The parameters $(\sigma_f^2, \ell^2, \sigma^2)$ were fixed to the values that generated the data. The experimental setup was repeated 10 times so as to obtain confidence intervals. We used thinned samples (by keeping one sample every 10 iterations) to calculate the means and covariances of the 200-dimensional posterior Gaussians. Figure 1.4(a) shows the KL divergence against the number of MCMC iterations for the 5-dimensional input dataset. It seems that for 200 training points and 5 dimensions, the function values are still highly correlated and thus *Gibbs* takes much longer for the KL divergence to drop to zero. Figure 1.4(b) shows the KL divergence against the input dimension after fixing the number of it-

[5]For *Gibbs* we used 2×10^4 iterations since the *region* and *control* algorithms require additional iterations during the adaption phase.



(a) (b) (c)

Figure 1.5: We show results for GP classification. Log-likelihood values are shown for MCMC samples obtained from (a) *Gibbs* and (b) *control* applied to the WBC dataset. In (c) we show the test errors (grey bars) and the average negative log likelihoods (black bars) on the WBC (left) and PID (right) datasets and compare with EP.

erations to be 3×10^4. Clearly *Gibbs* is very inefficient in low dimensions because of the highly correlated posterior. As dimension increases and the functions become sparsely sampled, *Gibbs* improves and eventually the KL divergences approaches zero. The *region* algorithm works better than *Gibbs* but in low dimensions it also suffers from the problem of high correlation. For the *control* algorithm we observe that the KL divergence is very close to zero for all dimensions. Note also that as we increase the number of dimensions *Gibbs* eventually becomes slightly better than the *control* algorithm (for $d = 8$ and onwards) since the function values tend to be independent from one another. Figure 1.4(c) shows the increase in the number of control variables used as the input dimension increases. The same plot shows the decrease of the average correlation coefficient of the GP prior as the input dimension increases. This is very intuitive, since one should expect the number of control variables to increase as the function values become more independent. In the limit when the function values are independent, there will be no accurate low-dimensional representation of the function values and the optimal number of control variables will tend to the number of function values sampled.

Next we consider two GP classification problems for which exact inference is intractable. GP classification involves a factorizing likelihood function. For the binary classification problem each factor $p(y_i|f_i)$ in the likelihood is defined based on the probit or logit model. Deterministic inference methods for GP classification are widely used in machine learning (Williams and Barber, 1998; Csato and Opper, 2002; Lawrence et al., 2002). Among these approaches, the expectation-propagation (EP) algorithm of Minka (2001) is found to be the most efficient (Kuss and Rasmussen, 2005). Our MCMC implementation confirms these findings since sampling using control variables gave similar classification accuracy to EP. We used the Wisconsin Breast Cancer (WBC) and the Pima Indians Diabetes (PID) binary classification datasets. The first consists of 683 examples (9 input dimensions) and the second of 768 examples (8 dimensions). 20% of the examples were used for testing in each case. The MCMC samplers were run for 5×10^4 iterations (thinned to one sample every five iterations) after a burn-in of 10^4 iterations. The hyper-parameters were fixed to those obtained by EP. Figures 1.5(a) and (b) shows the log-likelihood for MCMC samples on the WBC dataset, for the *Gibbs* and *control* algorithms respectively. It can be observed that mixing is far superior for the *control* algorithm and it has also converged to a much higher likelihood. In Figure 1.5(c)

we compare the test error and the average negative log likelihood in the test data obtained by the two MCMC algorithms with the results from EP. The proposed *control* algorithm shows similar classification performance to EP, while the Gibbs algorithm performs significantly worse on both datasets.

1.7 Transcriptional regulation

We consider a small biological sub-system where a set of target genes are regulated by one transcription factor (TF) protein. Ordinary differential equations (ODEs) can provide an useful framework for modelling the dynamics in these biological networks (Alon, 2006; Barenco et al., 2006; Rogers et al., 2006; Lawrence et al., 2007; Gao et al., 2008). The concentration of the TF and the gene specific kinetic parameters are typically unknown and need to be estimated by making use of a time-series of observed gene expression levels. We use a GP prior to model the unobserved TF activity, as proposed by Lawrence et al. (2007), and apply full Bayesian inference based on the MCMC. Next we discuss in detail this method.

