Gaussian Process Regression Networks

Andrew Gordon Wilson* University of Cambridge agw38@cam.ac.uk

David A. Knowles† University of Cambridge dak33@cam.ac.uk

Zoubin Ghahramani‡ University of Cambridge zoubin@eng.cam.ac.uk

Abstract

We introduce a new regression framework, Gaussian process regression networks (GPRN), which combines the structural properties of Bayesian neural networks with the nonparametric flexibility of Gaussian processes. This model accommodates input dependent signal and noise correlations between multiple response variables, input dependent length-scales and amplitudes, and heavy-tailed predictive distributions. We derive both efficient Markov chain Monte Carlo and variational Bayes inference procedures for this model. We apply GPRN as a multiple output regression and multivariate volatility model, demonstrating substantially improved performance over eight popular multiple output (multi-task) Gaussian process models and three multivariate volatility models on benchmark datasets, including a 1000 dimensional gene expression dataset.

1 Introduction

Gaussian process models have become exceptionally popular for solving non-linear regression and classification problems. They are expressive, interpretable, avoid over-fitting, and have impressive predictive performance in many thorough empirical comparisons (Rasmussen, 1996; Kuss and Rasmussen, 2005; Rasmussen and Williams, 2006).

In machine learning, Gaussian process regression developed out of neural networks research. Neal (1996) showed that Bayesian neural networks became Gaussian processes as the number of hidden units approached infinity, and conjectured that "there may be simpler ways to do inference in this case." These simple inference techniques became the cornerstone of subsequent Gaussian process models. However, neural networks had been motivated in part by their ability to capture correlations between multiple outputs (responses), by using adaptive hidden units that were shared between the outputs. In the infinite limit, this ability was lost.

Recently there has been an explosion of interest in extending the Gaussian process regression framework to account for *fixed* correlations between output variables (Alvarez and Lawrence, 2011; Yu et al., 2009; Alvarez and Lawrence, 2008; Bonilla et al., 2008; Osborne et al., 2008; Teh et al., 2005; Boyle and Frean, 2004). These are often called 'multi-task' learning or 'multiple output' regression models. Capturing correlations between outputs (response variables) can be used to make better predictions. Imagine we wish to predict cadmium concentrations in a region of the Swiss Jura, where geologists are interested in heavy metal concentrations. A standard Gaussian process regression model would only be able to use cadmium training measurements. With a multi-task method, we can also make use of correlated heavy metal measurements to enhance cadmium predictions (Goovaerts, 1997). We could further enhance predictions if we make use of how these (signal) correlations change with geographical location.

*http://mlg.eng.cam.ac.uk/andrew
†http://mlg.eng.cam.ac.uk/dave
‡http://mlg.eng.cam.ac.uk/zoubin

There has similarly been great interest in extending Gaussian process (GP) regression to account for input dependent noise variances (Goldberg et al., 1998; Kersting et al., 2007; Adams and Stegle, 2008; Turner and Sahani, 2008; Turner, 2010; Wilson and Ghahramani, 2010a,b; Lázaro-Gredilla and Titsias, 2011). Wilson and Ghahramani (2010b, 2011) and Fox and Dunson (2011) further extended the GP framework to accommodate input dependent noise correlations between multiple output (response) variables.

Other extensions include Gaussian process regression with non-stationary covariance function amplitudes (Turner and Sahani, 2008; Adams and Stegle, 2008) and length-scales (Gibbs, 1997; Schmidt and O'Hagan, 2003), and with heavy tailed predictive distributions (Neal, 1997; Vanhatalo et al., 2009) for outlier rejection (De Finetti, 1956; Dawid, 1973; O'Hagan, 1979).

In this paper, we introduce a new regression framework, Gaussian Process Regression Networks (GPRN), which combines the structural properties of Bayesian neural networks (Neal, 1996) with the nonparametric flexibility of Gaussian processes. This network is an adaptive mixture of Gaussian processes, which naturally accommodates input dependent signal and noise correlations between multiple output variables, input dependent length-scales and amplitudes, and heavy tailed predictive distributions, without expensive or numerically unstable computations.

We start by introducing the GPRN framework, and show how to perform efficient inference using both Markov chain Monte Carlo (MCMC) and variational Bayes (VB). Carefully following Alvarez and Lawrence (2011), we compare to eight multiple output GP models on gene expression and geostatistics datasets. We then compare to multivariate volatility models on several benchmark financial datasets, following Wilson and Ghahramani (2010b). In the Appendix, we review Gaussian process regression and the notation of Rasmussen and Williams (2006).

2 Gaussian Process Regression Networks

Given vector valued data points $\mathcal{D} = \{\boldsymbol{y}(x_i) : i = 1, \ldots, N\}$, where $x \in \mathcal{X}$ is an arbitrary input variable, we aim to predict $\mathbb{E}[\boldsymbol{y}(x_*)|x_*, \mathcal{D}]$ and $\mathrm{cov}[\boldsymbol{y}(x_*)|x_*, \mathcal{D}]$ at a test input x_*. We assume that the noise and signal correlations between the elements of $\boldsymbol{y}(x)$ may change as a function of x. Supposing x is time ($x = t$), a particular component of the p-dimensional vector $\boldsymbol{y}(t)$ could be the expression level of a particular gene at time t, and $\Sigma(t)$ would then represent the variances and correlations for these genes at time t. Instead of assuming only time dependence, we could have a more general input (predictor) variable $x \in \mathcal{X}$. Then we can imagine $\boldsymbol{y}(x)$ representing different heavy metal concentrations at a geographical location $x \in \mathbb{R}^2$. These are two examples from the experiments in Section 4.

We model $\boldsymbol{y}(x)$ as

$$\boldsymbol{y}(x) = W(x)[\boldsymbol{f}(x) + \sigma_f \boldsymbol{\epsilon}] + \sigma_y \boldsymbol{z} \tag{1}$$

where $\boldsymbol{\epsilon}$ and $\boldsymbol{z}$ are i.i.d. $\mathcal{N}(0, I)$ white noise[1], $W(x)$ is a $p \times q$ matrix of independent Gaussian processes such that $W(x)_{ij} \sim \mathcal{GP}(0, k_w)$, and $\boldsymbol{f}(x) = (f_1(x), \ldots, f_q(x))^\top$ is a $q \times 1$ vector of independent GPs with $f_i(x) \sim \mathcal{GP}(0, k_{f_i})$. Each of the latent Gaussian processes in $\boldsymbol{f}(x)$ have additive Gaussian noise. Changing variables to include the noise $\sigma_f \boldsymbol{\epsilon}$ we let $\hat{f}_i(x) \sim \mathcal{GP}(0, k_{\hat{f}_i})$, where

$$k_{\hat{f}_i}(x, x') = k_{f_i}(x, x') + \sigma_f^2 \delta_{xx'}\,, \tag{2}$$

and $\delta_{xx'}$ is the Kronecker delta.

We represent this *Gaussian process regression network* (GPRN) in Figure 1, labelling the length-scale hyperparameters for the kernels k_w and k_f as $\boldsymbol{\theta}_w$ and $\boldsymbol{\theta}_f$ respectively. We see the latent *node functions* $\hat{\boldsymbol{f}}(x)$ are connected together to form the outputs $\boldsymbol{y}(x)$. The strengths of the connections change as a function of x; the weights themselves – the entries of $W(x)$ – are functions. Old connections can break and

[1]The distribution of $\boldsymbol{z}$ could be Student-t, Laplace, or something different. Using diagonal noise would also be a straightforward extension.

