VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences
Computer Science

Multiple Output Gaussian Process Regression

Phillip Boyle and Marcus Frean

Technical Report CS-TR-05/2
April 2005

School of Mathematical and Computing Sciences
Victoria University
PO Box 600, Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

Multiple Output Gaussian Process Regression

Phillip Boyle and Marcus Frean

Technical Report CS-TR-05/2
April 2005

Abstract

Gaussian processes are usually parameterised in terms of their covariance functions. However, this makes it difficult to deal with multiple outputs, because ensuring that the covariance matrix is positive definite is problematic. An alternative formulation is to treat Gaussian processes as white noise sources convolved with smoothing kernels, and to parameterise the kernel instead. Using this, we extend Gaussian processes to handle multiple, coupled outputs.

1 Introduction

Gaussian process regression has many desirable properties, such as ease of obtaining and expressing uncertainty in predictions, the ability to capture a wide variety of behaviour through a simple parameterisation, and a natural Bayesian interpretation [14, 3, 8]. Because of this they have been suggested as replacements for supervised neural networks in non-linear regression [7, 17], extended to handle classification tasks [10, 16, 5], and used in a variety of other ways (e.g. [15, 13]). A Gaussian process (GP), as a set of jointly Gaussian random variables, is completely characterised by a covariance matrix with entries determined by a covariance function. Traditionally, such models have been specified by parameterising the covariance function (*i.e.* a function specifying the covariance of output values given any two input vectors). In general this needs to be a positive definite function to ensure positive definiteness of the covariance matrix.

Most GP implementations model only a single output variable. Attempts to handle multiple outputs generally involve using an independent model for each output - a method known as multi-kriging [17] - but such models cannot capture the structure in outputs that covary. As an example, consider the two tightly coupled outputs shown at the top of Figure 2, in which one output is simply a shifted version of the other. Here we have detailed knowledge of output 1, but sampling of output 2 is sparse. A model that treats the outputs as independent cannot exploit their obvious similarity - intuitively, we should make predictions about output 2 using what we learn from both output 1 *and* 2.

Joint predictions are possible (e.g. co-kriging [2]) but are problematic in that it is not clear how covariance functions should be defined [4]. Although there are many known positive definite autocovariance functions (e.g. Gaussians and many others [1, 8]), it is difficult to define cross-covariance functions that result in positive definite covariance matrices. Contrast this to neural network modelling, where the handling of multiple outputs is routine.

An alternative to directly parameterising covariance functions is to treat GPs as the outputs of stable linear filters. For a linear filter, the output in response to an input $x(t)$ is $y(t) = h(t) \star x(t) = \int_{-\infty}^{\infty} h(t-\tau)x(\tau)d\tau$, where $h(t)$ defines the impulse response of the filter and $\star$ denotes convolution. Provided the linear filter is stable and $x(t)$ is Gaussian white noise, the output process $y(t)$ is necessarily a Gaussian process. It is also possible to characterise p-dimensional stable linear filters, with M-inputs and N-outputs, by a set of $M \times N$ impulse responses. In general, the resulting N outputs are dependent Gaussian processes. Now we can model multiple dependent outputs by parameterising the set of impulse responses for a multiple output linear filter, and inferring the parameter values from data that we observe. Instead of specifying and parameterising positive definite covariance functions, we now specify and parameterise impulse responses. The only restriction is that the filter be linear and stable, and this is achieved by requiring the impulse responses to be absolutely integrable.

Constructing GPs by stimulating linear filters with Gaussian noise is equivalent to constructing GPs through kernel convolutions. A Gaussian process $V(s)$ can be constructed over a region $\mathcal{S}$ by convolving a continuous white noise process $X(s)$ with a smoothing kernel $h(s)$, $V(s) = h(s) \star X(s)$ for $s \in \mathcal{S}$, [6]. To this can be added a second white noise source, representing measurement uncertainty, and together this gives a model for observations Y. This view of GPs is shown in graphical form in Figure $1(a)$. The convolution approach has been used to formulate flexible nonstationary covariance functions [12, 11]. Furthermore, this idea can be extended to model multiple dependent output processes by assuming a single common latent process [6]. For example, two dependent processes $V_1(s)$ and $V_2(s)$ are constructed from a

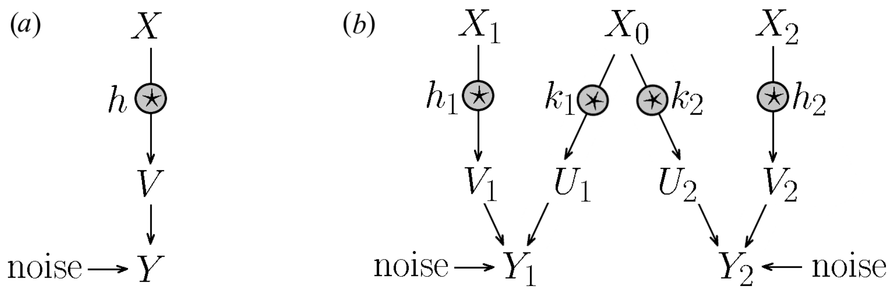

Figure 1: (a) Gaussian process prior for a single output. The output Y is the sum of two Gaussian white noise processes, one of which has been convolved ($\star$) with a kernel (h).
(b) The model for two dependent outputs Y_1 and Y_2. All of X_0, X_1, X_2 and the "noise" contributions are independent Gaussian white noise sources. Notice that if X_0 is forced to zero Y_1 and Y_2 become independent processes as in (a) - we use this as a control model.

shared dependence on $X(s)$ for $s \in \mathcal{S}_0$, as follows

$$V_1(s) = \int_{\mathcal{S}_0 \cup \mathcal{S}_1} h_1(s-\lambda)X(\lambda)d\lambda \quad \text{and} \quad V_2(s) = \int_{\mathcal{S}_0 \cup \mathcal{S}_2} h_2(s-\lambda)X(\lambda)d\lambda$$

where $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{S}_2$ is a union of disjoint subspaces. $V_1(s)$ is dependent on $X(s), s \in \mathcal{S}_1$ but not $X(s), s \in \mathcal{S}_2$. Similarly, $V_2(s)$ is dependent on $X(s), s \in \mathcal{S}_2$ but not $X(s), s \in \mathcal{S}_1$. This allows $V_1(s)$ and $V_2(s)$ to possess independent components.