Barenco et al. (2006) introduce a linear ODE model for gene activation from TF. This approach was extended by Rogers et al. (2006); Lawrence et al. (2007) to account for non-linear models. The general form of the ODE model for transcription regulation with a single TF has the form

$$\frac{dy_j(t)}{dt} = B_j + S_j g(f(t)) - D_j y_j(t), \tag{1.18}$$

where the changing level of a gene j's expression, $y_j(t)$, is given by a combination of basal transcription rate, B_j, sensitivity, S_j, to its governing TF's activity, $f(t)$, and the decay rate of the mRNA. The function g is typically a non-linear activation function that accounts for phenomena such as gene activation, gene repression and saturation effects. Later in this section, we give specific examples of g functions. Notice also that the TF protein concentration function $f(t)$ takes positive values. The differential equation can be solved for $y_j(t)$ giving

$$y_j(t) = \frac{B_j}{D_j} + \left(A_j - \frac{B_j}{D_j}\right) e^{-D_j t} + S_j e^{-D_j t} \int_0^t g(f(u)) e^{D_j u} du, \tag{1.19}$$

where A_j term arises from the initial condition. Due to the non-linearity of the g function that transforms the TF, the integral in the above expression is not analytically obtained. However, numerical integration can be used to accurately approximate the integral with a dense grid $(u_i)_{i=1}^P$ of points in the time axis and evaluating the function at the grid points $f_p = f(u_p)$. In this case the above equation can be written as

$$y_j(t) = \frac{B_j}{D_j} + \left(A_j - \frac{B_j}{D_j}\right) e^{-D_j t} + S_j e^{-D_j t} \sum_{p=1}^{P_t} w_p g(f_p) e^{D_j u_p}, \tag{1.20}$$

where the weights w_p arise from the numerical integration method used and, for example, can be given by the composite Simpson rule. Notice that the dense grid of function values $\{f_p\}_{p=1}^P$ does not have associated observed output data. As discussed shortly the number of discrete time points in which gene expression measurements are available is much sparser that the set of function points.

The TF concentration $f(t)$ in the above system of ODEs is a latent function that needs to be estimated. Additionally, the kinetic parameters of each gene $\boldsymbol{\alpha}_j = (B_j, D_j, S_j, A_j)$ are unknown and also need to be estimated. To infer these

quantities we use mRNA measurements (obtained from microarray experiments) of N target genes at T different time steps. Let y_{jt} denote the observed gene expression level of gene j at time t and let $\mathbf{y} = \{y_{jt}\}$ collect together all these observations. Assuming a Gaussian noise for the observed gene expressions the likelihood of our data has the form

$$p(\mathbf{y}|\mathbf{f}, \{\boldsymbol{\alpha}_j\}_{j=1}^N) = \prod_{j=1}^{N}\prod_{t=1}^{T} p(y_{jt}|\mathbf{f}_{1\leq p\leq P_t}, \boldsymbol{\alpha}_j, \sigma_j^2), \tag{1.21}$$

where each probability density in the above product is a Gaussian with mean given by eq. (1.20), $\mathbf{f}_{1\leq p\leq P_t}$ denotes the TF values up to time t and σ_j^2 is a gene-specific variance parameter. Notice that there are 5 parameters per gene and thus overall there are $5\times N$ likelihood parameters. The above likelihood function is non-Gaussian due to the non-linearity of g. Further, the above likelihood does not have a factorized form, as in the regression and classification cases, since an observed gene expression depends on the protein concentration activity in all previous times points. Also note that the discretization of the TF in P time points corresponds to a very dense grid, while the gene expression measurements are sparse, i.e. $P \gg T$.