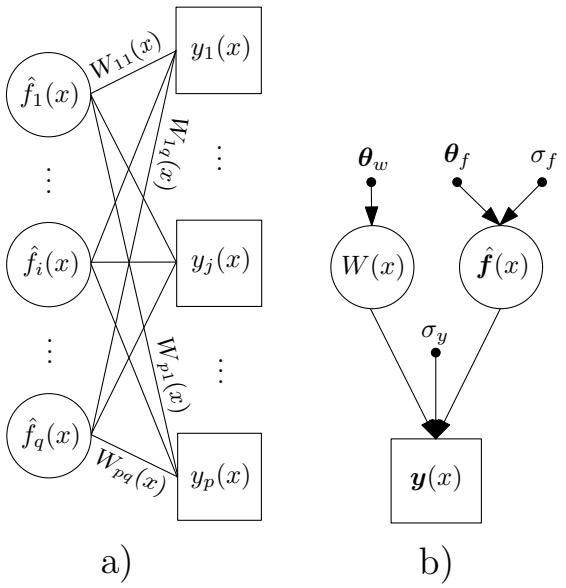

Figure 1: The Gaussian process regression network. Latent random variables and observables are respectively labelled with circles and squares, except for the weight functions in a). Hyperparameters are labelled with dots. a) This neural network style diagram shows the q components of the vector $\hat{\boldsymbol{f}}$ (GPs with additive noise), and the p components of the vector $\boldsymbol{y}$. The links in the graph, four of which are labelled, are latent random weight *functions*. Every quantity in this graph depends on the input x. This graph emphasises the adaptive nature of this network: links can change strength or even disappear as x changes. b) A directed graphical model showing the generative procedure with relevant variables.

new connections can form. This is an *adaptive* network, where the signal and noise correlations between the components of $\boldsymbol{y}(x)$ vary with x. [2]

To explicitly separate the dynamic signal and noise correlations, we re-write (1) as

$$\boldsymbol{y}(x) = \underbrace{W(x)\boldsymbol{f}(x)}_{\text{signal}} + \underbrace{\sigma_f W(x)\boldsymbol{\epsilon} + \sigma_y \boldsymbol{z}}_{\text{noise}} . \tag{3}$$

Conditioning on $W(x)$ in (3), we can better understand the signal correlations. In this case, each of the outputs $\boldsymbol{y}_i(x)$, $i = 1, \ldots, p$, is a Gaussian process with kernel

$$k_{y_i}(x, x') = \sum_{j=1}^{q} W_{ij}(x)[k_{f_j}(x, x') + \sigma_f^2]W_{ij}(x') + \sigma_y^2 . \tag{4}$$

Even if σ_f^2 and σ_y^2 are zero, so that this is a noise free regression, there are still signal correlations; the components of $\boldsymbol{y}$ are coupled through the matrix $W(x)$. Once the network has been trained, $W(x)$ is conditioned on the data $\mathcal{D}$, and so the predictive covariances of $\boldsymbol{y}(x_*)|\mathcal{D}$ are now influenced by the values of the observations themselves, and not just distances between the test point x_* and the observed points $x_1, \ldots, x_N$ as is the case for independent GPs; we can view (4) as an adaptive kernel learned from the data. There are three other interesting features in equation (4): 1) the amplitude of the covariance function $\sum_{j=1}^{q} W_{ij}(x)W_{ij}(x')$ is non-stationary (input dependent); 2) even if each of the kernels k_{f_j} has different *stationary* length-scales, the mixture of the kernels k_{f_j} is input dependent and so the effective overall length-scale is non-stationary; 3) the kernels k_{f_j} may be entirely different: some may be periodic, others squared exponential, others Brownian motion, etc. . So the overall covariance function (kernel) may be continuously switching between regions of entirely different covariance structures.

[2] Coincidentally, there is an unrelated paper called "Gaussian process networks" (Friedman and Nachman, 2000), which is about learning the structure of Bayesian networks – e.g. the direction of dependence between random variables.

In addition to modelling signal correlations, we can see from equation (3) that the GPRN is simultaneously a multivariate volatility model. The noise covariance is $\sigma_f^2 W(x)W(x)^\top + \sigma_y^2 I$. Since the entries of $W(x)$ are GPs, this noise model is an example of a *generalised Wishart process* (Wilson and Ghahramani, 2010b, 2011).

The number of nodes q influences how the model accounts for signal and noise correlations. If q is smaller than p, the dimension of $\boldsymbol{y}(x)$, the model performs dimensionality reduction and matrix factorization as part of the regression on $\boldsymbol{y}(x)$ and $\text{cov}[\boldsymbol{y}(x)]$. However, we may want $q > p$, for instance if the output space were one dimensional ($p = 1$). In this case we would need $q > 1$ to realise features 2 and 3 listed above. For a given dataset, we can vary q and select the value which gives the highest marginal likelihood on training data. We can also use 'automatic relevance determination' (MacKay and Neal, 1994) as a proxy for model selection for q for a given dataset. This is achieved by introducing $\{a_j\}$, signal variances for each node function j, so that $k_{\hat{f}_j} \to a_j k_{\hat{f}_j}$, and comparing magnitudes of the trained a_j.

When $q = p = 1$, the GPRN essentially becomes the nonstationary GP regression model of Adams and Stegle (2008) and Turner and Sahani (2008). Likewise, when the weight functions are constants the GPRN becomes the Semiparametric Latent Factor Model (SLFM) of Teh et al. (2005), except that the resulting GP regression network is less prone to over-fitting through its use of full Bayesian inference.[3] Indeed, in an implementation of GPRN, one can switch features on or off; to switch off changing correlations and multivariate volatility, set $\sigma_f^2 = 0$ and the length-scales for weight function kernels (k_w) to large fixed values.

3 Inference

Now that we have specified a prior $p(\boldsymbol{y}(x))$ at all points x in our domain $\mathcal{X}$, we wish to predict $\mathbb{E}[\boldsymbol{y}(x_*)|x_*, \mathcal{D}]$ and $\text{cov}[\boldsymbol{y}(x_*)|x_*, \mathcal{D}]$ at a test input x_*, given vector valued data $\mathcal{D} = \{\boldsymbol{y}(x_i) : i = 1, \ldots, N\}$. We do this using two different approaches – variational Bayes and Markov chain Monte Carlo (MCMC) – and we compare between these approaches. We also use variational Bayes to estimate all hyperparameters $\boldsymbol{\gamma} = \{\boldsymbol{\theta}_f, \boldsymbol{\theta}_w, \sigma_f, \sigma_y\}$, where, as before, $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_w$ are the length-scales of the node and weight function kernels.

As a first step, we re-write the prior in terms of $\boldsymbol{u} = (\hat{\mathbf{f}}, \mathbf{W})$, a vector composed of all the node and weight Gaussian process functions, evaluated at the training points $\{x_1, \ldots, x_N\}$. There will be q node functions and $p \times q$ weight functions. Therefore

$$p(\boldsymbol{u}|\sigma_f, \boldsymbol{\theta}_f, \boldsymbol{\theta}_w) = \mathcal{N}(0, C_B)\,, \tag{5}$$

where C_B is an $Nq(p+1) \times Nq(p+1)$ block diagonal matrix, since the weight and node functions are independent in the prior. The way we have ordered $\boldsymbol{u}$, the first q blocks are $N \times N$ covariance matrices $K_{\hat{f}}$ from the node kernel $k_{\hat{f}}$, and the last blocks are $N \times N$ covariance matrices K_w from the weight kernel k_w.

Next we specify our likelihood function, so we can use Bayes' theorem to find the posterior $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$. From (1), our likelihood is

$$p(\mathcal{D}|\boldsymbol{u}, \sigma_y) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}(x_i); W(x_i)\hat{\boldsymbol{f}}(x_i), \sigma_y^2 I)\,. \tag{6}$$

By incorporating noise on $\boldsymbol{f}$, the GP network accounts for multivariate volatility (as in (3)), without the need for costly or numerically unstable matrix inversions. For other multivariate volatility models, like multivariate GARCH (Bollerslev et al., 1988), or multivariate stochastic volatility (Harvey et al., 1994), the likelihood takes the form $p(\mathcal{D}|\beta) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$, and requires inversions of $p \times p$ covariance matrices. There are three other notable advantages to the inference with GPRN: 1) it is easy to simultaneously estimate $\boldsymbol{\mu}_i$ and Σ_i. Usually in the multivariate volatility setting, $\boldsymbol{\mu}_i$ is assumed to be a constant; 2) we

[3] In Teh et al. (2005), the weight constants are a large matrix of hyperparameters, determined through maximising a marginal likelihood.

can use a Student-t observation model instead of a Gaussian observation model, by letting $\boldsymbol{z}$ in (1) be t distributed, with minimal changes to the inference procedures; 3) we can transform the components of the product $W(x_i)\hat{\boldsymbol{f}}(x_i)$ so that the priors on the components of $\boldsymbol{y}(x_i)$ become *copula processes* (Wilson and Ghahramani, 2010a) and have whatever marginals we desire. We can also do this without significantly changing inference procedures.