In this paper, we model multiple outputs somewhat differently to [6]. Instead of assuming a single latent process defined over a union of subspaces, we assume multiple latent processes, each defined over $\Re^p$. Some outputs may be dependent through a shared reliance on common latent processes, and some outputs may possess unique, independent features through a connection to a latent process that affects no other output.

2 Two Dependent Outputs

Consider two outputs $Y_1(s)$ and $Y_2(s)$ over a region $\Re^p$, where $s \in \Re^p$. We have N_1 observations of output 1 and N_2 observations of output 2, giving us data $\mathcal{D}_1 = \{s_{1,i}\,,\, y_{1,i}\}_{i=1}^{N_1}$ and $\mathcal{D}_2 = \{s_{2,i}\,,\, y_{2,i}\}_{i=1}^{N_2}$. We wish to learn a model from the combined data $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2\}$ in order to predict $Y_1(s')$ or $Y_2(s')$, for $s' \in \Re^p$. As shown in Figure 1(b), we can model each output as the linear sum of three stationary Gaussian processes. One of these (V) arises from a noise source unique to that output, under convolution with a kernel h. A second (U) is similar, but arises from a separate noise source X_0 that influences *both* outputs (although via different kernels, k). The third is additive noise as before.

Thus we have $Y_i(s) = U_i(s) + V_i(s) + W_i(s)$, where $W_i(s)$ is a stationary Gaussian white noise process with variance, σ_i^2, $X_0(s), X_1(s)$ and $X_2(s)$ are independent stationary Gaussian white noise processes, $U_1(s), U_2(s), V_1(s)$ and $V_2(s)$ are Gaussian processes given by $U_i(s) = k_i(s) \star X_0(s)$ and $V_i(s) = h_i(s) \star X_i(s)$.

The k_1, k_2, h_1, h_2 are parameterised Gaussian kernels where $k_1(s) = v_1 \exp\left(-\frac{1}{2}s^T A_1 s\right)$, $k_2(s) = v_2 \exp\left(-\frac{1}{2}(s-\mu)^T A_2 (s-\mu)\right)$, and $h_i(s) = w_i \exp\left(-\frac{1}{2}s^T B_i s\right)$. Note that $k_2(s)$ is offset from zero by μ to allow modelling of outputs that are coupled and translated relative to one another.

We wish to derive the set of functions $C_{ij}^Y(d)$ that define the autocovariance ($i = j$) and cross-covariance ($i \neq j$) between the outputs i and j, for a given separation d between arbitrary inputs s_a and s_b. By solving a convolution integral, (appendix), $C_{ij}^Y(d)$ can be expressed in a closed form , and is fully determined by the parameters of the Gaussian kernels and the noise variances σ_1^2 and σ_2^2 as follows:

$$
\begin{aligned}
C_{11}^Y(d) &= C_{11}^U(d) + C_{11}^V(d) + \delta_{ab}\sigma_1^2 & \qquad C_{12}^Y(d) &= C_{12}^U(d) \\
C_{22}^Y(d) &= C_{22}^U(d) + C_{22}^V(d) + \delta_{ab}\sigma_2^2 & \qquad C_{21}^Y(d) &= C_{21}^U(d)
\end{aligned}
$$

where

$$
\begin{aligned}
C_{ii}^U(d) &= \frac{\pi^{\frac{p}{2}} v_i^2}{\sqrt{|A_i|}} \exp\left(-\frac{1}{4}\, d^T A_i d\right) \\
C_{12}^U(d) &= \frac{(2\pi)^{\frac{p}{2}} v_1 v_2}{\sqrt{|A_1 + A_2|}} \exp\left(-\frac{1}{2}(d-\mu)^T \Sigma (d-\mu)\right) \\
C_{21}^U(d) &= \frac{(2\pi)^{\frac{p}{2}} v_1 v_2}{\sqrt{|A_1 + A_2|}} \exp\left(-\frac{1}{2}(d+\mu)^T \Sigma (d+\mu)\right) = C_{12}^U(-d) \\
C_{ii}^V(d) &= \frac{\pi^{\frac{p}{2}} w_i^2}{\sqrt{|B_i|}} \exp\left(-\frac{1}{4}\, d^T B_i d\right)
\end{aligned}
$$

with $\Sigma \;=\; A_1(A_1 + A_2)^{-1}A_2 \;=\; A_2(A_1 + A_2)^{-1}A_1$.

Given $C_{ij}^Y(d)$ then, we can construct the covariance matrices $\mathbf{C}_{11}, \mathbf{C}_{12}, \mathbf{C}_{21}$, and $\mathbf{C}_{22}$ as follows

$$
\mathbf{C}_{ij} = \begin{bmatrix} C_{ij}^Y(s_{i,1} - s_{j,1}) & \cdots & C_{ij}^Y(s_{i,1} - s_{j,N_j}) \\ \vdots & \ddots & \vdots \\ C_{ij}^Y(s_{i,N_i} - s_{j,1}) & \cdots & C_{ij}^Y(s_{i,N_i} - s_{j,N_j}) \end{bmatrix} \tag{1}
$$

Together these define the positive definite symmetric covariance matrix $\mathbf{C}$ for the *combined* output data $\mathcal{D}$:

$$
\mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix} \tag{2}
$$

We define a set of hyperparameters $\mathbf{\Theta}$ that parameterise $\{v_1, v_2, w_1, w_2, A_1, A_2, B_1, B_2, \mu, \sigma_1, \sigma_2\}$. Now, we can calculate the log-likelihood

$$
\begin{aligned}
\mathcal{L} &= -\frac{1}{2}\log\left|\mathbf{C}\right| - \frac{1}{2}\mathbf{y}^T\mathbf{C}^{-1}\mathbf{y} - \frac{N_1 + N_2}{2}\log 2\pi \\
\text{where} \quad \mathbf{y}^T &= \begin{bmatrix} y_{1,1} & \cdots & y_{1,N_1} & y_{2,1} & \cdots & y_{2,N_2} \end{bmatrix}
\end{aligned}
$$

and $\mathbf{C}$ is a function of $\mathbf{\Theta}$ and $\mathcal{D}$.