To apply full Bayesian inference in the above model, we need to define prior distributions over all unknown quantities. The protein concentration $\mathbf{f}$ is a positive quantity, thus a suitable prior is to consider a GP prior for $\log \mathbf{f}$. The kernel function of this GP prior is chosen to be the squared-exponential kernel, $\exp(-\frac{1}{2\ell^2}(t-t')^2)$, where the variance of this kernel, the σ_f^2 in eq. (1.3), is fixed to one, which helps to avoid identifiability problems when interacting with the sensitivity parameter S_j. The lengthscale ℓ^2 is assigned a gamma prior. The kinetic parameters of each gene are all positive scalars and are represented in the log space. These parameters are given vague Gaussian priors. Each noise variance σ_j^2 is given a conjugate gamma prior. Sampling the GP function is done exactly as described in section 1.4; we have only to plug in the likelihood from eq. (1.21) in the MH step. Sampling from the kinetic parameters is carried out using Gaussian proposal distributions with diagonal covariance matrices that sample the positive kinetic parameters in the log space. Notice also that the kinetics parameters $\boldsymbol{\alpha}_j$ for gene j are sampled independently from the corresponding parameters of all other genes. This is because the conditional $p(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N|\mathbf{f})$ factorizes across different $\boldsymbol{\alpha}_j$s. Finally each noise variance σ_j^2 is sampled from its gamma conditional posterior.

We now consider two experiments where we apply the algorithm that uses control variables (see section 1.4.3) to infer the protein concentration of TFs that activate or repress a set of target genes. The latent function in these problems is always one-dimensional and densely discretized and as shown in section 1.6 the algorithm using control variables is the only one that can converge to the posterior distribution in a reasonable time.

We first consider the TF p53 which is a tumour repressor activated during DNA damage. According to Barenco et al. (2006), irradiation is performed to disrupt the equilibrium of the p53 network and the transcription of p53 target genes are then stimulated. Seven samples of the expression levels of five target genes in three replicas are collected as the raw time course data. The non-linear activation of the protein follows the Michaelis Menten kinetics inspired response (Alon, 2006) that allows saturation effects to be taken into account so as the g function in eq. (1.18) takes the form

$$g(f(t)) = \frac{f(t)}{\gamma_j + f(t)}, \tag{1.22}$$

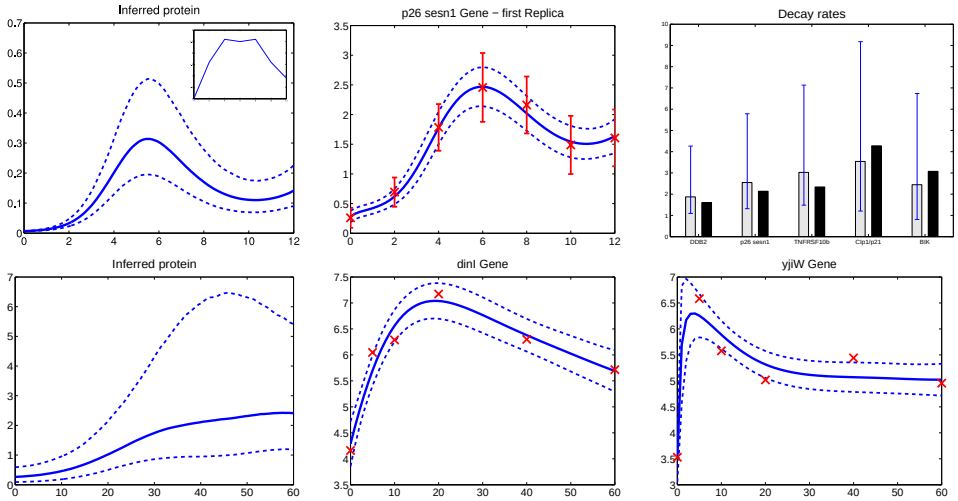


Figure 1.6: **First row:** The left plot shows the inferred TF concentration for p53; the small plot on top-right shows the ground-truth protein concentration obtained by a *Western blot* experiment Barenco et al. (2006). The middle plot shows the predicted expression of a gene obtained by the estimated ODE model; red crosses correspond to the actual gene expression measurements. The right-hand plot shows the estimated decay rates for all 5 target genes used to train the model. Grey bars display the parameters found by MCMC and black bars the parameters found in Barenco et al. (2006) using a linear ODE model. **Second row**: The left plot shows the inferred TF for LexA. Predicted expressions of two target genes are shown in the rest two plots. Error bars in all plots correspond to 95% credibility intervals.

