Transformations of Gaussian Process Priors

Roderick Murray-Smith[1,2]
[1]Dept. Computing Science
University of Glasgow
Glasgow G12 8QQ
Scotland, UK
rod@dcs.gla.ac.uk

Barak A. Pearlmutter[2]
[2]Hamilton Institute
National Univ. of Ireland, Maynooth
Co. Kildare, Ireland
barak@cs.may.ie

Abstract

Gaussian processes-prior systems generally consist of noisy measurements of samples of the putatively Gaussian process of interest, where the samples serve to constrain the posterior estimate. Here we consider the case where the measurements are instead *noisy weighted sums* of samples. This framework incorporates measurements of derivative information and of filtered versions of the process, thereby allowing GPs to perform sensor fusion and tomography, it allows certain group invariances (ie symmetries) to be weakly enforced, can be used to model heteroskedasticity in output variance, and under certain conditions it allows the dataset to be dramatically reduced in size. The method is applied to a sparsely sampled image, where each sample is taken using a broad and non-monotonic point spread function.

1 Introduction[1]

Gaussian process priors are increasingly used as a flexible nonparametric model in a range of application areas (e.g. O'Hagan, 1978; Rasmussen, 1996; Williams, 1998; Murray-Smith and Sbarbaro, 2002). Solak et al. (2003) used the fact that the derivative of a Gaussian process is itself a Gaussian process to integrate function and derivative observations. This is particularly useful when modeling nonlinear dynamic systems. Here we generalise the results to arbitrary transformations of a Gaussian process, which in discrete form can be summarised by a linear transformation. We show four major advantages this can offer:

1. We can fuse information from multiple sensors, where the (potentially nonlinear) transformation associated with the sensor can be approximated by a linear weighting on discretisation. GP inference can then solve ill-posed inverse problems.
2. We can add 'artificial' data points which introduce prior knowledge by enforcing certain chosen linear constraints, such as symmetry, or higher-order derivative operators.
3. We can choose $n \times N$ linear transformations, where N is the number of points in the original training set, which reduce the computational complexity to $O(n^3) + O(N^2)$. For $n \ll N$ this can lead to a significant improvement in speed. We show that such mappings can be derived from smooths of less refined models.
4. In many applications we can choose a series of linear transformations which compress the training set, as above, and correspond to multi-scale learning.

[1]Technical Report TR-2003-149, Dept. of Computing Science, Glasgow University, June, 2003.

2 Transformations of Gaussian Process priors

Consider N observations of inputs X and outputs Y, where we assume the Y are drawn from an N-dimensional normal distribution,

$$Y \sim \mathcal{N}(0, \Sigma),$$

where Σ is the $N \times N$ covariance matrix, the elements of which are functions of inputs X, an $N \times d$ matrix. The covariance function is of the form

$$\text{cov}(y_i, y_j) = C(x_i, x_j; \theta) = v_0 \exp\left(-\sum_k w_k (x_{i,k} - x_{j,k})^2 \right) + \sigma_y^2,$$

where hyperparameter vector $\theta = [w_1 \ldots w_d \sigma_y]$. This covariance function reflects prior beliefs that the target function is smooth, so penalising high-frequency components. The parameter w_k reflects the length-scale of changes in input dimension k.

We will now assume that instead of observing y's directly, we observe a transformation m of the latent variables y. In the continuous case

$$\begin{aligned} \textit{output} &= \int_\Omega \textit{system} \times \textit{input}\, d\Omega, \\ m(t) &= \int K(t, x)\, y(x)\, dx \end{aligned} \tag{1}$$

which could represent a nonlinear mapping from x to m. In discrete form this becomes

$$m_k = \sum_{j=1}^{N} K_{kj}\, Y_j. \tag{2}$$

In other words, for the vector of latent Y we observe outputs $M = KY$, where K is known. This could, for example, correspond to an inverse problem such as image restoration, where the observable is the image, the system is the lens, and the scenery is the input.