Learning a model now corresponds to either maximising the log-likelihood $\mathcal{L}$, or maximising the posterior probability $P(\mathbf{\Theta} \,|\, \mathcal{D})$. Alternatively, we can simulate the predictive distribution for y by taking samples from the joint $P(\mathbf{y}, \mathbf{\Theta} \,|\, \mathcal{D})$, using Markov Chain Monte Carlo methods [9].

The predictive distribution at a point s' on output i given $\mathbf{\Theta}$ and $\mathcal{D}$ is Gaussian with mean $\hat{y}'$ and variance $\sigma^2_{\hat{y}'}$ given by

$$
\begin{aligned}
& \hat{y}' = \mathbf{k}^T \mathbf{C}^{-1} \mathbf{y} \\
\text{and} \quad & \sigma^2_{\hat{y}'} = \kappa - \mathbf{k}^T \mathbf{C}^{-1} \mathbf{k} \\
\text{where} \quad & \kappa = C^Y_{ii}(0) = v_i^2 + w_i^2 + \sigma_i^2 \\
\text{and} \quad & \mathbf{k} = \left[C^Y_{i1}(s' - s_{1,1}) \ldots C^Y_{i1}(s' - s_{1,N_1}) \quad C^Y_{i2}(s' - s_{2,1}) \ldots C^Y_{i2}(s' - s_{2,N_2}) \right]^T
\end{aligned}
$$

2.1 Example 1 - Strongly dependent outputs over 1d input space

Consider two outputs, observed over a 1d input space. Let $A_i = \exp(f_i)$, $B_i = \exp(g_i)$, and $\sigma_i = \exp(\beta_i)$. Our hyperparameters are $\mathbf{\Theta} = \{v_1, v_2, w_1, w_2, f_1, f_2, g_1, g_2, \mu, \beta_1, \beta_2\}$ where each element of $\mathbf{\Theta}$ is a scalar.

We set Gaussian priors over $\mathbf{\Theta}$, as follows:
$P(v_i) = \mathcal{N}(v_i, 0, 0.5^2)$, $P(w_i) = \mathcal{N}(w_i, 0, 0.5^2)$, $P(f_i) = \mathcal{N}(f_i, 3.5, 1^2)$, $P(g_i) = \mathcal{N}(g_i, 3.5, 1^2)$, $P(\mu) = \mathcal{N}(\mu, 0, 0.5^2)$, $P(\beta_i) = \mathcal{N}(\beta_i, -4, 0.75^2)$.
where $\mathcal{N}(x, \mu, \sigma^2)$ is a Gaussian distribution for variable x with mean μ and variance σ^2.

We generated $N = 48$ data points by taking $N_1 = 32$ samples from output 1 and $N_2 = 16$ samples from output 2. The samples from output 1 were linearly spaced in the interval $[-1, 1]$ and those from output 2 were uniformly spaced in the region $[-1, -0.15] \cup [0.65, 1]$. All samples were taken under additive Gaussian noise, $\sigma = 0.025$. To build our model, we maximised $P(\mathbf{\Theta}|\mathcal{D}) \propto P(\mathcal{D} \,|\, \mathbf{\Theta})\, P(\mathbf{\Theta})$ using a multistart conjugate gradient algorithm, with 5 starts, sampling from $P(\mathbf{\Theta})$ for initial conditions.

The resulting dependent model is shown in Figure 2 along with an independent (control) model with no coupling (see Figure 1). Observe that the dependent model has learned the coupling and translation between the outputs, and has filled in output 2 where samples are missing. The control model cannot achieve such infilling as it is consists of two independent Gaussian processes.

2.2 Example 2 - Strongly dependent outputs over 2d input space

Consider two outputs, observed over a 2d input space. Let

$$
A_i = \frac{1}{\alpha_i^2} \mathbf{I} \qquad B_i = \frac{1}{\tau_i^2} \mathbf{I} \qquad \text{where } \mathbf{I} \text{ is the identity matrix.}
$$

Furthermore, let $\sigma_i = \exp(\beta_i)$. In this toy example, we set $\mu = 0$, so our hyperparameters become $\mathbf{\Theta} = \{v_1, v_2, w_1, w_2, \alpha_1, \alpha_2, \tau_1, \tau_2 \beta_1, \beta_2\}$ where each element of $\mathbf{\Theta}$ is a scalar.

We set Gaussian priors over $\mathbf{\Theta}$, as follows:
$P(v_i) = \mathcal{N}(v_i, 1, 0.5^2)$, $P(w_i) = \mathcal{N}(w_i, 0, 0.5^2)$, $P(\alpha_i) = \mathcal{N}(\alpha_i, 0.3, 0.1^2)$, $P(\tau_i) = \mathcal{N}(\tau_i, 0.3, 0.1^2)$, $P(\beta_i) = \mathcal{N}(\beta_i, -3.7, 0.5^2)$.

We generated 117 data points by taking 81 samples from output 1 and 36 samples from output 2. Both sets of samples formed uniform lattices over the region $[-0.9, 0.9] \otimes [-0.9, 0.9]$ and were taken with additive Gaussian noise, $\sigma = 0.025$. To build our model, we maximised $P(\mathbf{\Theta}|\mathcal{D})$ as before.

The dependent model is shown in Figure 3 along with an independent control model. The dependent model has filled in output 2 where samples are missing. Again, the control model cannot achieve such in-filling as it is consists of two independent Gaussian processes.

