Gaussian Process Regression Networks Supplementary Material

Andrew Gordon Wilson, David A. Knowles, Zoubin Ghahramani
University of Cambridge
agw38@cam.ac.uk, dak33@cam.ac.uk, zoubin@eng.cam.ac.uk

In this supplementary material, we discuss some further details of our ESS and VB inference (Sections 1 and 2), the computational complexity of our inference procedures (Section 3), and the correlation structure induced by the GPRN model (Section 4). We also discuss multimodality in the GPRN posterior (Section 5), SVLMC, and some background information and notation for Gaussian process regression (Section 7). Figure 1 shows the network structure of GPRN learned on the JURA dataset.

1 Elliptical Slice Sampling

To sample from $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$, 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 correlations between the weight and node functions in the posterior $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$. In general, MCMC samples from $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$ mix poorly because of the strong correlations in the prior $p(\boldsymbol{u}|\sigma_f, \boldsymbol{\theta}_f, \boldsymbol{\theta}_w)$ 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. Since there are no costly or numerically unstable matrix inversions in the likelihood of we also find sampling to be efficient.

With a sample from $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$, 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 our generative GPRN model we can then construct samples of

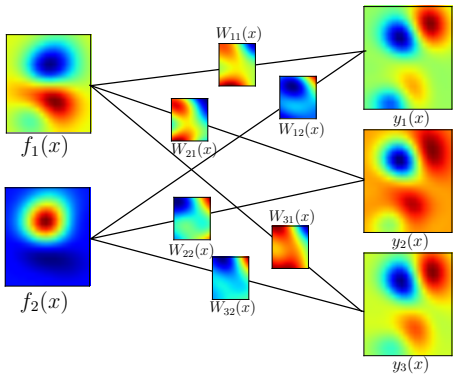

Figure 1: Network structure for the Jura dataset learnt by GPRN. The spatially varying node and weight functions are shown, along with the predictive means for the observations. The three output dimensions are cadmium, nickel and zinc concentrations respectively.

$p(\boldsymbol{y}(x_*)|W_*^i, \boldsymbol{f}_*^i, \sigma_f, \sigma_y)$, from which we can construct the predictive distribution

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

We see that even with a Gaussian observation model, the predictive distribution in (1) is an infinite mixture of Gaussians, and will generally be heavy tailed. Since mixtures of Gaussians are dense in the set of probability distributions, the predictive distribution for GPRN is highly flexible.

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.

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 $\mathbf{v} = \{\mathbf{f}, \mathbf{W}, \sigma_f^2, \sigma_y^2, a_j\}$, where the $\{a_j\}$ are signal variance hyperparameters for each node function j, so that $k_{\hat{f}_j} \to a_j k_{\hat{f}_j}$.

We specify inverse Gamma priors on $\{\sigma_f^2, \sigma_y^2, a_j\}$:

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

We use a variational posterior of the following form:

$$q(\mathbf{v}) = q_{\sigma_y^2}(\sigma_y^2) \prod_{j=1}^{Q} q_{\mathbf{f}_j}(\mathbf{f}_j) q_{\sigma_{fj}^2}(\sigma_{fj}^2) q_{a_j}(a_j) \prod_{i=1}^{P} q_{\mathbf{W}_{ij}}(\mathbf{W}_{ij}) q_{\hat{f}_{nj}}(\hat{f}_{nj})$$

where $q_{\sigma_y^2}, q_{\sigma_{fj}^2}$ and q_{a_j} are inverse Gamma distributions; $q_{f'_{nj}}, q_{\hat{f}_{nj}}$ are univariate normal distributions; and $q_{\mathbf{f}_j}$ and $q_{\mathbf{W}_{ij}}$ are multivariate normal distributions. The lengthscales $\{\theta_f, \theta_w\}$ do not appear here because they are optimised in the M-step below (we could equivalently consider a point mass "distribution" on the lengthscales).

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

$$w_{nij} := W_{ij}(x_n), \qquad f'_{nj} := f_j(x_n) \tag{2}$$

$$t_{nij} := w_{nij} \hat{f}_{nj}, \qquad s_{in} := \sum_j t_{nij} \tag{3}$$

We refer to these as "derived" variables. Note that the observations $y_i(x_n) \sim \mathcal{N}(s_{in}, \sigma_y^2)$ and that $\hat{f}_{nj} \sim \mathcal{N}(f'_{nj}, \sigma_{f_j}^2)$. These derived variables are all given univariate normal "pseudo-marginals" which Variational message passing uses as conduits to pass appropriate moments, resulting in the same updates as standard