The discretised form K defines a linear transformation of Y, which results in M. The vector M is therefore drawn from an n-dimensional normal distribution:

$$M \sim \mathcal{N}(0, K\Sigma K^T + \Sigma_M),$$

where Σ_M is the $n \times n$ diagonal matrix of observation variances.

If we wish to predict some M_2 given X_1, M_1, K_1, and X_2, K_2 then the conditional mean and variance are

$$\mu_{2.1} = K_2 \Sigma_{12} K_1^T (K_1 \Sigma K_1^T + \Sigma_M)^{-1} M_1 \tag{3}$$

$$\Sigma_{2.1} = \Sigma_M - K_2 \Sigma_{12} K_1^T (K_1 \Sigma K_1^T + \Sigma_M)^{-1} K_1 \Sigma_{21} K_2^T \tag{4}$$

By selecting the transformation K_2, associated with the mapping from the latent space y to the outputs at the test points x_2, we can perform inference to any of the variables chosen. If $K_2 = I$, then we are inferring y directly from observations of M_1, and implicitly solving the inverse problem of finding the conditional mean and variance of the latent variable y.

In cases where the mapping is applied to the observation variance term σ_y^2, this would provide a straighforward way of introducing heteroskedasticity into the GP model.

2.1 Learning the covariance function parameters

The log-likelihood, given the training data M_1 is

$$L = -\frac{1}{2} \log |K_1 \Sigma_1 K_1^T + \Sigma_M| - \frac{1}{2} M_1^T (K_1 \Sigma_1 K_1^T + \Sigma_M)^{-1} M_1 - \frac{N_1}{2} \log 2\pi.$$

If we wish to maximise the likelihood, we use the derivative with respect to the hyperparameters θ, where $Q = K_1 \Sigma_1 K_1^T + \Sigma_M$,

$$\frac{\partial L}{\partial \theta} = -\frac{1}{2}\text{tr}\left(Q^{-1}\frac{\partial Q}{\partial \theta}\right) + \frac{1}{2}M_1^T Q^{-1}\frac{\partial Q}{\partial \theta}Q^{-1}M_1 \quad (5)$$

and optimise the hyperparameters using an appropriate routine – we used a conjugate gradient approach, or use a Markov-Chain Monte Carlo algorithm to implement a numerical integration.

The ability to adapt the parameters of the covariance function means that the regularising effect is automatically estimated from the data, reducing the w_k of uninformative input dimensions (see discussion in Williams (1998)) – this is important in learning in general, but especially interesting for the inverse problem aspects of this paper.

If K is uncertain, then we can take a parametric model $K(t, x; \theta)$, and identify θ, or potentially use a second Gaussian process as a prior for the mapping $K(t, x)$. The covariance function and mean function can be chosen appropriately, depending on knowledge of the mapping from x, y to m.

2.2 Examples of transformations

The linear transformation K can be used to perform a number of roles:

2.2.1 Filtering the data

The K can represent filters applied to the latent variables before observation, reflecting sensor characteristics or intervening transformation of the states by other means. As noted above, the sensor characteristics described in $K(t, x)$ could be nonlinear, changing with state x, while retaining a linear transformation K on discretisation. Explicitly building the sensor characteristics into the model will tend to be better conditioned than simply pre-filtering the data with an inverse model.

2.2.2 Enforcing constraints

We can add new data points which enforce constraints, such that a weighted sum of outputs equals some constant. For example, symmetry in the y-axis for a one-dimensional function can be achieved using matrices of the form