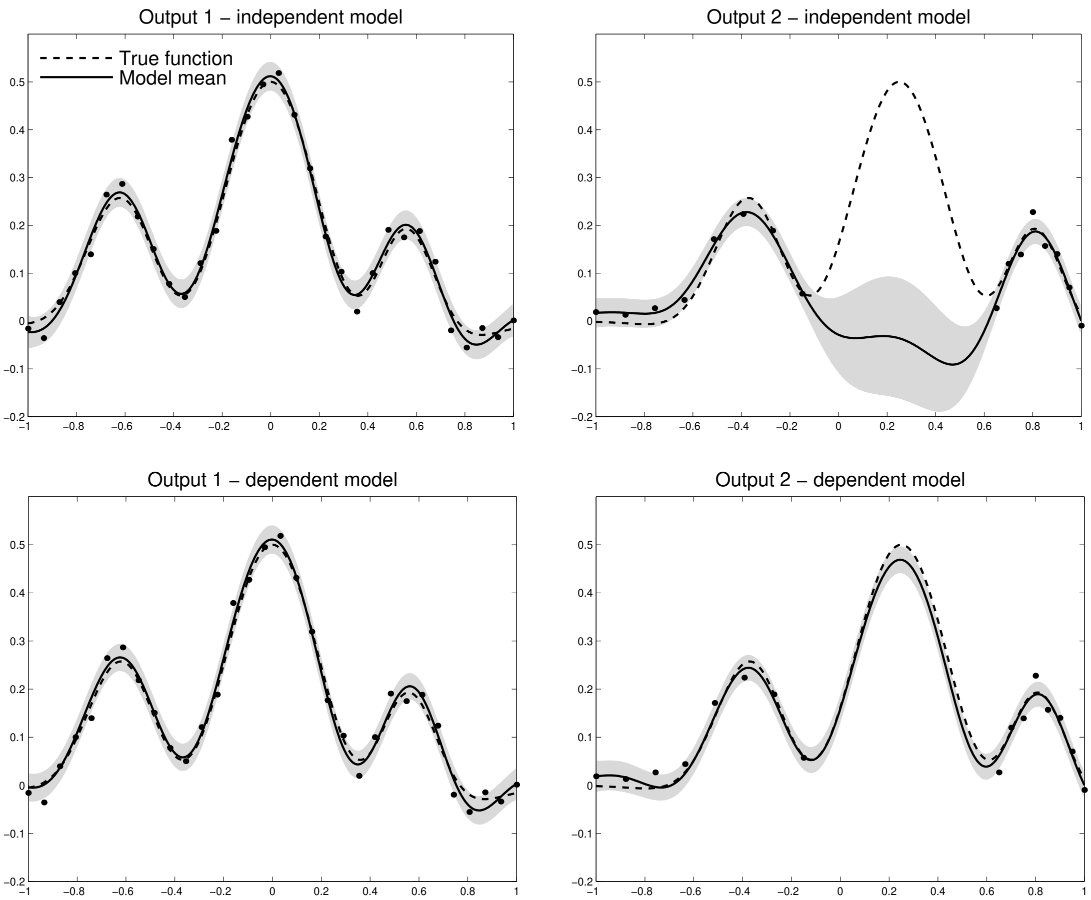

Figure 2: Strongly dependent outputs where output 2 is simply a translated version of output 1, with independent Gaussian noise, $\sigma = 0.025$. The solid lines represent the model, the dotted lines are the true function, and the dots are samples. The shaded regions represent 1σ error bars for the model prediction. *(top)* Independent model of the two outputs. *(bottom)* Dependent model.

2.3 Example 3 - Partially Coupled Outputs

The examples in sections 2.1 and 2.2 show GP models of two dependent outputs which had no independent components. The top of Figure 4 shows a case in which a pair of outputs are dependent, but also have features that are independent of each other.

Our hyperparameters $\mathbf{\Theta}$ are the same as those from section 2.1 and again we use Gaussian priors. We generated 48 data points by taking 32 samples from output 1 and 16 samples from output 2. The samples were linearly spaced in the interval $[-1, 1]$ for output 1 and $[-1, 0]$ for output 2. All samples were taken with additive Gaussian noise, $\sigma = 0.025$.

As before, we then maximised $P(\mathbf{\Theta}|\mathcal{D})$ and made predictions with the resulting model along with an independent model for comparison. As Figure 4 shows, the dependent model has learned the coupling between the outputs, and attempts to fill in output 2 where samples are missing. The in-filling is not as striking as the previous examples because output 2 possesses an independent component, but is much better than the default GP model.

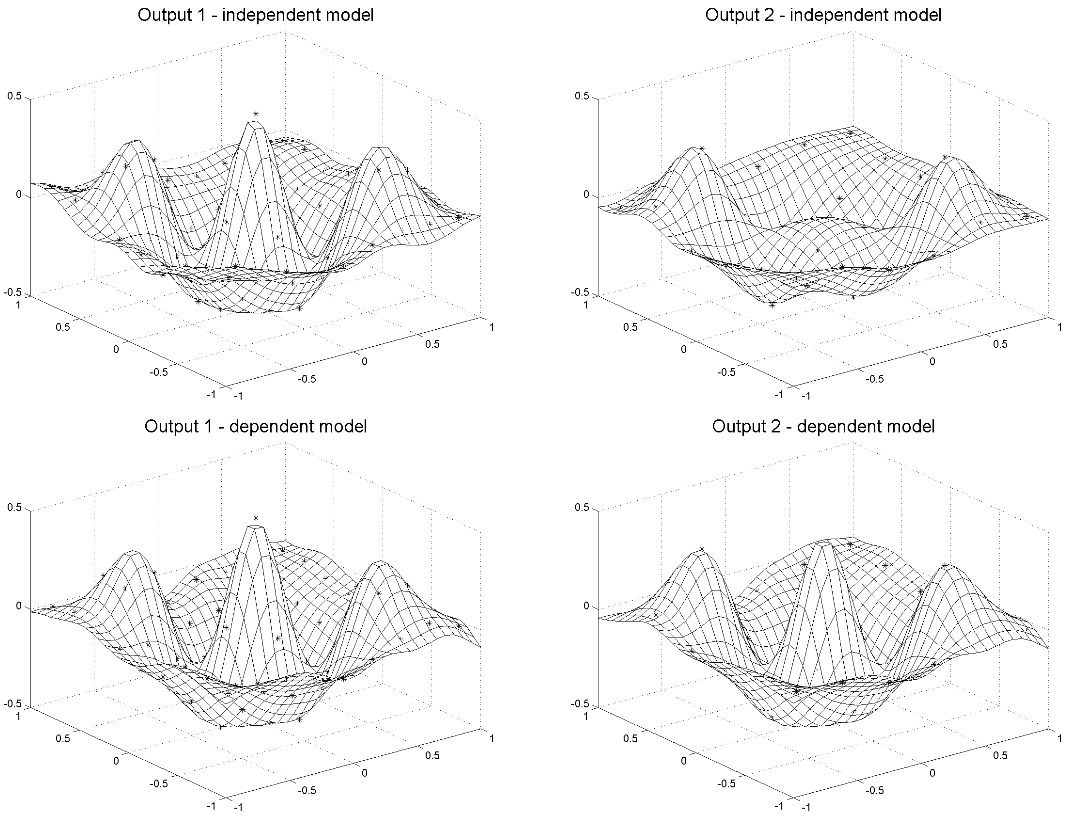

Figure 3: Strongly dependent outputs where output 2 is simply a copy of output 1, with independent Gaussian noise. *(top)* Independent model of the two outputs. *(bottom)* Dependent model. Output 1 is modelled well by both models. Output 2 is modelled well only by the dependent model