Now that we have specified our prior and likelihood, we can apply Bayes' theorem:

$$p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma}) \propto p(\mathcal{D}|\boldsymbol{u}, \sigma_y) p(\boldsymbol{u}|\boldsymbol{\theta}_f, \boldsymbol{\theta}_{\boldsymbol{w}}, \sigma_f)\,. \tag{7}$$

In the next sections, we discuss how to either sample from or use variational Bayes to approximate this posterior in (7), so that we can estimate $p(\boldsymbol{y}(x_*)|\mathcal{D})$. We also use variational Bayes to learn the hyperparameters $\boldsymbol{\gamma}$.

3.1 Markov chain Monte Carlo

To sample from (7), we could use a Gibbs sampling scheme which would have conjugate posterior updates, alternately conditioning on weight and node functions. However, this Gibbs cycle would mix poorly because of the tight correlations between the weights and the nodes. In general, MCMC samples from (7) mix poorly because of the strong correlations in the prior imposed by C_B. The sampling process is also often slowed by costly matrix inversions in the likelihood.

We use Elliptical Slice Sampling (Murray et al., 2010), a recent MCMC technique specifically designed to sample from posteriors with tightly correlated Gaussian priors. It does joint updates and has no free parameters. We find that it mixes well. And since there are no costly or numerically unstable matrix inversions in the likelihood of (6) we also find sampling to be highly efficient.

With a sample from (7), we can sample from the predictive $p(W(x_*), \boldsymbol{f}(x_*)|\boldsymbol{u}, \sigma_f, \mathcal{D})$. Let $W_*^i, \boldsymbol{f}_*^i$ be the i^{th} such joint sample. Using (3) we can then construct samples of $p(\boldsymbol{y}(x_*)|W_*^i, \boldsymbol{f}_*^i, \sigma_f, \sigma_y)$, from which we can construct the predictive distribution

$$p(\boldsymbol{y}(x_*)|\mathcal{D}) = \lim_{J\to\infty} \frac{1}{J} \sum_{i=1}^{J} p(\boldsymbol{y}(x_*)|W_*^i, \boldsymbol{f}_*^i, \sigma_f, \sigma_y)\,. \tag{8}$$

We see that even with a Gaussian observation model, the predictive distribution in (8) is an infinite mixture of Gaussians, and will generally be heavy tailed and therefore robust to outliers.

Mixing was assessed by looking at trace plots of samples, and the likelihoods of these samples. Specific information about how long it takes to sample a solution for a given problem is in the experiments section.

3.2 Variational Bayes

We perform variational EM (Jordan et al., 1999) to fit an approximate posterior q to the true posterior p, by minimising the Kullback-Leibler divergence $KL(q||p) = -H[q(\mathbf{v})] - \int q(\mathbf{v}) \log p(\mathbf{v}) d\mathbf{v}$, where $H[q(\mathbf{v})] = -\int q(\mathbf{v}) \log q(\mathbf{v}) d\mathbf{v}$ is the entropy and $\mathbf{v} = \{\mathbf{f}, \mathbf{W}, \sigma_f^2, \sigma_y^2, a_j\}$.

E-step. We use Variational Message Passing (Winn and Bishop, 2006) under the Infer.NET framework (Minka et al., 2010) to estimate the posterior over $\mathbf{v} = \{\mathbf{f}, \mathbf{W}, \sigma_f^2, \sigma_y^2, a_j\}$. We specify inverse Gamma priors on $\{\sigma_f^2, \sigma_y^2, a_j\}$:

$$\sigma_{fj}^2 \sim \text{IG}(\alpha_{\sigma_f^2}, \beta_{\sigma_f^2}),\ \sigma_y^2 \sim \text{IG}(\alpha_{\sigma_y^2}, \beta_{\sigma_y^2}),\ a_j \sim \text{IG}(\alpha_a, \beta_a).$$

For mathematical and computational convenience we introduce the following variables which are deterministic functions of the existing variables in the model:

$$w_{nij} := W_{ij}(x_n), \qquad f'_{nj} := f_j(x_n) \tag{9}$$

$$t_{nij} := w_{nij} \hat{f}_{nj}, \qquad s_{in} := \sum_j t_{nij} \tag{10}$$

Note that the observations $y_i(x_n) \sim \mathcal{N}(s_{in}, \sigma_y^2)$ and that $\hat{f}_{nj} \sim \mathcal{N}(f'_{nj}, \sigma^2_{f_j})$. Variational message passing uses these deterministic factors and the associated "pseudo-marginals" as conduits to pass appropriate moments, resulting in the same updates as standard VB (Winn and Bishop, 2006). The full model can now be written as

$$
\begin{aligned}
p(\mathbf{v}) \propto \mathrm{IG}(\sigma_y^2; \alpha_{\sigma_y^2}, \beta_{\sigma_y^2}) \prod_{j=1}^{Q} \Bigg(& \mathcal{N}(\mathbf{f}_j; 0, a_j K_{f_j}) \\
\mathrm{IG}(\sigma^2_{fj}; \alpha_{\sigma_f^2}, \beta_{\sigma_f^2}) \mathrm{IG}(a_j; \alpha_a, \beta_a) \prod_{i=1}^{P} \Bigg[& \mathcal{N}(\mathbf{W}_{ij}; 0, K_w) \\
\prod_{n=1}^{N} \delta(w_{nij} - W_{ij}(x_n)) \delta(f'_{nj} - \hat{f}_j(x_n)) \mathcal{N}(\hat{f}_{nj}; f'_{nj}, \sigma^2_{f_j}) & \\
\delta(t_{nij} - w_{nij} \hat{f}_{nj}) \delta(s_{in} - \sum_j t_{nij}) \mathcal{N}(y_i(x_n); s_{in}, \sigma_y^2) \Bigg] \Bigg) &
\end{aligned}
$$

We use a variational posterior of the following form:

$$
\begin{aligned}
q(\mathbf{v}) = q_{\sigma_y^2}(\sigma_y^2) \prod_{j=1}^{Q} q_{\mathbf{f}_j}(\mathbf{f}_j) q_{\sigma^2_{fj}}(\sigma^2_{fj}) q_{a_j}(a_j) \prod_{i=1}^{P} q_{\mathbf{W}_{ij}}(\mathbf{W}_{ij}) & \\
\prod_{n=1}^{N} q_{w_{nij}}(w_{nij}) q_{f'_{nj}}(f'_{nj}) q_{\hat{f}_{nj}}(\hat{f}_{nj}) q_{t_{nij}}(t_{nij}) q_{s_{in}}(s_{in}) &
\end{aligned}
$$

where $q_{\sigma_y^2}, q_{\sigma^2_{fj}}$ and q_{a_j} are inverse Gamma distributions; $q_{w_{nij}}, q_{f'_{nj}}, q_{\hat{f}_{nj}}, q_{t_{nij}}$ and $q_{s_{in}}$ are univariate normal distributions; and $q_{\mathbf{f}_j}$ and $q_{\mathbf{W}_{ij}}(\mathbf{W}_{ij})$ are multivariate normal distributions.