where the Michaelis constant for the jth gene, given by γ_j, is an additional likelihood parameter that is inferred by MCMC. Note that since $f(t)$ is positive the GP prior is placed on the $\log f(t)$. Gene expressions for the genes are available for $T = 7$ different times. To apply MCMC we discretize $\mathbf{f}$ using a grid of $P = 121$ points. During sampling, 7 control variables were needed to obtain the desirable acceptance rate. Running time was 4 hours for 5×10^5 sampling iterations plus 5×10^4 burn-in iterations. Acceptance rate for $\mathbf{f}$ after burn in was between $15\% - 25\%$. The first row of Figure 1.6 summarizes the estimated quantities obtained from MCMC simulation.

Next we consider the TF LexA in E.Coli that acts as a repressor. In the repression case there is an analogous Michaelis Menten model (Alon, 2006) where the non-linear function g takes the form:

$$g(f(t)) = \frac{1}{\gamma_j + f(t)}. \tag{1.23}$$

Again the GP prior is placed on the log of the TF activity. We applied our method to the same microarray data considered by Rogers et al. (2006) where mRNA measurements of 14 target genes are collected over six time points. The amount of LexA is reduced after UV irradiation, decreasing for a few minutes and then recovering to its normal level. For this dataset, the expression of the 14 genes were available for $T = 6$ times. Notice that the number of likelihood parameters in this model is $14 \times 6 = 84$. The GP function $\mathbf{f}$ was discretized using 121 points. The result for the inferred TF profile along with predictions of two target genes are shown in the second row of Figure 1.6. Our inferred TF profile and reconstructed target gene profiles are similar to those obtained in Rogers et al. (2006). However, for certain

genes, our model provides a better fit to the gene profile.

1.8 Dealing with large datasets

The application of GP models becomes intractable when the dimension of the vector of function values $\mathbf{f}$ needed to specify the likelihood is very large. This is because we need to store a large matrix of size $n \times n$ and invert this matrix (see eq. (1.6) and (1.7)) which scales as $O(n^3)$. For regression and classification problems, where the dimension of $\mathbf{f}$ grows linearly with the number of training examples, this is the well-known problem of large datasets (Csato and Opper, 2002; Smola and Bartlett, 2001; Seeger et al., 2003; Snelson and Ghahramani, 2006; Quiñonero Candela and Rasmussen, 2005). Notice that GP models become intractable for large datasets not only in the case of non-Gaussian likelihood functions but also for the standard regression problem; observe that the posterior GP described by eq. (1.8) and (1.9) requires the inversion of a $n \times n$ matrix. Next we discuss how we can deal with the problem of large datasets in the context of MCMC inference. Vanhatalo and Vehtari (2007) have also addressed the same problem.

A simple way to reduce the complexity of the GP model is to decrease the dimension of $\mathbf{f}$. In problems having facrorizing likelihoods, this implies that we have to ignore the large majority of the training examples and use only a small subset of the data. A more advanced strategy is to construct a sparse approximation based on a carefully chosen set of support or inducing variables (Csato and Opper, 2002; Smola and Bartlett, 2001; Seeger et al., 2003; Snelson and Ghahramani, 2006; Quiñonero Candela and Rasmussen, 2005; Titsias, 2009). In the context of MCMC, this framework fits naturally within the sampling scheme that uses control variables which are exactly analogous to the inducing variables. One way to construct an approximate GP model that can deal with a very large dimension of $\mathbf{f}$ is to modify the prior $p(\mathbf{f})$. By using a set of auxiliary control variables $\mathbf{f}_c$, which are function points drawn from the GP, we can write $p(\mathbf{f})$ as

$$p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{f}_c)p(\mathbf{f}_c)d\mathbf{f}_c. \tag{1.24}$$