3 Time Series Forecasting

Consider the observation of multiple time series, where some of the series lead or predict the others. We simulated a set of three time series for 100 steps each (figure 5) where series 3 was positively coupled to a lagged version of series 1 ($lag = 0.5$) and negatively coupled to a lagged version of series 2 ($lag = 0.6$). Given the 300 observations, we built a dependent GP model of the three time series and compared them with independent GP models. The dependent GP model incorporated a prior belief that series 3 was coupled to series 1 and 2, with the lags unknown. The independent GP model assumed no coupling between its outputs, and consisted of three independent GP models. We queried the models for forecasts of the future 10 values of series 3. It is clear from figure 5 that the dependent GP model does a far better job at forecasting the dependent series 3. The independent model becomes inaccurate after just a few time steps into the future. This inaccuracy is expected as knowledge of series 1 and 2 is required to accurately predict series 3. The dependent GP model performs well as it has learned that series 3 is positively coupled to a lagged version of series 1 and negatively coupled to a lagged version of series 2.

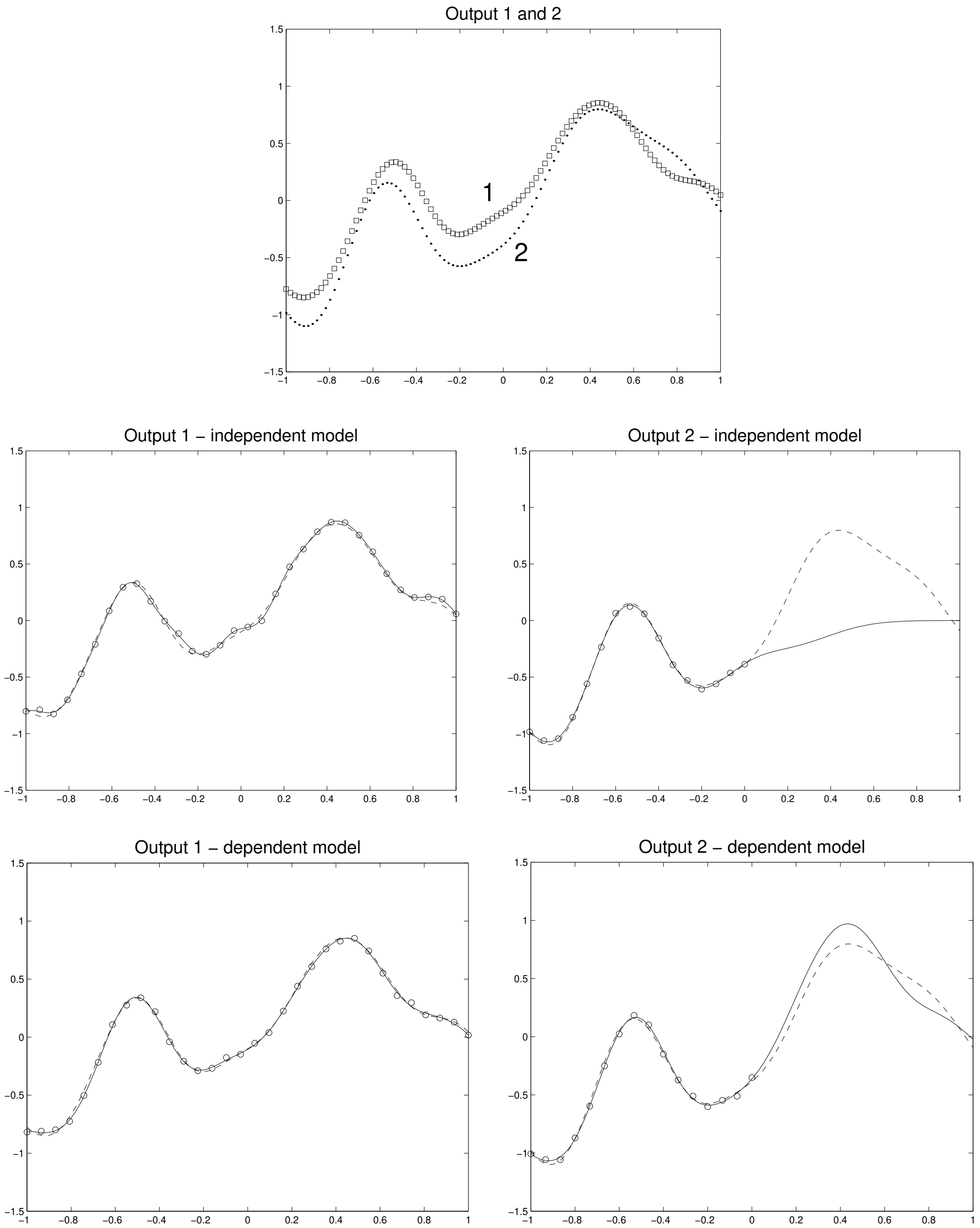

Figure 4: Dependent but unique outputs. *(top)* Outputs 1 and 2 are overlaid to illustrate the (partial) coupling between them. Note that these outputs are not entirely dependent. *(middle)* Independent model of the two outputs. *(bottom)* Dependent model.