VB (see Winn and Bishop (2006) for details). Using the derived variables the full model can be written as

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

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

$$\begin{aligned} \log \mathcal{N}(\mathbf{f}_j; \mathbf{0}, a_j K_f) &\stackrel{c}{=} -\frac{1}{2} \log |a_j K_j| - \frac{1}{2} \mathbf{f}_j^T (a_j K_f)^{-1} \mathbf{f}_j \\ &= -\frac{N}{2} \log a_j - \frac{1}{2} \log |K_j| - \frac{1}{2} a_j^{-1} \mathbf{f}_j^T K_f^{-1} \mathbf{f}_j \end{aligned} \tag{4}$$

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 $\mathrm{IG}\left(a_j; \frac{N}{2} - 1, \frac{1}{2} \langle \mathbf{f}_j^T K_f^{-1} \mathbf{f}_j \rangle\right)$. 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 (4)), which is straightforward to evaluate and differentiate. From (4) 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 since $\mathbf{f}_j$ has multivariate normal variational posterior. Analogously for θ_w we consider the contribution of $\mathcal{N}(\mathbf{W}_{pq}; 0, K_W)$.

VB predictive distribution. The predictive distribution for the output $\mathbf{y}^*(x)$ at a new input location x is calculated as

$$p(\mathbf{y}^*(x)|\mathcal{D}) = \int p(\mathbf{y}^*(x)|W(x), f(x)) p(W(x), f(x)|\mathcal{D}) dW df \tag{5}$$

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

$$p(\mathbf{y}^*(x)|\mathcal{D}) = \int p(\mathbf{y}^*(x)|W(x), \hat{f}(x)) q(W) q(\hat{f}) dW d\hat{f} \tag{6}$$

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

$$\bar{\mathbf{y}}^*(x)_i = \sum_k \mathbb{E}(W_{ik}^*)\mathbb{E}[\hat{f}_k^*] \tag{7}$$

$$\mathrm{cov}(\mathbf{y}^*(x))_{ij} = \sum_k [\mathbb{E}(W_{ik}^*)\mathbb{E}(W_{jk}^*)\mathrm{var}(\hat{f}_k^*) + \delta_{ij}\mathrm{var}(W_{ik}^*)\mathbb{E}(\hat{f}_k^{*2})] + \delta_{ij}\mathbb{E}[\sigma_y^2] \tag{8}$$

where $\delta_{ij} = \mathbb{I}[i = j]$ is the Kronecker delta function, $W_{ik}^* = W_{ik}(x)$ and $\hat{f}_k^* = \hat{f}_k(x)$. The moments of W_{ik}^* and $\hat{f}_k^*$ under q are straightforward to obtain from $q(W)$ and $q(f)$ respectively using the standard GP prediction equations (see Rasmussen and Williams (2006)). It is also of interest to calculate the *noise* covariance. Recall our model can be written as

$$\boldsymbol{y}(x) = \underbrace{W(x)\boldsymbol{f}(x)}_{\text{signal}} + \underbrace{\sigma_f W(x)\boldsymbol{\epsilon} + \sigma_y \boldsymbol{z}}_{\text{noise}} \tag{9}$$

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

$$\mathrm{cov}(\boldsymbol{n}(x))_{ij} = \sum_k [\mathbb{E}[\sigma_{f_k}^2]\mathbb{E}(W_{ik}^*)\mathbb{E}(W_{jk}^*) + \delta_{ij}\mathrm{var}(W_{jk}^*)] + \delta_{ij}\mathbb{E}[\sigma_y^2] \tag{10}$$

3 Computational Considerations

The computational complexity of a Markov chain Monte Carlo GPRN approach is mainly limited by taking the Cholesky decomposition of the block diagonal C_B, an $Nq(p+1) \times Nq(p+1)$ matrix in the prior on GP function values. 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[1] 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 complexity of $\mathcal{O}(Nqp + Np^3)$ per likelihood evaluation. 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 $\mathcal{O}(Nqp)$.

The computational complexity of variational Bayes 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.

We see that when fixing q and p, the computational complexity of GPRN scales cubically with the number of data points, like standard Gaussian process regression. On modern computers, this limits GPRN to datasets with fewer than about $N = 30000$ points. However, one could adopt a sparse representation of GPRN, for example following the DTC, (Csató and Opper, 2001; Seeger et al., 2003; Quiñonero-Candela and Rasmussen, 2005; Rasmussen and Williams, 2006), PITC (Quiñonero-Candela and Rasmussen, 2005), or FITC (Snelson and Ghahramani, 2006) approximations, which would lead to $\mathcal{O}(M^2N)$ scaling where

[1] In this context, "inverting" means decomposing (e.g. a Cholesky decomposition of) the matrix $\Sigma(x)$ in question, for instance to take the the determinant of $\Sigma^{-1}(x)$, or the matrix vector product $\boldsymbol{y}^\top(x)\Sigma^{-1}(x)\boldsymbol{y}(x)$.