The intractable term in this expression is the conditional distribution $p(\mathbf{f}|\mathbf{f}_c)$ which has an $n \times n$ full covariance matrix: $K_{f,f} - K_{f,f_c}K_{f_c,f_c}^{-1}K_{f,f_c}^{\top}$. Clearly, we cannot simulate from this conditional Gaussian, because of the prohibitively large full covariance matrix. Therefore, the algorithm using control variables is not computationally tractable. To overcome this problem, we can modify the GP prior by replacing $p(\mathbf{f}|\mathbf{f}_c)$ with a simpler distribution. The simplest choice is to use a delta function centered at the mean of $p(\mathbf{f}|\mathbf{f}_c)$, given by $K_{f,f_c}K_{f_c,f_c}^{-1}\mathbf{f}_c$. This allows to analytically marginalize out $\mathbf{f}$ and obtain the joint probability model:

$$q(\mathbf{y}, \mathbf{f}_c) = \int p(\mathbf{y}|\mathbf{f})\delta(\mathbf{f} - K_{f,f_c}K_{f_c,f_c}^{-1}\mathbf{f}_c)p(\mathbf{f}_c)d\mathbf{f} = p(\mathbf{y}|K_{f,f_c}K_{f_c,f_c}^{-1}\mathbf{f}_c)p(\mathbf{f}_c). \tag{1.25}$$

This modified GP model corresponds to the projected process approximation (Csato and Opper, 2002; Seeger et al., 2003). An MCMC algorithm applied to this model requires only sampling $\mathbf{f}_c$. Further, notice that the control points algorithm (see **Algorithm 1**) in this case reduces to the Gibbs-like algorithm. A more advanced approximation to the GP prior is obtained by the sparse pseudo-inputs GP method of Snelson and Ghahramani (2006) which is also referred as fully independent training conditional (FITC) in (Quiñonero Candela and Rasmussen, 2005). Here, $q(\mathbf{f}|\mathbf{f}_c) = \prod_{i=1}^{n} p(f_i|\mathbf{f}_c)$, where each $p(f_i|\mathbf{f}_c)$ is a marginal conditional prior with

mean $K(\mathbf{x}_i, X_c)K^{-1}_{f_c,f_c}\mathbf{f}_c$ and variance $k(\mathbf{x}_i, \mathbf{x}_i) - K(\mathbf{x}_i, X_c)K^{-1}_{f_c,f_c}K(X_c, \mathbf{x}_i)$. This approximation keeps only the diagonal elements of the covariance matrix of $p(\mathbf{f}|\mathbf{f}_c)$. The algorithm using control points can be applied exactly as described in **Algorithm 1**. Notice that for factorizing likelihoods, the step of sampling $\mathbf{f}$ given $\mathbf{f}_c$ simplifies to n independent problems since the posterior $p(\mathbf{f}|\mathbf{f}_c, \mathbf{y})$ factorizes across the dimensions of $\mathbf{f}$, exactly as the prior. This implies that we could also marginalize out $\mathbf{f}$ numerically in such case. Extensions of the FITC approximation can be considered by representing exactly only small blocks of the covariance matrix of $p(\mathbf{f}|\mathbf{f}_c)$ (Quiñonero Candela and Rasmussen, 2005).

A different approach for sampling in large GP models, is to follow the variational framework (Titsias, 2009; Csato and Opper, 2002; Seeger et al., 2003). In this method, the GP prior $p(\mathbf{f})$ is not modified, but instead a variational distribution is fitted to the exact posterior $p(\mathbf{f}, \mathbf{f}_c|\mathbf{y})$. The variational distribution factorizes as follows

$$q(\mathbf{f}, \mathbf{f}_c) = p(\mathbf{f}|\mathbf{f}_c)\phi(\mathbf{f}_c), \tag{1.26}$$

where the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$ is the one part of the variational distribution, while the other part, $\phi(\mathbf{f}_c)$, is an unknown (generally non-Gaussian) distribution that is defined optimally through the minimization of the Kullback-Leibler divergence between $q(\mathbf{f}, \mathbf{f}_c)$ and the exact posterior $p(\mathbf{f}, \mathbf{f}_c|\mathbf{y})$. The optimal setting for $\phi(\mathbf{f}_c)$ is given by

$$\phi(\mathbf{f}_c) \propto p(\mathbf{f}_c) \exp\left\{\int p(\mathbf{f}|\mathbf{f}_c) \log p(\mathbf{f}|\mathbf{y}) d\mathbf{f}\right\}, \tag{1.27}$$