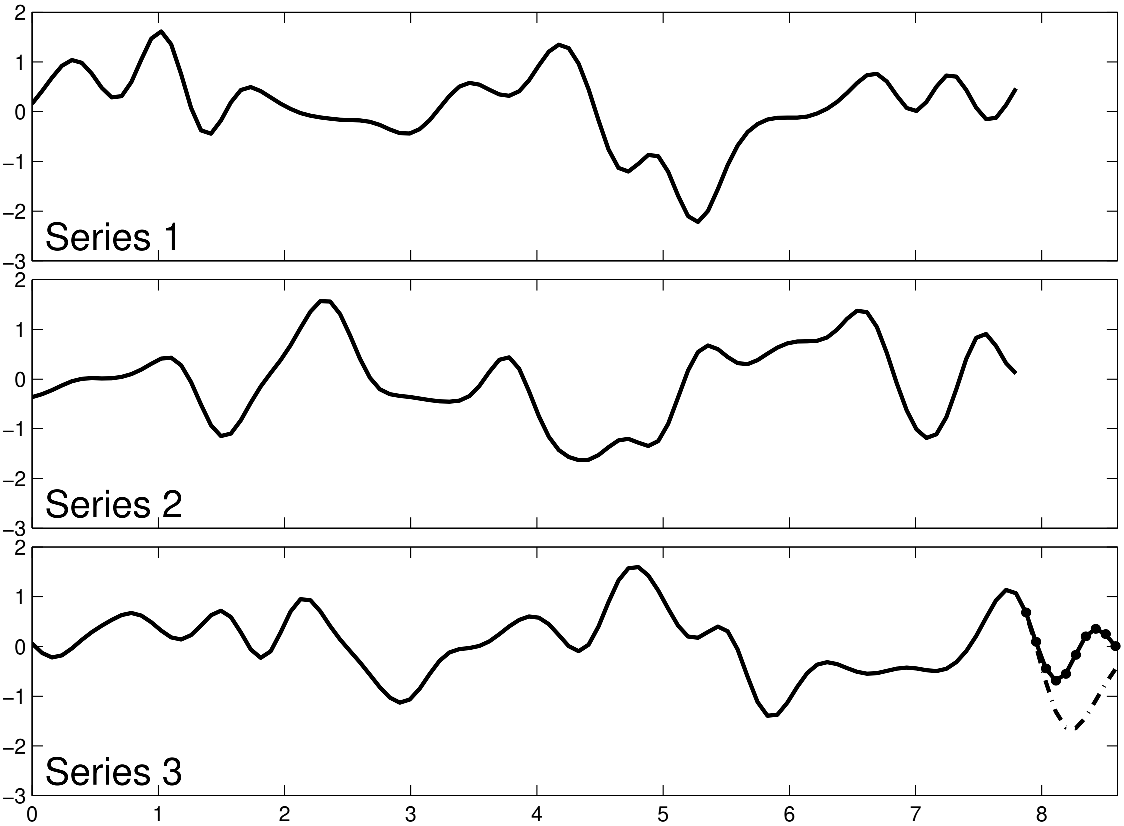

Figure 5: Three coupled time series, where series 1 and series 2 predict series 3. Forecasting for series 3 begins after 100 time steps where $t = 7.8$. The dependent model forecast is shown with a solid line, and the independent (control) forecast is shown with a broken line. The dependent model does a far better job at forecasting the next 10 steps of series 3 (black dots).

4 Multiple Outputs and Non-stationary Kernels

The convolution framework described here for constructing GPs can be extended to build models capable of modelling N-outputs, each defined over a p-dimensional input space. In general, we can define a model where we assume M-independent Gaussian white noise processes $X_1(s) \dots X_M(s)$, N-outputs $U_1(s) \dots U_N(s)$, and $M \times N$ kernels $\{\{k_{mn}(s)\}_{m=1}^{M}\}_{n=1}^{N}$ where $s \in \Re^p$. The autocovariance ($i = j$) and cross-covariance ($i \neq j$) functions between output processes i and j become (appendix A.1)

$$C_{ij}^{U}(d) = \sum_{m=1}^{M} \int_{\Re^p} k_{mi}(s) k_{mj}(s+d) ds \tag{3}$$

and the matrix defined by equation 2 is extended to

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \dots & \mathbf{C}_{1N} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{N1} & \dots & \mathbf{C}_{NN} \end{bmatrix}$$

If we have R_i observations of output i, then we have $R = \sum_{i=1}^{N} R_i$ observations in total and $\mathbf{C}$ is a $R \times R$ matrix. Our combined data set becomes $\mathcal{D} = \{\mathcal{D}_1 \cdots \mathcal{D}_N\}$, where $\mathcal{D}_i = \{(s_{i,1}\,,\, y_{i,1}) \cdots (s_{i,R_i}\,,\, y_{i,R_i})\}$.

The log-likelihood becomes

$$\begin{aligned} \mathcal{L} &= -\frac{1}{2}\log\left|\mathbf{C}\right| - \frac{1}{2}\mathbf{y}^T\mathbf{C}^{-1}\mathbf{y} - \frac{R}{2}\log 2\pi \\ \text{where} \quad \mathbf{y}^T &= \left[(y_{1,1}\cdots y_{1,R_1}) \quad \cdots \quad (y_{i,1}\cdots y_{i,R_i}) \quad \cdots \quad (y_{N,1}\cdots y_{N,R_N})\right] \end{aligned}$$

and the predictive distribution at a point s' on output i given $\mathbf{\Theta}$ and $\mathcal{D}$ is Gaussian with mean $\hat{y}'$ and variance $\sigma_{\hat{y}'}^2$ given by

$$\begin{aligned} \hat{y}' &= \mathbf{k}^T\mathbf{C}^{-1}\mathbf{y} \\ \text{and} \quad \sigma_{\hat{y}'}^2 &= \kappa - \mathbf{k}^T\mathbf{C}^{-1}\mathbf{k} \\ \text{where} \quad \kappa &= C_{ii}^{Y}(0) = v_i^2 + w_i^2 + \sigma_i^2 \\ \text{and} \quad \mathbf{k}^T &= \left[\,\mathbf{k}_1^T \cdots \mathbf{k}_j^T \cdots \mathbf{k}_N^T\,\right] \\ \text{and} \quad \mathbf{k}_j^T &= \left[C_{ij}^{Y}(s' - s_{j,1}) \dots C_{ij}^{Y}(s' - s_{j,R_j})\right] \end{aligned}$$

The kernels used in (3) need not be Gaussian, and need not be spatially invariant, or stationary. We require kernels that are absolutely integrable, $\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} |k(s)| d^p s < \infty$. This provides a large degree of flexibility, and is an easy condition to uphold. It would seem that an absolutely integrable kernel would be easier to define and parameterise than a positive definite function. On the other hand, we require a closed form of $C_{ij}^{Y}(d)$ and this may not be attainable for some non-Gaussian kernels.