The updates for $\mathbf{f}, \mathbf{W}, \sigma_f^2, \sigma_y^2$ are standard VB updates and are available in Infer.NET. The update for the ARD parameters a_j however required specific implementation. The factor itself is

$$
\begin{aligned}
\log \mathcal{N}(\mathbf{f}_j; \mathbf{0}, a_j K_f) &\stackrel{c}{=} -\frac{1}{2} \log |a_j K_j| - \frac{1}{2} \mathbf{f}_j^T (a_j K_f)^{-1} \mathbf{f}_j \\
&= -\frac{N}{2} \log a_j - \frac{1}{2} \log |K_j| - \frac{1}{2} a_j^{-1} \mathbf{f}_j^T K_f^{-1} \mathbf{f}_j \qquad (11)
\end{aligned}
$$

where $\stackrel{c}{=}$ denotes equality up to an additive constant. Taking expectations with respect to $\mathbf{f}$ under q we obtain the VMP message to a_j as being $\mathrm{IG}\left(a_j; \frac{N}{2} - 1, \frac{1}{2}\langle \mathbf{f}_j^T K_f^{-1} \mathbf{f}_j \rangle\right)$. Since the variational posterior on $\mathbf{f}$ is multivariate normal the expectation $\langle \mathbf{f}_j^T K_f^{-1} \mathbf{f}_j \rangle$ is straightforward to calculate.

M-step. In the M-step we optimise the variational lower bound with respect to the log length scale parameters $\{\theta_f, \theta_w\}$, using gradient descent with line search. When optimising θ_f we only need to consider the contribution to the lower bound of the factor $\mathcal{N}(\mathbf{f}_j; 0, a_j K_{f_j})$ (see (11)), which is straightforward to evaluate and differentiate (see Appendix). For θ_w we consider the contribution of $\mathcal{N}(\mathbf{W}_{pq}; 0, K_W)$.

3.3 Computational Considerations

GPRN is mainly limited by taking the Cholesky decomposition of the block diagonal C_B, an $Nq(p+1) \times Nq(p+1)$ matrix. But pq of these blocks are the same $N \times N$ covariance matrix K_w for the weight functions, and q of these blocks are the covariance matrices $K_{\hat{f}_i}$ associated with the node functions, and $\mathrm{chol}(\mathrm{blkdiag}(A, B, \dots)) = \mathrm{blkdiag}(\mathrm{chol}(A), \mathrm{chol}(B), \dots)$. Therefore assuming the node functions share the same covariance function (which they do in our experiments), the complexity of this operation is only $\mathcal{O}(N^3)$, the same as for regular Gaussian process regression. At worst it is $\mathcal{O}(qN^3)$, assuming different covariance functions for each node.

Sampling also requires likelihood evaluations. Since there are input dependent noise correlations between the elements of the p dimensional observations $\boldsymbol{y}(x_i)$, multivariate volatility models would normally require inverting a $p \times p$ covariance matrix N times, like MGARCH (Bollerslev et al., 1988) or multivariate stochastic volatility models (Harvey et al., 1994). This would lead to a total complexity of $\mathcal{O}(Nqp + Np^3)$. However, by working directly with the noisy $\hat{\boldsymbol{f}}$ instead of the noise free $\boldsymbol{f}$, evaluating the likelihood requires no costly or numerically unstable inversions, and thus has a complexity of only $\mathcal{O}(Nqp)$. This allows GPRN to scale to high dimensions; indeed we have a 1000 dimensional gene expression experiment in Section 4.

The computational complexity of VB is dominated by the $\mathcal{O}(N^3)$ inversions required to calculate the covariance of the node and weight functions in the E-step. Naively q and qp such inversions are required per iteration for the node and weight functions respectively, giving a total complexity of $\mathcal{O}(qpN^3)$. However, under VB the covariances of the weight functions for the same p are all equal, reducing the complexity to $\mathcal{O}(qN^3)$. If p is large the $\mathcal{O}(pqN^2)$ cost of calculating the weight function means may become significant. Although the per iteration cost of VB is actually higher than for MCMC far fewer iterations are typically required to reach convergence.

Overall, even though the GPRN accounts for input dependent signal correlations (rather than fixing the correlations like other multi-task methods), the computational demands of GPRN compare favourably to most multi-task GP models, which commonly have a complexity of $\mathcal{O}(p^3N^3)$ (Alvarez and Lawrence, 2011).

4 Experiments

We compare the GPRN to multi-task learning and multivariate volatility models, and we also use the GPRN to gain new scientific insights into the data we model. Furthermore, we compare between variational Bayes (VB) and Markov chain Monte Carlo (MCMC) inference within the GPRN framework. To keep our comparisons up to date, we *exactly* reproduce many of the experiments in recent papers by Alvarez and Lawrence (2011) and Wilson and Ghahramani (2010b) on benchmark datasets. In the multi-task setting, there are p dimensional observations $\boldsymbol{y}(x)$, and the goal is to use the correlations between the elements of $\boldsymbol{y}(x)$ to make better predictions of $\boldsymbol{y}(x_*)$, for a test input x_*, than if we were to treat the dimensions independently. A major difference between GPRN and alternative multi-task models is that the GPRN accounts for signal correlations that *change* with x, rather than fixed correlations. It also accounts for changing noise correlations (multivariate volatility).

We compare to the following multi-task GP methods: 1) the linear model of coregionalisation (LMC) (Journel and Huijbregts, 1978; Goovaerts, 1997), 2) the intrinsic coregionalisation model (ICM) (Goovaerts, 1997), 3) ordinary co-kriging (Cressie, 1993; Goovaerts, 1997; Wackernagel, 2003), 4) the semiparametric latent factor model (SLFM) (Teh et al., 2005), 5) convolved multiple output Gaussian processes (CMOGP) (Barry and Jay, 1996; Ver Hoef and Barry, 1998; Boyle and Frean, 2004), 6) standard independent Gaussian processes (GP), 7) and the DTC (Csató and Opper, 2001; Seeger et al., 2003; Quiñonero-Candela and Rasmussen, 2005; Rasmussen and Williams, 2006) , 8) FITC (Snelson and Ghahramani, 2006), and 9) PITC (Quiñonero-Candela and Rasmussen, 2005) sparse approximations for CMOGP (Alvarez and Lawrence, 2011), which we respectively label as MDTC, MFITC and MPITC. Detail about each of these methods is in Alvarez and Lawrence (2011). We compare on a 3 dimensional geostatistics heavy metal dataset from the Swiss Jura, where 28% of the observations for one of the outputs (response variables) is missing, and on gene expression datasets with 50 and 1000 dimensional time dependent outputs $\boldsymbol{y}(t)$.

In the multi-task experiments, the GPRN accounts for input dependent noise covariance matrices $\text{cov}[\boldsymbol{y}(x)] = \Sigma(x)$. To specifically test GPRN's ability to model input dependent noise covariances (multivariate volatility), we also compare predictions of $\Sigma(x)$ to those made by popular multivariate volatility models – full BEKK MGARCH (Engle and Kroner, 1995), generalised Wishart processes (Wilson and Ghahramani, 2010b), the original Wishart process (Bru, 1991; Gouriéroux et al., 2009), and empirical estimates – on benchmark return series datasets which are especially suited to MGARCH (Poon and Granger, 2005; Hansen and Lunde, 2005; Brownlees et al., 2009; McCullough and Renfro, 1998; Brooks et al., 2001).

In all experiments, GPRN uses a squared exponential covariance function for its node functions, and another squared exponential covariance function for its weight functions.

4.1 Gene Expression

Tomancak et al. (2002) measured gene expression levels every hour for 12 hours during Drosophila embryogenesis; they then repeated this experiment for an independent replica (a second independent time series). Gene expression is activated and deactivated by transcription factor proteins. We focus on genes which are thought to at least be regulated by the transcription factor `twi`, which influences mesoderm and muscle development in Drosophila (Zinzen et al., 2009). The assumption is that these gene expression levels are all correlated. We would like to use how these correlations change over time to make better predictions of time varying gene expression in the presence of transcription factors. In total there are 1621 genes (outputs) at $N = 12$ time points (inputs), on two independent replicas. For training, $p = 50$ random genes were selected from the first replica, and the corresponding 50 genes in the second replica were used for testing. We then repeated this experiment 10 times with a different set of genes each time, and averaged the results. We then repeated the whole experiment, but with $p = 1000$ genes. We used *exactly* the same training and testing sets as Alvarez and Lawrence (2011).