$M \ll N$. If the input space $\mathcal{X}$ is on a full (but not necessarily equispaced) grid, in that it can be expressed as a cartesian product of the input locations for each dimension, then the complexity in N reduces to $\mathcal{O}(N \log N)$.

Fixing q and N, the per iteration (for MCMC or VB) computational complexity of GPRN scales linearly with p. Overall, the computational demands of GPRN compare favourably to most multi-task GP models, which commonly have a complexity of $\mathcal{O}(p^3N^3)$ (e.g. SLFM, LMC, and CMOGP in the experiments) and do not account for either input dependent signal or noise correlations. Moreover, multivariate volatility models, which account for input dependent noise correlations, are commonly intractable for $p > 5$ (Engle, 2002; Gouriéroux et al., 2009). The 1000 dimensional gene expression experiment is tractable for GPRN, but intractable for the alternative multi-task models used on the 50 dimensional gene expression set, and the multivariate volatility models.

Using either MCMC or VB with GPRN, the memory requirement is $\mathcal{O}(N^2)$ if all covariance kernels share the same hyperparameters, is $\mathcal{O}(qN^2)$ is the node functions have different kernel hyperparameters, and $\mathcal{O}(pq^2N^2)$ if all GPs have different kernel hyperparameters. This memory requirement comes from storing the information in the block diagonal C_B in the prior $p(\boldsymbol{u}|\sigma_f, \boldsymbol{\theta}_f, \boldsymbol{\theta}_w)$.

4 Covariance structure

In this section we derive the covariance structure induced by the GPRN of Section 2 in the main text, explicitly in terms of the kernel functions k_w and k_f for the weight and node functions. We assume that the weight functions share a kernel k_w and the node functions share a kernel k_f.

In the GPRN specification of Section 2 in the main text, a priori, at a particular location x,

$$\mathbb{E}[\boldsymbol{y}(x)] = 0\,, \tag{11}$$

$$\mathrm{cov}(\boldsymbol{y}(x)) = (1 + \sigma_f^2 + \sigma_y^2)I\,, \tag{12}$$

because $W(x)$ and $\boldsymbol{f}(x)$ are comprised of independent mean zero Gaussian processes. One can always preprocess data so that (11) and (12) accord with prior beliefs. The modelling power of standard Gaussian process regression comes primarily through the covariance kernel and its hyperparameters, which allow for a flexible covariance structure between data points at different locations. Likewise, the modelling power of GPRN comes from the induced process over $\Sigma(x_*) = \mathrm{cov}[\boldsymbol{y}(x_*)|x_*]$ and how its covariance structure directly relates to the Gaussian process covariance kernels k_w and k_f used in the weight and node functions. Given the weight and node functions, the covariance matrix $\Sigma(x_*) = \mathrm{cov}[\boldsymbol{y}(x_*)|x_*]$ is

$$\Sigma(x_*) = \underbrace{W(x_*)\boldsymbol{f}(x_*)\boldsymbol{f}(x_*)^\top W(x_*)^\top}_{\text{signal}} + \underbrace{\sigma_f^2 W(x_*)W(x_*)^\top + \sigma_y^2 I}_{\text{noise}}\,. \tag{13}$$

Theorem 4.1 gives the covariance structure for the induced process in (13). The covariances induced by the signal and noise models are separately labelled.

Theorem 4.1. *Abbreviating the kernels $k_f(x, x')$ and $k_w(x, x')$ as k_f and k_w, for $i \neq j \neq l \neq s$, and locations x and x',*

$$cov[\Sigma_{ii}(x), \Sigma_{ii}(x')] = \underbrace{2q^2k_w^2k_f^2 + 2q(k_w^2 + k_v^2)}_{signal} + \underbrace{2q\sigma_f^4k_w^2 + \sigma_y^4}_{noise}\,. \tag{14}$$

$$cov[\Sigma_{ij}(x), \Sigma_{ji}(x')] = cov(\Sigma_{ij}(x), \Sigma_{ij}(x')) = \underbrace{0.5(3q + q^2)k_w^2k_f^2 + qk_w^2}_{signal} + \underbrace{\sigma_f^4k_w^2 + \sigma_y^4}_{noise}\,. \tag{15}$$