5 Conclusion

We have shown how the Gaussian Process framework can be extended to multiple output variables without assuming them to be independent. Multiple processes can be handled by inferring convolution kernels instead of covariance functions. This makes it easy to construct the required positive definite covariance matrices for covarying outputs.

One application of this work is to learn the spatial translations between outputs. However the framework developed here is more general than this, as it can model data that arises from multiple sources, only some of which are shared. Our examples show the infilling of sparsely sampled regions that becomes possible in a model that permits coupling between outputs. Another application is the forecasting of dependent time series. Our example shows how learning couplings between multiple time series may aid in forecasting, particularly when the series to be forecast is dependent on previous or current values of other series.

Dependent Gaussian processes should be particularly valuable in cases where one output is expensive to sample, but covaries strongly with a second that is cheap. By inferring both the coupling and the independent aspects of the data, the cheap observations can be used as a proxy for the expensive ones.

A Appendices

A.1 Auto-Covariance and Cross-Covariance Functions

Consider M independent, stationary, Gaussian white noise processes, $X_1(s)\dots X_M(s)\,, s \in \Re^p$, producing N-outputs, $Y_1(s)\dots Y_N(s)$, with the n^{th} defined as follows:

$$Y_n(s) = U_n(s) + W_n(s)$$

where $W_n(s)$ is stationary Gaussian white noise, and $U_n(s)$ is defined by a sum of convolutions:

$$\begin{aligned}
U_n(s) &= \sum_{m=1}^{M} h_{mn}(s) \star X_m(s) \\
&= \sum_{m=1}^{M} \int_{\Re^p} h_{mn}(\alpha) X_m(s-\alpha) d^p\alpha
\end{aligned}$$

where, h_{mn} is the kernel connecting latent input m to output n.

The function $C^Y_{ij}(s_a, s_b)$ defines the auto ($i == j$) and cross covariance ($i \neq j$) between $Y_i(s_a)$ and $Y_j(s_b)$, and is derived as follows:

$$C^Y_{ij}(s_a, s_b) = C^U_{ij}(s_a, s_b) + \sigma_i^2 \delta_{ij}\,\delta_{ab}$$

where σ_i^2 is the variance of $W_i(s)$, and

$$\begin{aligned}
C^U_{ij}(s_a, s_b) &= E\left\{U_i(s_a)U_j(s_b)\right\} \qquad\qquad (U_i(s), U_j(s) \text{ are zero mean processes}) \\
&= E\left\{\sum_{m=1}^{M}\int_{\Re^p} h_{mi}(\alpha) X_m(s_a-\alpha) d^p\alpha \sum_{n=1}^{M}\int_{\Re^p} h_{nj}(\beta) X_n(s_b-\beta) d^p\beta\right\} \\
&= \sum_{m=1}^{M}\sum_{n=1}^{M}\int_{\Re^p}\int_{\Re^p} h_{mi}(\alpha) h_{nj}(\beta) E\left\{X_m(s_a-\alpha)X_n(s_b-\beta) d^p\alpha\, d^p\beta\right\}
\end{aligned}$$

where we have changed the order of expectation and integration because $\int_{\Re^p} |h_{mn}(s)|^2 d^p s < \infty \quad \forall m, n,$ i.e. $h_{mn}(s)$ are finite energy kernels (corresponding to stable linear filters).

Now, $X_m(s_1)$ and $X_m(s_2)$ are Gaussian white noise processes, that only covary if $m = n$ and $s_1 = s_2$, so

$$
\begin{aligned}
C_{ij}^{U}(s_a, s_b) &= \sum_{m=1}^{M} \int_{\Re^p} \int_{\Re^p} h_{mi}(\alpha) h_{nj}(\beta) \delta\big(\alpha - [s_a - s_b + \beta]\big) d^p\alpha \, d^p\beta \\
&= \sum_{m=1}^{M} \int_{\Re^p} h_{mj}(\beta) h_{mi}(\beta + (s_a - s_b)) d^p\beta
\end{aligned}
$$

which is the sum of kernel correlations.

If the kernels are stationary, then we can define a stationary $C_{ij}^{U}(\cdot)$ in terms of a separation vector $d_s = s_a - s_b$.

$$
C_{ij}^{U}(d_s) = \sum_{m=1}^{M} \int_{\Re^p} h_{mj}(\beta) h_{mi}(\beta + d_s) d^p\beta \tag{4}
$$

A.2 Covariance functions for Gaussian Kernels

Equation 4 defines the auto and cross covariance and is repeated here for convenience.

$$
C_{ij}^{U}(d_s) = \sum_{m=1}^{M} \int_{\Re^p} h_{mj}(\beta) h_{mi}(\beta + d_s) d^p\beta
$$

Here, we set the kernels to parameterised Gaussians and solve the integral.

Let

$$
h_{mn}(s) = v_{mn} \exp\left(-\frac{1}{2}(s - \mu_{mn})^{\mathrm{T}} \mathbf{A}_{mn} (s - \mu_{mn})\right)
$$

where $v_{mn} \in \Re$, $s, \mu_{mn} \in \Re^p$, and $\mathbf{A}_{mn}$ is a $p \times p$ positive definite matrix.