where we assume that the integral inside the exponential can be either computed analytically or approximated accurately using some numerical integration method. For instance, for a log Gaussian Cox model (Diggle et al., 1998; Rue et al., 2009) this integral can be obtained analytically and generally for factorizing likelihoods the computations involve n independent one-dimensional numerical integration problems. Given that we can integrate out $\mathbf{f}$ in eq. (1.27), we can sample from $\phi(\mathbf{f}_c)$, using for instance the Gibbs-like algorithm. The whole representation of the variational distribution in eq. (1.26) will have an analytic part, the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$, and a numerical part expressed by a set of samples drawn from $\phi(\mathbf{f}_c)$.

Prediction in sparse GP models typically involves some additional approximations that crucially avoid the computations of intractable terms such as the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$. For instance, the prediction of the function values $\mathbf{f}_*$ in some inputs X_*, given by eq. (1.6), can be expressed as $p(\mathbf{f}_*|\mathbf{y}) = \int p(\mathbf{f}_*|\mathbf{f}, \mathbf{f}_c)p(\mathbf{f}, \mathbf{f}_c|\mathbf{y})d\mathbf{f}d\mathbf{f}_c$. However, $p(\mathbf{f}_*|\mathbf{f}, \mathbf{f}_c)$ cannot be computed because of the large dimension of $\mathbf{f}$. Thus, we need to approximate it by replacing it with $p(\mathbf{f}_*|\mathbf{f}_c)$. This allows to further marginalize $\mathbf{f}$ analytically out of the above Bayesian integral and subsequently use only the set of samples corresponding to the control variables in order to obtain a prediction based on Monte Carlo.

1.9 Discussion

Gaussian processes allow for inference over latent functions using a Bayesian nonparametric framework. In this chapter, we discussed MCMC algorithms that can be used for inference in GP models. The more advanced algorithm that we presented uses control variables which act as approximate low dimensional summaries of the function values that we need to sample from. We showed that this sampling scheme can efficiently deal with highly correlated posterior distributions.

MCMC allows for full Bayesian inference in the transcription factor networks application. An important direction for future research will be scaling the models used to much larger systems of ODEs with multiple interacting transcription factors. In such case the GP model becomes much more complicated and several latent functions need to be estimated simultaneously.

Regarding deterministic versus stochastic inference, in simple GP models with factorizing likelihoods and small number of hyperparameters, deterministic methods, if further developed, can lead to reliable inference methods. However, in more complex GP models having non-factorizing likelihood functions and large number of hyperparameters, we believe that MCMC is currently the only reliable way of carrying out accurate full Bayesian inference. Of course, what often decides which will be the approximate inference method used is the application at hand and the context of this application. In a mainstream machine learning application involving large datasets and where fast inference is required, deterministic methods are usually preferred simply because they are faster. In contrast, in applications related to scientific questions that need to be be carefully addressed by carrying out a statistical data analysis, MCMC is preferred.

Acknowledgments

This work is funded by EPSRC Grant No EP/F005687/1 "Gaussian Processes for Systems Identification with Applications in Systems Biology".

Bibliography

Abrahamsen, P. (1997). A review of Gaussian random fields and correlation functions. Technical Report 917, Norwegian Computing Center.

Adams, R. P., Murray, I., and MacKay, D. J. (2009). The Gaussian process density sampler. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 9–16.

Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman and Hall/CRC.

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, 18:343–373.

Barenco, M., Tomescu, D., Brewer, D., Callard, R. Stark, J., and Hubank, M. (2006). Ranked prediction of p53 targets using hidden variable dynamic modeling. *Genome Biology*, 7(3).

Christensen, O. F., Roberts, G. O., and Sköld (2006). Robust Markov chain Monte carlo methods for spatial generalized linear mixed models. *Journal of Computational and Graphical Statistics*, 15:1–17.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. John Wiley & Sons, New Tork.

Csato, L. and Opper, M. (2002). Sparse online Gaussian processes. *Neural Computation*, 14:641–668.

Diggle, P. J., Tawn, J. A., and Moyeed, R. A. (1998). Model-based Geostatistics (with discussion). *Applied Statistics*, 47:299–350.