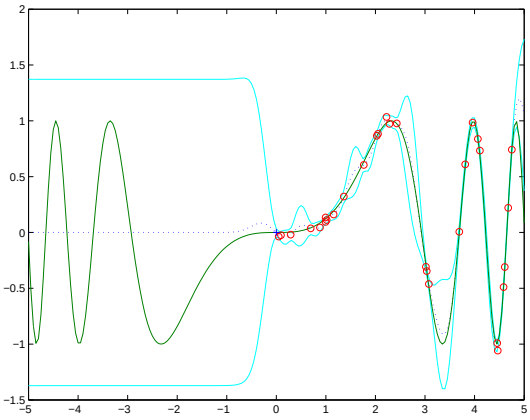
$$K_{\text{even}} = \begin{bmatrix} 1 & & & & & -1 \\ & 1 & & & -1 & \\ & & 1 & -1 & & \end{bmatrix}, \qquad K_{\text{odd}} = \begin{bmatrix} 1 & & & & & 1 \\ & 1 & & & 1 & \\ & & 1 & 1 & & \end{bmatrix}$$

for

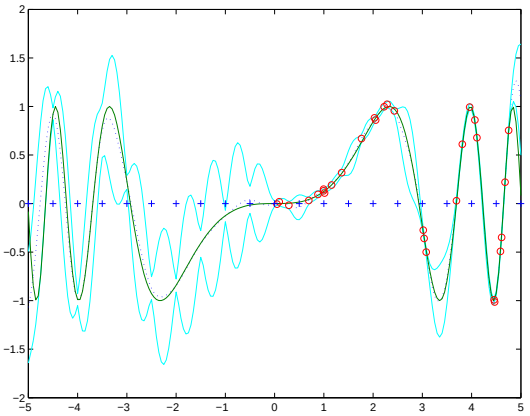
$$X = \begin{bmatrix} x_1 & x_2 & x_3 & -x_3 & -x_2 & -x_1 \end{bmatrix}^T, \quad M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T.$$

which will produce an even or odd function depending on the matrix chosen. Examples of inference with Gaussian process priors incorporating such symmetry constraints are shown in Figure 1.

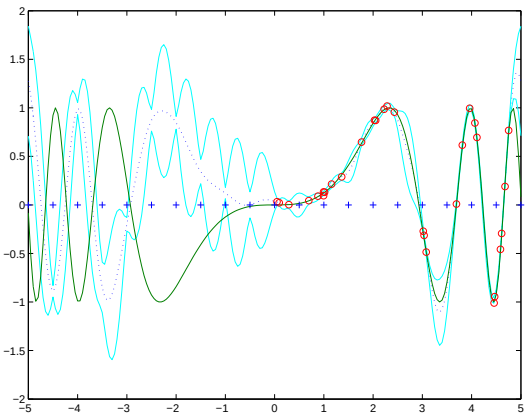
An alternative approach to enforce symmetry would be by appropriate design of the covariance function, which would be more appropriate for fully symmetric functions. The use of individual data points as constraints does have potential advantages where prior knowledge of symmetry is restricted to localised regions.



(a) No symmetry constraints



(b) Odd symmetry constraints



(c) Even symmetry constraints

Figure 1: Illustration of use of artificial data points enforcing symmetry locally. Circles are normal observed outputs, crosses are points on x-axis where symmetry constraint has been added. Model mean and 2 std. dev. contours are shown. Note that because of this sparse enforcement of symmetry the error region about the inferred symmetric portion of the curve is much looser than on the side with the data.

2.2.3 Differentiation

An example of enforcing weighted constraints is to represent derivatives. These can be implemented by differences, e.g. for first and second derivatives,

$$K' = \frac{1}{\Delta x}\begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix}, K'' = \frac{1}{\Delta x}\begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix},$$

where Δx indicates the distance between points in x. We can continue in this manner to be able to add arbitrary linear combinations of higher-order derivatives, i.e. differential forms. We can therefore add prior knowledge of combinations of derivatives of any order, by including fictitious pairs of data points (x_1, x_2), and their known derivative m, or include information from different sensors which measure different derivatives of y.