Now let $a, b \in \Re^p$ and $\mathbf{A}, \mathbf{B}$ be $p \times p$ positive definite matrices. Define

$$
\begin{aligned}
f(s, a, b, \mathbf{A}, \mathbf{B}) &= \int_{\Re^p} \exp\left(-\frac{1}{2}(s - a)^{\mathrm{T}} \mathbf{A} (s - a)\right) \exp\left(-\frac{1}{2}(s - b)^{\mathrm{T}} \mathbf{B} (s - b)\right) d^p s \\
&= \exp\left(-\frac{1}{2}c\right) \int_{\Re^p} \exp\left(-\frac{1}{2}(s - \mu)^{\mathrm{T}} \mathbf{G} (s - \mu)\right) d^p s
\end{aligned}
$$

where $\mathbf{G} = \mathbf{A} + \mathbf{B}$, $\mu = \mathbf{G}^{-1}(\mathbf{A}a + \mathbf{B}b)$, and $c = a^{\mathrm{T}}\mathbf{A}a + b^{\mathrm{T}}\mathbf{B}b - \mu^{\mathrm{T}}\mathbf{G}\mu$

$$
f(s, a, b, \mathbf{A}, \mathbf{B}) = \exp\left(-\frac{1}{2}c\right) \frac{(2\pi)^{\frac{p}{2}}}{\sqrt{|\mathbf{G}|}}
$$

Now,

$$
\begin{aligned}
c &= a^{\mathrm{T}}\mathbf{A}a + b^{\mathrm{T}}\mathbf{B}b - \mu^{\mathrm{T}}\mathbf{G}\mu \\
&= (b - \epsilon)^{\mathrm{T}} \mathbf{\Sigma} (b - \epsilon) + g
\end{aligned}
$$

where $\mathbf{\Sigma} = \mathbf{A} - \mathbf{A}\mathbf{G}^{-1}\mathbf{A} = \mathbf{A}\mathbf{G}^{-1}\mathbf{B}$ and $\epsilon = \mathbf{\Sigma}^{-1}\mathbf{B}\mathbf{G}^{-1}\mathbf{A}a = a$ and
$g = -\epsilon^{\mathrm{T}}\mathbf{\Sigma}\epsilon + a^{\mathrm{T}}\mathbf{A}a - a^{\mathrm{T}}\mathbf{A}\mathbf{G}^{-1}\mathbf{A}a = 0$ so,

$$
f(s, a, b, \mathbf{A}, \mathbf{B}) = \frac{(2\pi)^{\frac{p}{2}}}{\sqrt{|\mathbf{A} + \mathbf{B}|}} \exp\left(-\frac{1}{2}(b - a)^{\mathrm{T}} \mathbf{\Sigma} \, (b - a)\right)
$$

We can now write

$$
\begin{aligned}
C_{ij}^{U}(d_s) &= \sum_{m=1}^{M} \int_{\Re^p} \left\{ v_{mj} \exp\left(-\frac{1}{2}(\beta - \mu_{mj})^{\mathrm{T}} \mathbf{A}_{mj} (\beta - \mu_{mj}) \right) \right. \\
&\quad \left. \times \; v_{mi} \exp\left(-\frac{1}{2}(\beta + d_s - \mu_{mi})^{\mathrm{T}} \mathbf{A}_{mi} (\beta + d_s - \mu_{mi}) \right) \right\} d^p\beta \\
&= \sum_{m=1}^{M} f(\beta, \mu_{mj}, d_s - \mu_{mi}, \mathbf{A}_{mj}, \mathbf{A}_{mi}) \\
&= \sum_{m=1}^{M} \frac{(2\pi)^{\frac{p}{2}}}{\sqrt{|\mathbf{A}_{mj} + \mathbf{A}_{mi}|}} \exp\left(-\frac{1}{2}(d_s - [\mu_{mi} - \mu_{mj}])^{\mathrm{T}} \mathbf{\Sigma} \left(d_s - [\mu_{mi} - \mu_{mj}] \right) \right)
\end{aligned}
$$

where $\mathbf{\Sigma} = \mathbf{A}_{mi}(\mathbf{A}_{mi} + \mathbf{A}_{mj})^{-1}\mathbf{A}_{mj}$

References

[1] Abrahamsen, P. A review of gaussian random fields and correlation functions. Tech. Rep. 917, Norwegian Computing Center, Box 114, Blindern, N-0314 Oslo, Norway, 1997.

[2] Cressie, N. *Statistics for Spatial Data.* Wiley, 1993.

[3] Gibbs, M. *Bayesian Gaussian Processes for Classification and Regression.* PhD thesis, University of Cambridge, Cambridge, U.K., 1997.

[4] Gibbs, M., and MacKay, D. J. Efficient implementation of gaussian processes. www.inference.phy.cam.ac.uk/mackay/abstracts/gpros.html, 1996.

[5] Gibbs, M. N., and MacKay, D. J. Variational gaussian process classifiers. *IEEE Trans. on Neural Networks 11*, 6 (2000), 1458–1464.

[6] Higdon, D. Space and space-time modelling using process convolutions. In *Quantitative methods for current environmental issues* (2002), C. Anderson, V. Barnett, P. Chatwin, and A. El-Shaarawi, Eds., Springer Verlag, pp. 37–56.

[7] MacKay, D. J. Gaussian processes: A replacement for supervised neural networks? *In NIPS97 Tutorial*, 1997.

[8] MacKay, D. J. *Information theory, inference, and learning algorithms.* Cambridge University Press, 2003.

[9] Neal, R. Probabilistic inference using markov chain monte carlo methods. Tech. Report CRG-TR-93-1, Dept. of Computer Science, Univ. of Toronto, 1993.

[10] Neal, R. Monte carlo implementation of gaussian process models for bayesian regression and classification. Tech. Rep. CRG-TR-97-2, Dept. of Computer Science, Univ. of Toronto, 1997.

[11] Paciorek, C. *Nonstationary Gaussian processes for regression and spatial modelling.* PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A., 2003.

[12] Paciorek, C., and Schervish, M. Nonstationary covariance functions for gaussian process regression. *Submitted to NIPS*, 2004.

[13] Rasmussen, C., and Kuss, M. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems* (2004), vol. 16.

[14] Rasmussen, C. E. *Evaluation of Gaussian Processes and other methods for Non-Linear Regression.* PhD thesis, Graduate Department of Computer Science, University of Toronto, 1996.

[15] Tipping, M. E., and Bishop, C. M. Bayesian image super-resolution. In *Advances in Neural Information Processing Systems* (2002), S. Becker S., Thrun and K. Obermayer, Eds., vol. 15, pp. 1303 – 1310.

[16] WILLIAMS, C. K., AND BARBER, D. Bayesian classification with gaussian processes. *IEEE trans. Pattern Analysis and Machine Intelligence 20*, 12 (1998), 1342 – 1351.

[17] WILLIAMS, C. K., AND RASMUSSEN, C. E. Gaussian processes for regression. In *Advances in Neural Information Processing Systems* (1996), D. Touretzsky, M. Mozer, and M. Hasselmo, Eds., vol. 8.