We use a relatively small $p = 50$ dataset so that we are able to compare with popular alternative multi-task methods (LMC, CMOGP, SLFM) which have a complexity of $\mathcal{O}(N^3p^3)$ and would not scale to $p = 1000$ (Alvarez and Lawrence, 2011). For $p = 1000$, we compare to the sparse convolved multiple output GP methods (MFITC, MDTC, and MPITC) of Alvarez and Lawrence (2011). In both of these regressions, the GPRN is accounting for multivariate volatility; this is the first time a multivariate stochastic volatility model has been estimated for $p > 50$ (Chib et al., 2006). We assess performance using standardised mean square error (SMSE) and mean standardized log loss (MSLL), as defined in Rasmussen and Williams (2006) on page 23, and discussed in Alvarez and Lawrence (2011) on page 1469. Using the empirical mean and variance to fit the data would give an SMSE and MSLL of 1 and 0 respectively. The smaller the SMSE and more negative the MSLL the better.

The results are in Table 1, under the headings `GENE` (50D) and `GENE` (1000D). For `SET 1` of the 50D dataset, we used Replica 1 and Replica 2 in Alvarez and Lawrence (2011) respectively as training and testing replicas. We follow Alvarez and Lawrence (2011) and reverse training and testing replicas to create `SET 2`. The results for LMC, CMOGP, MFITC, MPITC, and MDTC are reproduced from Alvarez and Lawrence (2011). GPRN significantly outperforms all of the other models, with between 46% and 68% of the SMSE, and similarly strong results on the MSLL error metric. On the 50D dataset the MCMC and VB results are comparable. However, on the 1000D dataset GPRN with VB noticeably outperforms GPRN with MCMC, likely because MCMC is not mixing as well in high dimensions. Indeed VB may have an advantage over MCMC in high dimensions. On the other hand, GPRN with MCMC is still robust, outperforming all the other methods on the 1000 dimensional dataset.

On both the 50 and 1000 dimensional datasets, the marginal likelihood for the network structure is sharply peaked at $q = 1$. This is evidence for the hypothesis that there is only the one transcription factor `twi` controlling the expression levels of the genes in question. These datasets can be found in Neil Lawrence's GPSIM toolbox: `http://staffwww.dcs.shef.ac.uk/people/N.Lawrence/gpsim/`

Typical GPRN (VB) runtimes for the 50D and 1000D datasets were respectively 12 seconds and 330 seconds.

4.2 Jura Geostatistics

Here we are interested in predicting concentrations of cadmium at 100 locations within a 14.5 km^2 region of the Swiss Jura. For training, we have access to measurements of cadmium at 259 neighbouring locations. We also have access to nickel and zinc concentrations at these 259 locations, as well as at the 100 locations we wish to predict cadmium. While a standard Gaussian process regression model would only be able to make use of the cadmium training measurements, a multi-task method can use the correlated nickel and zinc measurements to enhance predictions[4]. With GPRN we can also make use of how the correlations between nickel, zinc, and cadmium change with location to further enhance predictions.

Here the network structure with by far the highest marginal likelihood has $q = 2$ latent node functions. The node and weight functions learnt using VB for this setting are shown in Figure 2. Since there are

[4]This can be seen as a multivariate missing data problem, with $p = 3$ outputs.

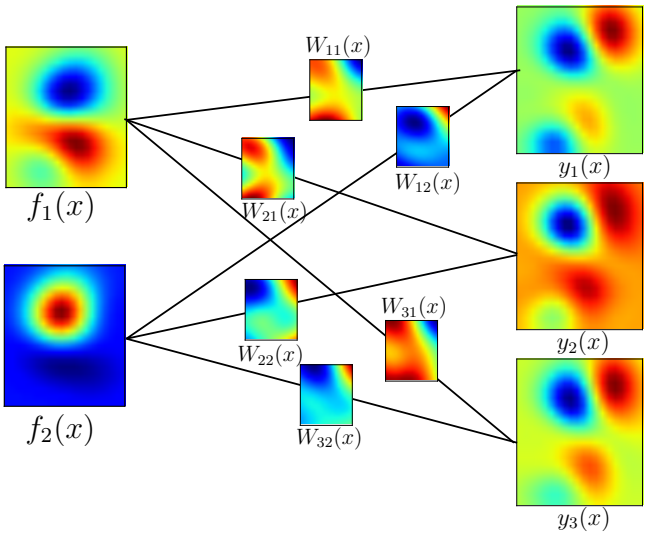

Figure 2: Network structure for the Jura dataset learnt by GPRN. The layout here is as in Figure 1, with the spatially varying node and weight functions shown, along with the predictive means for the observations. The three output dimensions are cadmium, nickel and zinc concentrations respectively.

$p = 3$ output dimensions, the result $q < p$ suggests that heavy metal concentrations in the Swiss Jura are correlated. Indeed, using our model we can observe the *spatially varying* correlations between heavy metal concentrations, as shown for cadmium and zinc in Figure 3. Although the correlation between cadmium and zinc is generally positive (with values around 0.6), there is a region where the correlations drop of noticeably, perhaps corresponding to a geological structure. The quantitative results in Table 1 confirm that the ability of GPRN to learn these spatially varying correlations is beneficial in terms of being able to predict cadmium concentrations.

We assess performance quantitatively using mean absolute error (MAE) between the predicted and true cadmium concentrations. We restart the experiment 10 times with different initialisations of the parameters, and average the MAE. The results are marked by `JURA` in Table 1. This experiment follows Goovaerts (1997) and Alvarez and Lawrence (2011). The results for SLFM, ICM and CMOGP are from Alvarez and Lawrence (2011), and the results for co-kriging are from Goovaerts (1997). It is unclear what preprocessing was performed for these methods, but we found log transforming and normalising each dimension to have zero mean and unit variance to be beneficial due to the skewed distribution of the y-values (but we also include results on untransformed data, marked with *). All of the multiple output methods give lower MAE than using an independent GP, and GPRN outperforms SLFM and the other multiple output methods.

For the `JURA` dataset, the improved performance of GPRN is at the cost of a slightly greater runtime. However, GPRN is accounting for input dependent signal and noise correlations, unlike the other methods. Moreover, the complexity of GPRN scales with p as $\mathcal{O}(Nqp)$, unlike the other methods which scale as $\mathcal{O}(N^3p^3)$ (Alvarez and Lawrence, 2011). This is why GPRN runs relatively quickly on the 1000 dimensional gene expression dataset, for which the other methods are intractable. This data is available from `http://www.ai-geostats.org/`.

4.3 Multivariate Volatility

In the previous experiments the GPRN implicitly accounted for multivariate volatility (input dependent noise covariance) in making predictions of $\boldsymbol{y}(x_*)$. The GPRN incorporates a generalised Wishart process (Wilson and Ghahramani, 2010b, 2011) noise model into a more general model which can also account for signal correlations and other nonstationarities. Although our focus is the ability of GPRN to model input dependent correlations in a multiple output regression setting, we here test the GPRN explicitly as

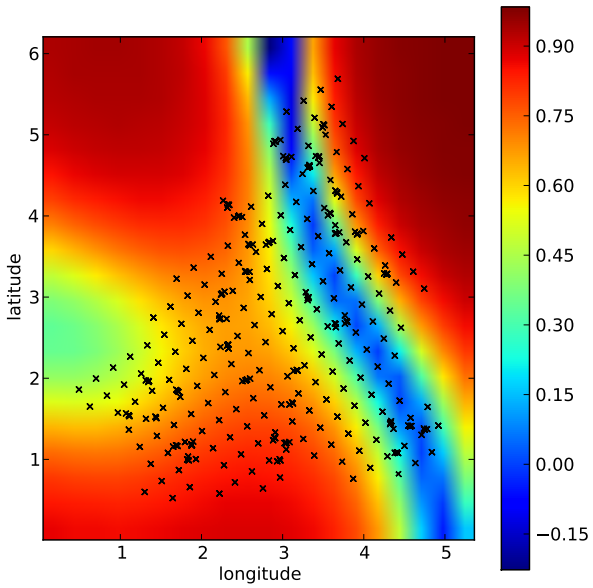

Figure 3: Spatially dependent correlation between cadmium and zinc learnt by the GPRN. Markers show the locations where measurements were made.