3 Fusion of multiple transformations of latent variables

In the case of an observation vector M composed of a number of vectors $M_i = K_i Y$, we have

$$M = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_k \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ \vdots \\ K_k \end{bmatrix} Y = KY.$$

We can now integrate multiple observations which might be a mixture of readings from different physical sensors, artificial data points in the form of constraints on the function, or differential operators applied to the data, to derive a model based on a latent variable y which is compatible with all of them. Such consistent integration of multiple observations, constraints and derivatives is far from trivial, as can be observed in the theoretical and practical problems associated with design and verification of gain scheduled and fuzzy controllers (Leith and Leithead, 1999).

3.1 Relevance for solving inverse problems

If the filters K_i are derived from the physics of the sensing mechanisms, does this approach give us any advantages for solving inverse problems? Standard approaches to inversion of ill-posed problems use regularisation where solution components corresponding to small singular values are filtered out. A common approach would use $Y = K^+ M = (K^T K)^{-1} K^T M$, where the inversion would be based around SVD or the Generalised SVD approach, including a filter matrix L would filter the singular values of $(K^T K)$. Specific examples of this include Tikhonov regularisation, where a regularisation operator $\Omega(Y)$, is added – minimising $||KY - M|| + \Omega(Y)$. See Hansen (1997) for a review.

In the GP case presented in this paper, the smoothness constraint is provided by the covariance function. As shown in equation (3), $Y = \Sigma_{12} K^T (K \Sigma K^T)^{-1} M$. Numerically, the inversion of $K \Sigma K^T$ should be better conditioned. Via the covariance function we effectively include estimated or prior knowledge about noise in Y and M, and correlation among elements of Y, which improve the condition number of the matrix $K \Sigma K^T$ and have a regularising effect on the solution.

3.2 Example: Reconstruction of Images from Ganglion Cell Signals

Consider a $k \times k$ pixel image measured using noisy sensors, then linearly transformed by a suite $m \ll k^2$ of on-center off-surround (see Figure 2 for the activation function as a function of distance from the centre) receptive fields prior to transmission through a noisy channel. Given the values received, along with a noise model of the channel and knowledge

of the receptive fields, we wish to estimate the original image. This reconstruction problem, intended to be reminiscent of interpretation of signals sent through the optic nerve, is shown in Figure 3, with varying levels of sparsity, $k = 41, m = 225$ and 1000 pixels in image available.

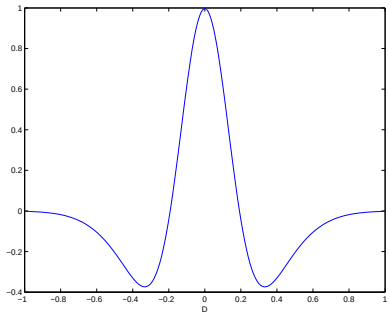


Figure 2: Activation function of centre-surround 'neurons' for distance from centre

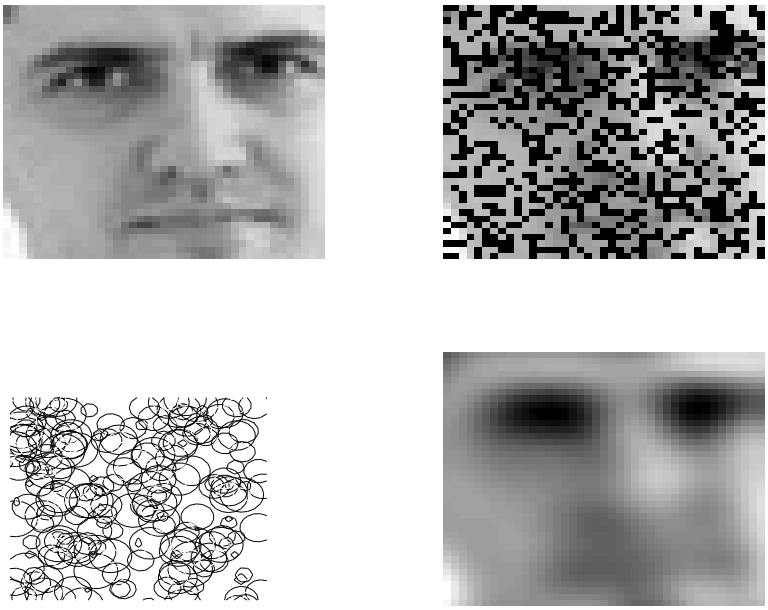


Figure 3: Example of use of GP for solving inverse problem. Source image (top left) is sparsely presented, with additive noise (top right) to neurons, and responses on output 'neurons' measured (bottom left). Inference in GP model to training data gives inferred reconstructed image (bottom right).