Doob, J. L. (1953). *Stochastic Processes*. John Wiley and Sons.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Einstein, A. (1905). On the movement of small particles suspended in a stationary liquid by the molecular kinetic theory of heat. Dover Publ.

Gao, P., Honkela, A., Lawrence, N., and Rattray, M. (2008). Gaussian Process Modelling of Latent Chemical Species: Applications to Inferring Transcription Factor Activities . In *ECCB08*.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2004). *Bayesian Data Analysis*. Chapman and Hall.

Gelman, A., Roberts, G. O., and Gilks, W. R. (1996). Efficient metropolis jumping rules. *In Bayesian statistics*, 5.

Gibbs, M. N. and MacKay, D. J. C. (2000). Variational Gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464.

Gilks, W. R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, 41(2):337–348.

Haario, H., Saksman, E., and Tamminen, J. (2001). An adaptive metropolis algorithm. *Bernoulli*, 7:223–240.

Kuss, M. and Rasmussen, C. E. (2005). Assessing Approximate Inference for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 6:1679–1704.

Lawrence, N. D., Sanguinetti, G., and Rattray, M. (2007). Modelling transcriptional regulation using Gaussian processes. In *Advances in Neural Information Processing Systems, 19*. MIT Press.

Lawrence, N. D., Seeger, M., and Herbrich, R. (2002). Fast sparse Gaussian process methods: the informative vector machine. In *Advances in Neural Information Processing Systems, 13*. MIT Press.

Minka, T. (2001). Expectation propagation for approximate Bayesian inference. In *UAI*, pages 362–369.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer, New York. Lecture Notes in Statistics 118.

Neal, R. M. (1997). Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical report, Dept. of Statistics, University of Toronto.

Neal, R. M. (1998). Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. In Jordan, M. I., editor, *Learning in Graphical Models*, pages 205–225. Dordrecht: Kluwer Academic Publishers:.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society, Series B*, 40(1):1–42.

Opper, M. and Archambeau, C. (2009). The variational gaussian approximation revisited. *to appear in Neural Computation*, 21(3).

Quiñonero Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer-Verlag, 2nd edition.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1996). Weak convergence and optimal scaling of random walk metropolis algorithms. *The Annals of Applied Probability*, 7:110–120.

Rogers, S., Khanin, R., and Girolami, M. (2006). Bayesian model-based inference of transcription factor activity. *BMC Bioinformatics*, 8(2).

Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications*. Monographs on Statistics and Applied Probability. Chapman & Hall, London.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, 71 (2):319–392.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels*. MIT Press, Cambridge MA.

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

Seeger, M., Williams, C. K. I., and Lawerence, N. D. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *In C.M. Bishop and B. J. Frey, editors, Proceedings of the Ninth International Workshop on Artificial Intelligence*. MIT Press.

Smola, A. J. and Bartlett, P. (2001). Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems, 13*. MIT Press.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian process using pseudo inputs. In *Advances in Neural Information Processing Systems, 13*. MIT Press.

Stein, M. L. (1999). *Interpolation of Spatial Data*. Springer, New York.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Twelfth International Conference on Artificial Intelligence and Statistics, JMLR: W and CP*, volume 5, pages 567–574.

Titsias, M. K., Lawrence, N. D., and Rattray, M. (2009). Efficient sampling for gaussian process inference using control variables. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1681–1688.

Uhlenbeck, G. E. and Ornstein, L. S. (1930). On the Theory of Brownian Motion. *Physics Review*, 36:823–841.

Vanhatalo, J. and Vehtari, A. (2007). Sparse Log Gaussian Processes via MCMC for Spatial Epidemiology. *Journal of Machine Learing Research: Worskshop and conference proceedings*, 1:73–89.

Wahba, G. (1990). Spline Models for Observational Data. *Society for Industrial and Applied Mathematics*, 59.

Wang, M. C. and Uhlenbeck, G. E. (1945). On the Theory of the Brownian Motion II. *Reviews of Modern Physics*, 17(2-3):323–342.

Wiener, N. (1923). Differential space. *Journal of Mathematical Physics*, 2:131–174.

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