a model of multivariate volatility, and assess predictions of $\Sigma(t) = \mathrm{cov}[\boldsymbol{y}(t)]$, where the observations $\boldsymbol{y}$ are time dependent. We make historical predictions at observed time points, and one day ahead forecasts. Historical predictions can be used, for example, to understand a past financial crisis. We follow Wilson and Ghahramani (2010b) exactly, and predict $\Sigma(t)$ for returns on three currency exchanges (`EXCHANGE`) and five equity indices (`EQUITY`) processed exactly as in Wilson and Ghahramani (2010b). These datasets are especially suited to MGARCH, the most popular multivariate volatility model, and have become a benchmark for assessing GARCH models (Poon and Granger, 2005; Hansen and Lunde, 2005; Brownlees et al., 2009; McCullough and Renfro, 1998; Brooks et al., 2001). We make 200 historical predictions of $\Sigma(x)$ and 200 one step ahead forecasts. The forecasts are assessed using the log likelihood of the new observations under the predicted covariance, denoted $\mathcal{L}$ Forecast. We compare to full BEKK MGARCH (Engle and Kroner, 1995), the generalised Wishart process (Wilson and Ghahramani, 2010b), the original Wishart process (Bru, 1991; Gouriéroux et al., 2009), and using the empirical covariance of the training set. We see in Table 1 that GPRN (VB) is competitive with MGARCH, even though these datasets are particularly suited to MGARCH. The Historical MSE for `EXCHANGE` is between the learnt covariance $\Sigma(x)$ and $\mathbf{y}(x)\mathbf{y}(x)^\top$, so the high MSE values for GPRN on `EXCHANGE` are essentially training error, and are less meaningful than the encouraging step ahead forecast likelihoods. The historical predictions are more relevant in `EQUITY`, where we can compare to the true covariances. See Wilson and Ghahramani (2010b) for details.

GPRN and GWP are both highly flexible but fully Bayesian models for multivariate volatility, so it understandable that their performance is comparable. While GPRN (MCMC) sometimes outperforms MGARCH, the GWP, and the WP, it is often outperformed by GPRN (VB) on the multivariate volatility, perhaps suggesting convergence problems. These data were obtained using Bloomberg (`http://www.bloomberg.com/`).

Table 1: Comparative performance on all datasets.

GENE (50D)	Average SMSE	Average MSLL
SET 1:		
GPRN (VB)	0.3356 ± 0.0294	$\mathbf{-0.5945 \pm 0.0536}$
GPRN (MCMC)	$\mathbf{0.3236 \pm 0.0311}$	-0.5523 ± 0.0478
LMC	0.6909 ± 0.0294	-0.2687 ± 0.0594
CMOGP	0.4859 ± 0.0387	-0.3617 ± 0.0511
SLFM	0.6435 ± 0.0657	-0.2376 ± 0.0456
SET 2:		
GPRN (VB)	0.3403 ± 0.0339	$\mathbf{-0.6142 \pm 0.0557}$
GPRN (MCMC)	$\mathbf{0.3266 \pm 0.0321}$	-0.5683 ± 0.0542
LMC	0.6194 ± 0.0447	-0.2360 ± 0.0696
CMOGP	0.4615 ± 0.0626	-0.3811 ± 0.0748
SLFM	0.6264 ± 0.0610	-0.2528 ± 0.0453
GENE (1000D)	Average SMSE	Average MSLL
GPRN (VB)	$\mathbf{0.3473 \pm 0.0062}$	$\mathbf{-0.6209 \pm 0.0085}$
GPRN (MCMC)	0.4520 ± 0.0079	-0.4712 ± 0.0327
MFITC	0.5469 ± 0.0125	-0.3124 ± 0.0200
MPITC	0.5537 ± 0.0136	-0.3162 ± 0.0206
MDTC	0.5421 ± 0.0085	-0.2493 ± 0.0183
JURA	Average MAE	Training Time (secs)
GPRN (VB)	$\mathbf{0.4040 \pm 0.0006}$	3781
GPRN* (VB)	0.4525 ± 0.0036	4560
SLFM (VB)	0.4247 ± 0.0004	1643
SLFM* (VB)	0.4679 ± 0.0030	1850
SLFM	0.4578 ± 0.0025	792
Co-kriging	0.51	
ICM	0.4608 ± 0.0025	507
CMOGP	0.4552 ± 0.0013	784
GP	0.5739 ± 0.0003	74
EXCHANGE	Historical MSE	$\mathcal{L}$ Forecast
GPRN (VB)	3.83×10^{-8}	**2073**
GPRN (MCMC)	6.120×10^{-9}	2012
GWP	$\mathbf{3.88 \times 10^{-9}}$	2020
WP	$\mathbf{3.88 \times 10^{-9}}$	1950
MGARCH	3.96×10^{-9}	2050
Empirical	4.14×10^{-9}	2006
EQUITY	Historical MSE	$\mathcal{L}$ Forecast
GPRN (VB)	0.978×10^{-9}	2740
GPRN (MCMC)	$\mathbf{0.827 \times 10^{-9}}$	2630
GWP	2.80×10^{-9}	**2930**
WP	3.96×10^{-9}	1710
MGARCH	6.69×10^{-9}	2760
Empirical	7.57×10^{-9}	2370

5 Discussion

A Gaussian process regression network (GPRN) has a simple and interpretable structure, and generalises many of the recent extensions to the Gaussian process regression framework. The model naturally accommodates input dependent signal and noise correlations between multiple output variables, heavy tailed predictive distributions, input dependent length-scales and amplitudes, and adaptive covariance functions. Furthermore, GPRN has scalable inference procedures, and strong empirical performance on several benchmark datasets.

In the future, it would be enlightening to use GPRN with different types of adaptive covariance structures, particularly in the case where $p = 1$ and $q > 1$; in one dimensional output space it would be easy, for instance, to visualise a process gradually switching between brownian motion, periodic, and smooth covariance functions. It would also be interesting to apply this adaptive network to classification. We hope the GPRN will inspire further research into adaptive networks, and further connections between different areas of machine learning and statistics.

6 Acknowledgements

Thanks to Mauricio Álvarez and Neil Lawrence for their valuable feedback about the gene expression datasets.

7 Appendix

7.1 Gaussian processes

Since we extend the Gaussian process framework, we briefly review Gaussian process regression, some notation, and expand on some of the points in the introduction. For more detail see Rasmussen and Williams (2006).

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. Using a Gaussian process, we can define a distribution over functions $w(x)$:

$$w(x) \sim \mathcal{GP}(m(x), k(x, x')), \tag{12}$$

where w is the output variable, x is an arbitrary (potentially vector valued) input variable, and the mean $m(x)$ and covariance function (or kernel) $k(x, x')$ are respectively defined as

$$m(x) = \mathbb{E}[w(x)]\,, \tag{13}$$

$$k(x, x') = \mathrm{cov}[w(x), w(x')]\,. \tag{14}$$

This means that any collection of function values has a joint Gaussian distribution:

$$(w(x_1), w(x_2), \ldots, w(x_N))^\top \sim \mathcal{N}(\boldsymbol{\mu}, K)\,, \tag{15}$$

where the $N \times N$ covariance matrix K has entries $K_{ij} = k(x_i, x_j)$, and the mean $\boldsymbol{\mu}$ has entries $\boldsymbol{\mu}_i = m(x_i)$. The properties of these functions (smoothness, periodicity, etc.) are determined by the kernel function. The squared exponential kernel is popular:

$$k_{\mathrm{SE}}(x, x') = A \exp(-0.5||x - x'||^2/l^2)\,. \tag{16}$$

Functions drawn from a Gaussian process with this kernel function are smooth, and can display long range trends. The length-scale *hyperparameter* l is easy to interpret: it determines how much the function values $w(x)$ and $w(x + a)$ depend on one another, for some constant $a \in \mathcal{X}$. When the length-scale is learned from data, it is useful for determining how far into the past one should look in order to make good forecasts. $A \in \mathbb{R}$ is the *amplitude* coefficient, which determines the marginal variance of $w(x)$ in the prior, $\mathrm{Var}[w(x)] = A$, and the magnitude of covariances between $w(x)$ at different inputs x.