4 Learning with large data-sets

A major limiting factor in the acceptance of GP-prior approaches in practice is the computational effort associated with large training sets, as the complexity grows at $O(N^3)$ for a training set with N points. Attempts to overcome this include the use of the Nyström method (Williams and Seeger, 2001), selection mechanisms (Seeger et al., 2003), mixtures of GPs (Shi et al., 2002), and Bayesian committee machine (Tresp, 2000).

A key feature of the filtering approach is that the K_i need not be square matrices. In fact in many applications the filter can represent a significant reduction in the number of data

points, so K will be $n \times N$ where $n \ll N$. Note that in the equations for the inference and likelihood calculations we needed to invert $K\Sigma K^T$, which is the major computational hurdle for this method, scaling as $O(N^3)$. For nonsquare K we now need only invert an $n \times n$ matrix, as opposed to an $N \times N$. We still need to calculate the covariance values of Σ for all N points, but this is $O(N^2)$. To further increase the efficiency of the method we can eliminate points from the calculation of the covariance matrix Σ which correspond to a column of entries in $K_{i,j}\, y_j$ which are below some threshold ϵ. In such cases, the original observation y_j associated with this column has little impact on the model's predictions at the chosen test points.

To summarize, when $n \ll N$ this method results in substantially decreased computational burden because

$$\textit{complexity} = \overbrace{O(n^3)}^{\text{invert}} + \overbrace{O(N^2)}^{\text{covar}} \ll \overbrace{O(N^3)}^{\text{naive}}$$

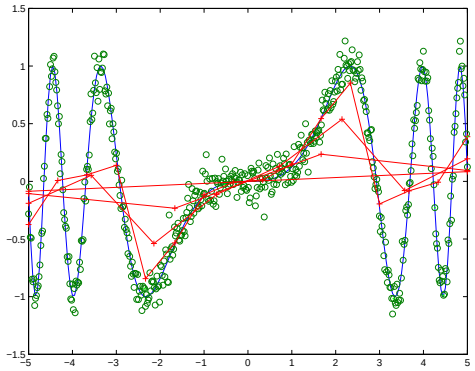
4.1 Reusing *effective kernels* from earlier models

A practical approach for finding a suitable K, with $n \ll N$, is use our prior knowledge of the problem to determine appropriate filters. An alternative is to base the filter on existing approximate models, which might be less computationally expensive to estimate. We now generalise this idea to a broader class of model – we take an existing nonlinear representation of the input–output relationship from any linear-in-the-parameters nonlinear empirical model, and at any input point of interest, we can calculate the effective kernel of the model.