$$cov[\Sigma_{ij}(x), \Sigma_{ls}(x')] = 0\,. \tag{16}$$

Proof 4.1. *$W_{ij}(x) \sim \mathcal{GP}(0, k_w)$ and $\boldsymbol{f}_i(x) \sim \mathcal{GP}(0, k_f)$. We first consider the contribution from the signal, $A(x) = W(x)\boldsymbol{f}(x)\boldsymbol{f}(x)^\top W(x)^\top$. The entry $A_{11}(x) = \sum_{i=1}^q W_{1i}(x)\boldsymbol{f}_i(x) \sum_{j=1}^q W_{1j}(x)\boldsymbol{f}_j(x)$. We will focus*

on this entry, since all the diagonal entries are i.i.d., and $cov(A_{11}(x), A_{11}(x')) = cov(A_{ii}(x), A_{ii}(x'))$. *Abbreviating* $W_{11}(x'), W_{12}(x'), \boldsymbol{f}_1(x'), \boldsymbol{f}_2(x')$ *respectively as* $W'_{11}, W'_{12}, \boldsymbol{f}'_1, \boldsymbol{f}'_2$, *and* $W_{11}(x), W_{12}(x), \boldsymbol{f}_1(x), \boldsymbol{f}_2(x)$ *as* $W_{11}, W_{12}, \boldsymbol{f}_1, \boldsymbol{f}_2$, *and given that the node and weight functions are independent,*

$$cov(A_{11}(x), A_{11}(x')) = q cov(W_{11}^2 \boldsymbol{f}_1^2, W_{11}'^2 \boldsymbol{f}_1'^2) + 2q(q-1) cov(W_{11}\boldsymbol{f}_1 W_{12}\boldsymbol{f}_2, W'_{11}\boldsymbol{f}'_1 W'_{12}\boldsymbol{f}'_2)\,. \tag{17}$$

$$cov(W_{11}W_{12}\boldsymbol{f}_1\boldsymbol{f}_2, W'_{11}W'_{12}\boldsymbol{f}'_1\boldsymbol{f}'_2) = \mathbb{E}[W_{11}W_{12}\boldsymbol{f}_1\boldsymbol{f}_2 W'_{11}W'_{12}\boldsymbol{f}'_1\boldsymbol{f}'_2] - \mathbb{E}[W_{11}W_{12}\boldsymbol{f}_1\boldsymbol{f}_2]\mathbb{E}[W'_{11}W'_{12}\boldsymbol{f}'_1\boldsymbol{f}'_2]\,, \tag{18}$$

$$= \mathbb{E}[W_{11}W_{12}\boldsymbol{f}_1\boldsymbol{f}_2 W'_{11}W'_{12}\boldsymbol{f}'_1\boldsymbol{f}'_2]\,. \tag{19}$$

Since W_{11} *and* W'_{11} *are jointly Gaussian and marginally* $\mathcal{N}(0,1)$ *random variables, we can write*

$$W'_{11} = k_w W_{11} + \sqrt{1-k_w^2}\gamma_1\,, \tag{20}$$

where γ_1 *is* $\mathcal{N}(0,1)$ *and independent from* W_{11} *and* W'_{11}. *Substitutions similar to* (20) *can be made for* $W'_{12}, \boldsymbol{f}'_1, \boldsymbol{f}'_2$, *so that*

$$\mathbb{E}[W_{11}W_{12}\boldsymbol{f}_1\boldsymbol{f}_2 W'_{11}W'_{12}\boldsymbol{f}'_1\boldsymbol{f}'_2] = k_w^2 k_f^2\,. \tag{21}$$

Likewise,

$$cov(W_{11}(x)^2\boldsymbol{f}_1(x)^2, W_{11}(x')^2\boldsymbol{f}_1(x')^2) = 2(k_w^2k_f^2 + k_w^2 + k_v^2), \tag{22}$$

using $\mathbb{E}[a^4] = 3$ *if* $a \sim \mathcal{N}(0,1)$. *Therefore*

$$cov(A_{ii}(x), A_{ii}(x')) = 2q(k_w^2k_f^2 + k_w^2 + k_f^2) + 2q(q-1)k_w^2k_f^2 = 2q^2k_w^2k_f^2 + 2q(k_w^2 + k_f^2)\,. \tag{23}$$

Equations (14)-(16) *follow similarly.*

Corollary 4.1. *Since* $\mathbb{E}[\Sigma(x)] = I =$ *constant, and the covariances are proportional to the kernel functions(s), the induced process on* $\Sigma(x)$ *is weakly stationary if the kernel function(s) are stationary – that is, if* $k(x,x') = k(x+a, x'+a)$ *for all feasible constants* a *and all kernels* k. *This holds, for example, if* $k(x,x')$ *is a function of* $||x-x'||$.