The Ornstein-Uhlenbeck kernel is also widely applied:

$$k_{\mathrm{OU}}(x, x') = \exp(-||x - x'||/l)\,. \tag{17}$$

In one dimension it is the covariance function of an Ornstein-Uhlenbeck process (Uhlenback and Ornstein, 1930), which was introduced to model the velocity of a particle undergoing Brownian motion. With this kernel, the corresponding GP is a continuous time AR(1) process with Markovian dynamics: $w(x + a)$ is independent of $w(x - a)$ given $w(x)$ for any constant a. Indeed the OU kernel belongs to a more general class of Matérn kernels,

$$k_{\text{Matérn}}(x, x') = \frac{2^{1-\alpha}}{\Gamma(\alpha)}(\frac{\sqrt{2\alpha}||x - x'||}{l})^{\alpha} K_{\alpha}(\frac{\sqrt{2\alpha}||x - x'||}{l})\,, \tag{18}$$

where K_α is a modified Bessel function (Abramowitz and Stegun, 1964). In one dimension the corresponding GP is a continuous time AR(p) process, where $p = \alpha + 1/2$.[5] The OU kernel is recovered by setting $\alpha = 1/2$.

There are many other useful kernels, like the periodic kernel (with a period that can be learned from data), or the Gibbs kernel (Gibbs, 1997) which allows for input dependent length-scales. Kernels can be combined together, e.g. $k = a_1 k_1 + a_2 k_2 + a_3 k_3$, and the relative importance of each kernel can be determined from data (e.g. from estimating a_1, a_2, a_3). Rasmussen and Williams (2006) and Bishop (2006) have a discussion about how to create and combine kernels.

Suppose we are doing a regression using points $\{y(x_1), \ldots, y(x_N)\}$ from a noisy function $y = w(x) + \epsilon$, where ϵ is additive i.i.d Gaussian noise, such that $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Letting $\boldsymbol{y} = (y(x_1), \ldots, y(x_N))^\top$, and $\boldsymbol{w} = (w(x_1), \ldots, w(x_N)^\top$, we have $p(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}, \sigma_n^2 I)$ and $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}, K)$ as above. For notational simplicity, we assume $\boldsymbol{\mu} = 0$. For a test point $w(x_*)$, the joint distribution $p(w(x_*), \boldsymbol{y})$ is Gaussian:

$$\begin{bmatrix} w(x_*) \\ \boldsymbol{w} \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \begin{bmatrix} k(x_*, x_*) & \boldsymbol{k}_*^\top \\ \boldsymbol{k}_* & K + \sigma_n^2 I \end{bmatrix})\,, \tag{19}$$

where K is defined as above, and $(\boldsymbol{k}_*)_i = k(x_*, x_i)$ with $i = 1, \ldots, N$. We can therefore condition on $\boldsymbol{y}$ to find $p(w(x_*)|\boldsymbol{y}) = \mathcal{N}(\mu_*, v_*)$ where

$$\mu_* = \boldsymbol{k}_*^\top (K + \sigma_n^2 I)^{-1} \boldsymbol{y}\,, \tag{20}$$

$$v_* = k(x_*, x_*) - \boldsymbol{k}_*^\top (K + \sigma_n^2 I)^{-1} \boldsymbol{k}_*\,. \tag{21}$$

We can find this more laboriously by noting that $p(\boldsymbol{w}|\boldsymbol{y})$ and $p(w(x_*)|\boldsymbol{w})$ are Gaussian and integrating, since $p(w(x_*)|\boldsymbol{y}) = \int p(w(x_*)|\boldsymbol{w}) p(\boldsymbol{w}|\boldsymbol{y}) d\boldsymbol{w}$.

We see that (21) doesn't depend on the data $\boldsymbol{y}$, just on how far away the test point x_* is from the training inputs $\{x_1, \ldots, x_N\}$.

In regards to the introduction, we also see that for this standard Gaussian process regression, the observation model $p(y|w)$ is Gaussian, the predictive distribution in (20) and (21) is Gaussian, the marginals in the prior (from marginalising equation (15)) are Gaussian, the noise is constant, and in the popular covariance functions given, the amplitude and length-scale are constant. A brief discussion of multiple outputs, noise models with dependencies, and non-Gaussian observation models can be found in sections 9.1, 9.2 and 9.3 on pages 190-191 of Rasmussen and Williams (2006), available free online at the book website `www.gaussianprocess.org/gpml`. An example of an input dependent length-scale is in section 4.2 on page 43.

7.2 Constraining W

It is possible to reduce the number of modes in the posterior by somehow constraining the weights W to be positive. For MCMC it is straightforward to do this by exponentiating the weights, as in Adams

[5] Discrete time autoregressive processes such as $w(t+1) = w(t) + \epsilon(t)$, where $\epsilon(t) \sim \mathcal{N}(0, 1)$, are widely used in time series modelling and are a particularly simple special case of Gaussian processes.

and Stegle (2008) and Adams et al. (2010). For VB it is more straightforward to explicitly constrain the weights to be positive using a truncated Gaussian representation. We found that these extensions did not significantly improve empirical performance, although exponentiating the weights sometimes improved numerical stability for MCMC on the multivariate volatility experiments. For Adams and Stegle (2008) exponentiating the weights will have been more valuable because they use Expectation Propagation which is known to perform badly in the presence of multimodality. MCMC and VB approaches are more robust to this problem.

7.3 VB M-step

From (11) we have:

$$\begin{aligned}&\langle \log \mathcal{N}(\mathbf{f}_j; 0, a_j K_{f_j})\rangle_q \\ &\stackrel{c}{=} -\frac{N}{2}\log a_j - \frac{1}{2}\log|K_{f_j}| - \frac{1}{2}\langle a_j^{-1}\rangle\langle \mathbf{f}_j^T K_{f_j}^{-1}\mathbf{f}_j\rangle\end{aligned}$$

We will need the gradient with respect to θ_f:

$$\begin{aligned}&\frac{\partial\langle \log \mathcal{N}(\mathbf{f}_j; 0, a_j K_{f_j})\rangle}{\partial\theta_f} \\ &= -\frac{1}{2}\mathrm{tr}\left(K_{f_j}^{-1}\frac{\partial K_{f_j}}{\partial\theta_f}\right) - \frac{1}{2}\langle a_j^{-1}\rangle\langle \mathbf{f}_j^T K_{f_j}^{-1}\frac{\partial K_{f_j}}{\partial\theta_f}K_{f_j}^{-1}\mathbf{f}_j\rangle\end{aligned}$$

The expectations here are straightforward to compute analytically.

7.4 VB predictive distributions

The predictive distribution is calculated as

$$p(\mathbf{y}^*(x)|\mathcal{D}) = \int p(\mathbf{y}^*(x)|W(x), f(x))p(W(x), f(x)|\mathcal{D})dWdf \tag{22}$$

VB fits the approximation $p(W(x), f(x)|\mathcal{D}) = q(W)q(f)$, so the approximate predictive is

$$p(\mathbf{y}^*(x)|\mathcal{D}) = \int p(\mathbf{y}^*(x)|W(x), f(x))q(W)q(f)dWdf \tag{23}$$

We can calculate the mean and covariance of this distribution analytically:

$$\bar{\mathbf{y}}^*(x)_i = \sum_k \mathbb{E}(W_{ik})\mathbb{E}[f_k] \tag{24}$$

$$\mathrm{cov}(\mathbf{y}^*(x))_{ij} = \sum_k [\mathbb{E}(W_{ik})\mathbb{E}(W_{jk})\mathrm{var}(f_k) + \delta_{ij}(\mathrm{var}(W_{ik})\mathbb{E}(f_k^2))] + \delta_{ij}\sigma_y^2 \tag{25}$$

It is also of interest to calculate the *noise* covariance. Recall our model can be written as

$$\boldsymbol{y}(x) = \underbrace{W(x)\boldsymbol{f}(x)}_{\text{signal}} + \underbrace{\sigma_f W(x)\boldsymbol{\epsilon} + \sigma_y \boldsymbol{z}}_{\text{noise}} \tag{26}$$

Let $\boldsymbol{n} = \sigma_f W(x)\boldsymbol{\epsilon} + \sigma_y \boldsymbol{z}$ be the noise. The covariance of $\mathbf{n}$ is then

$$\mathrm{cov}(\boldsymbol{n})_{ij} = \sum_k [\mathbb{E}[\sigma_{f_k}^2]\mathbb{E}(W_{ik})\mathbb{E}(W_{jk}) + \delta_{ij}\mathrm{var}(W_{jk})] + \delta_{ij}\sigma_y^2 \tag{27}$$

References

Abramowitz, M. and Stegun, I. (1964). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover publications.

Adams, R. and Stegle, O. (2008). Gaussian process product models for nonparametric nonstationarity. In *Proceedings of the 25th international conference on Machine learning*. ACM.

Adams, R. P., Dahl, G. E., and Murray, I. (2010). Incorporating side information into probabilistic matrix factorization using Gaussian processes. In Grünwald, P. and Spirtes, P., editors, *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 1–9.

Alvarez, M. and Lawrence, N. (2008). Sparse convolved Gaussian processes for multi-output regression. In *NIPS*.

Alvarez, M. and Lawrence, N. (2011). Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, 12:1425–1466.

Barry, R. and Jay, M. (1996). Blackbox kriging: spatial prediction without specifying variogram models. *Journal of Agricultural, Biological, and Environmental Statistics*, pages 297–322.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bollerslev, T., Engle, R. F., and Wooldridge, J. M. (1988). A capital asset pricing model with time-varying covariances. *The Journal of Political Economy*, 96(1):116–131.

Bonilla, E., Chai, K., and Williams, C. (2008). Multi-task Gaussian process prediction. In *NIPS*.

Boyle, P. and Frean, M. (2004). Dependent Gaussian processes. In *NIPS*.

Brooks, C., Burke, S., and Persand, G. (2001). Benchmarks and the accuracy of GARCH model estimation. *International Journal of Forecasting*, 17:45–56.

Brownlees, C. T., Engle, R. F., and Kelly, B. T. (2009). A practical guide to volatility forecasting through calm and storm. Available at SSRN: http://ssrn.com/abstract=1502915.

Bru, M. (1991). Wishart processes. *Journal of Theoretical Probability*, 4(4):725–751.

Chib, S., Nardari, F., and Shephard, N. (2006). Analysis of high dimensional multivariate stochastic volatility models. *Journal of Econometrics*, 134(2):341–371.

Cressie, N. (1993). Statistics for spatial data (wiley series in probability and statistics).

Csató, L. and Opper, M. (2001). Sparse representation for gaussian process models. In *Advances in neural information processing systems 13: proceedings of the 2000 conference*, volume 13, page 444. The MIT Press.

Dawid, A. (1973). Posterior expectations for large observations. *Biometrika*, 60(3):664.

De Finetti, B. (1956). The Bayesian approach to the rejection of outliers. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability: held at the Statistical Laboratory, University of California, June 30-July 30, l960*, page 199. University of California Press.

Engle, R. and Kroner, K. (1995). Multivariate simultaneous generalized ARCH. *Econometric theory*, 11(01):122–150.

Fox, E. and Dunson, D. (2011). Bayesian nonparametric covariance regression. *Arxiv preprint arXiv:1101.2017*.

Friedman, N. and Nachman, I. (2000). Gaussian process networks. In *Proc. Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI00)*, pages 211–219.

Gibbs, M. (1997). *Bayesian Gaussian Process for Regression and Classification*. PhD thesis, Dept. of Physics, University of Cambridge.

Goldberg, P. W., Williams, C. K., and Bishop, C. M. (1998). Regression with input-dependent noise: A Gaussian process treatment. In *NIPS*.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford University Press, USA.

Gouriéroux, C., Jasiak, J., and Sufana, R. (2009). The Wishart autoregressive process of multivariate stochastic volatility. *Journal of Econometrics*, 150(2):167–181.

Hansen, P. R. and Lunde, A. (2005). A forecast comparison of volatility models: Does anything beat a GARCH(1,1). *Journal of Applied Econometrics*, 20(7):873–889.

Harvey, A., Ruiz, E., and Shephard, N. (1994). Multivariate stochastic variance models. *The Review of Economic Studies*, 61(2):247–264.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Journel, A. and Huijbregts, C. (1978). *Mining geostatistics*. Academic Press (London and New York).

Kersting, K., Plagemann, C., Pfaff, P., and Burgard, W. (2007). Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, pages 393–400. ACM.

Kuss, M. and Rasmussen, C. (2005). Assessing approximate inference for binary Gaussian process classification. *The Journal of Machine Learning Research*, 6:1679–1704.

Lázaro-Gredilla, M. and Titsias, M. (2011). Variational heteroscedastic gaussian process regression.

MacKay, D. and Neal, R. (1994). Automatic relevance determination for neural networks. *Technical report*.

McCullough, B. and Renfro, C. (1998). Benchmarks and software standards: A case study of GARCH procedures. *Journal of Economic and Social Measurement*, 25:59–71.

Minka, T. P., Winn, J. M., Guiver, J. P., and Knowles, D. A. (2010). Infer.NET 2.4. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

Murray, I., Adams, R. P., and MacKay, D. J. (2010). Elliptical Slice Sampling. *JMLR: W&CP*, 9:541–548.

Neal, R. (1996). *Bayesian learning for neural networks*. Springer Verlag.

Neal, R. (1997). Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. *Arxiv preprint physics/9701026*.

O'Hagan, A. (1979). On outlier rejection phenomena in Bayes inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 358–367.

Osborne, M., Roberts, S., Rogers, A., Ramchurn, S., and Jennings, N. (2008). Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 109–120. IEEE Computer Society.

Poon, S.-H. and Granger, C. W. (2005). Practical issues in forecasting volatility. *Financial Analysts Journal*, 61(1):45–56.

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

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, Dept. of Computer Science, University of Toronto.

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

Schmidt, A. and O'Hagan, A. (2003). Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758.

Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse gaussian process regression. In *Workshop on AI and Statistics*, volume 9, page 2003.

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18:1257.

Teh, Y., Seeger, M., and Jordan, M. (2005). Semiparametric latent factor models. In *Workshop on Artificial Intelligence and Statistics*, volume 10.

Tomancak, P., Beaton, A., Weiszmann, R., Kwan, E., Shu, S., Lewis, S., Richards, S., Ashburner, M., Hartenstein, V., Celniker, S., et al. (2002). Systematic determination of patterns of gene expression during drosophila embryogenesis. *Genome Biol*, 3(12):0081–0088.

Turner, R. and Sahani, M. (2008). Modeling natural sounds with modulation cascade processes. *Advances in neural information processing systems*, 21.

Turner, R. E. (2010). *Statistical Models for Natural Sounds*. PhD thesis, University College London.

Uhlenback, G. and Ornstein, L. (1930). On the theory of brownian motion. *Phys. Rev.*, 36:823–841.

Vanhatalo, J., Jylanki, P., and Vehtari, A. (2009). Gaussian process regression with Student-t likelihood. In *NIPS*.

Ver Hoef, J. and Barry, R. (1998). Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Planning and Inference*, 69(2):275–294.

Wackernagel, H. (2003). *Multivariate geostatistics: an introduction with applications*. Springer Verlag.

Wilson, A. G. and Ghahramani, Z. (2010a). Copula processes. In *NIPS*.

Wilson, A. G. and Ghahramani, Z. (2010b). Generalised Wishart Processes. *Arxiv preprint arXiv:1101.0240*.

Wilson, A. G. and Ghahramani, Z. (2011). Generalised Wishart Processes. In *Uncertainty in Artificial Intelligence*. AUAI Press.

Winn, J. and Bishop, C. M. (2006). Variational message passing. *Journal of Machine Learning Research*, 6(1):661.

Yu, B., Cunningham, J., Santhanam, G., Ryu, S., Shenoy, K., and Sahani, M. (2009). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of neurophysiology*, 102(1):614.

Zinzen, R., Girardot, C., Gagneur, J., Braun, M., and Furlong, E. (2009). Combinatorial binding predicts spatio-temporal cis-regulatory activity. *Nature*, 462(7269):65–70.