Corollary 4.2. *If* $k(x,x') \to 0$ *as* $||x-x'|| \to \infty$ *then*

$$\lim_{||x-x'||\to\infty} cov[\Sigma_{ij}(x), \Sigma_{ls}(x')] = 0\,, \quad \forall\ i,j,l,s\,. \tag{24}$$

Theorem 5.1 and its corollaries clarify the importance of the kernels k_w and k_f and their hyperparameters (particularly length-scales): through k_w and k_f we can explicitly specify the desired prior covariance structure of the GPRN model.

5 Multimodality in the GPRN posterior

It is possible to reduce the number of modes in the posterior by constraining the weights W or nodes $\mathbf{f}$ to be positive. For MCMC it is straightforward to do this by exponentiating the weights, as in Adams 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 in their case would centre probability mass between symmetric modes. MCMC and VB approaches are more robust to this problem. MCMC can explore these symmetric modes, and VB will concentrate on one of these modes without losing the expressivity of the GPRN prior.

6 SVLMC

In the main text, we compared to 8 multiple output GP methods: CMOGP, SLFM, ICM, co-kriging, LMC, CMOFITC, CMODTC, CMOPITC. We also tested SVLMC, but found that it was not tractable on the gene expression or jura datasets, and since it does not account for input dependent noise correlations, it is not applicable to the multivariate volatility datasets. In this section, we briefly present some of the results that we did find. The SVLMC does not incorporate hyperparameters, and we found performance on these datasets to be sensitive to covariance kernel hyperparameters, particularly length-scales. So we introduced covariance kernel hyperparameters (like length-scales) into the SVLMC. We found the covariogram (based on the ACFs), which is typically used in geostatistics, too crude to set these hyperparameters, so we used the GPRN to learn hyperparameters. We ran an instance of SVLMC on the $p = 50$ gene expression dataset. After having taken 5×10^7 samples (taking 20 hours on a 2.3 GHz i5 Intel Duo Core processor), the SMSE was 4.94, compared to an SMSE of 0.32 and 0.34 with GPRN (ESS) and GPRN (VB) respectively. For GPRN (ESS) we took 6000 samples, which took 40 seconds. GPRN (VB) took 12 seconds to converge.

7 Gaussian processes

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 $w(x)$:

$$w(x) \sim \mathcal{GP}(m(x), k(x, x')), \tag{25}$$

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

$$m(x) = \mathbb{E}[w(x)]\,, \tag{26}$$
$$k(x, x') = \mathrm{cov}[w(x), w(x')]\,. \tag{27}$$

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

$$(w(x_1), w(x_2), \ldots, w(x_N))^\top \sim \mathcal{N}(\boldsymbol{\mu}, K)\,, \tag{28}$$

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:

$$k_{\mathrm{SE}}(x, x') = A \exp(-0.5||x - x'||^2/l^2)\,. \tag{29}$$

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:

$$k_{\mathrm{OU}}(x, x') = \exp(-||x - x'||/l)\,. \tag{30}$$

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,

$$k_{\text{Matérn}}(x,x') = \frac{2^{1-\alpha}}{\Gamma(\alpha)}(\frac{\sqrt{2\alpha}||x-x'||}{l})^{\alpha}K_{\alpha}(\frac{\sqrt{2\alpha}||x-x'||}{l}), \tag{31}$$

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 $p = \alpha + 1/2$.[2] 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. $k = a_1k_1 + a_2k_2 + a_3k_3$, 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:

$$\begin{bmatrix} w(x_*) \\ \boldsymbol{w} \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \begin{bmatrix} k(x_*, x_*) & \boldsymbol{k}_*^\top \\ \boldsymbol{k}_* & K + \sigma_n^2 I \end{bmatrix}), \tag{32}$$

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

$$\mu_* = \boldsymbol{k}_*^\top (K + \sigma_n^2 I)^{-1} \boldsymbol{y}, \tag{33}$$

$$v_* = k(x_*, x_*) - \boldsymbol{k}_*^\top (K + \sigma_n^2 I)^{-1} \boldsymbol{k}_*. \tag{34}$$

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 (34) 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 (33) and (34) is Gaussian, the marginals in the prior (from marginalising equation (28)) 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.

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 *ICML*.

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.

[2] 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.

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.

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.

Engle, R. (2002). New frontiers for ARCH models. *Journal of Applied Econometrics*, 17(5):425–446.

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

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

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.

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.

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

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.

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

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