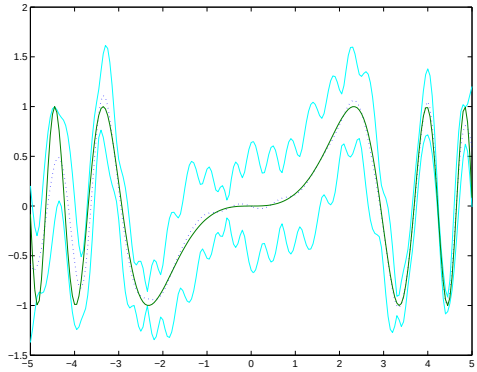
For any basis function model, such as an RBF network, spline model etc, with basis functions $\phi_i(x)$, and weighting parameters θ_i, the estimated output $\hat{y^*}$ for a test input x^* is $\hat{y^*} = \sum_i \phi_i(x^*)\hat{\theta}_i = \Phi(x^*)\hat{\theta}$, where the parameters are identified using standard approaches, e.g. $\hat{\theta} = \Phi(X)^+ Y$. We can now reinterpret the basis function model as smoothing the training outputs, $\hat{y^*} = \Phi(x^*)\Phi(X)^+ Y$, where the vector $k_* = \Phi(x^*)\Phi(X)^+$ is the *effective kernel*, a weighting of the y's in the training set for the model prediction at test point x^*. Repeating this at all points in the training set gives us the smoothing matrix $S = \Phi(X)\Phi(X)^+$. The larger the value of the entries $S_{i,j}$, the more leverage observation y_j has on the prediction of $\hat{y}_i$. We can use this effective kernel as a way of generating rows of the linear transformation matrix K to create new, filtered training data. The filter will be well-suited to the specific modelling task, and its application creates 'high-value' data points.

4.2 Example: Multi-scale learning

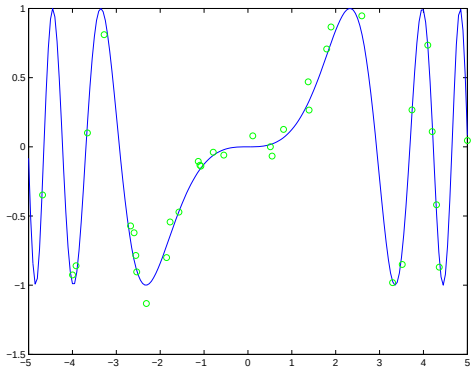
We illustrate the ability of the approach to reduce the computational complexity of learning, by creating a series of filters which include progressively more points from the training data, filtered at finer scales. In this case the filters are noncausal, triangular weighting filters, starting with two points and doubling the number of points in each filter. To show that the approach can perform well with very small training sets, which incorporate information from a larger number of training points, see Figure 4.



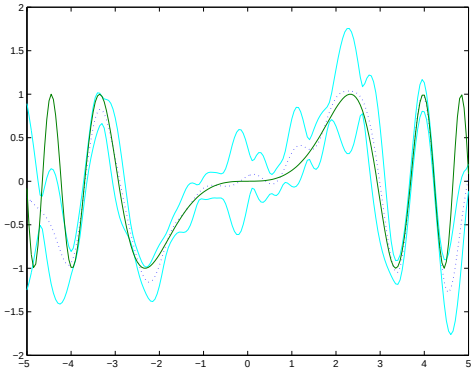
(a) Original training data (circles) and filtered subsets (crosses)



(b) GP model response compared to true function



(c) Subset of original training data (circles)



(d) GP model response compared to true function

Figure 4: Illustration of multiple-filter approach to reduce an original training set of 500 points down to 35 training points. This is significantly better than subselecting 35 of the original training points as can be seen in the lower figures.

5 Conclusions

We have demonstrated how transformations of Gaussian process priors can, for known transformations, allow us to use GPs to consistently fuse information from multiple sensors, which is of immediate practical importance in many engineering applications. We also demonstrate the use of GPs to solve ill-posed inverse problems. The amount of noise on both latent variables and observed variables, and the amount of regularisation required in the inversion process, are automatically optimised during adaptation of the model covariance hyperparameters.[2] More detailed comparison of these benefits with the algorithms currently used in the inverse-problems community is required.

The incorporation of 'artificial' data points is a novel way to introduce prior knowledge by enforcing certain chosen linear constraints, such as symmetry, or higher-order derivative operators, which is easy to use, and has application in a range of areas. The reduction in the computational complexity to $O(n^3) + O(N^2)$ for GP's may also be very significant in

[2]Optimised – as far as the choice of prior made is appropriate for a given task.

broadening the application base of GP inference, and there is great scope for extension of the methods to create interesting multi-scale learning algorithms, and for stepwise optimisation or integration of covariance hyperparameters.

Acknowledgements

The authors gratefully acknowledge the support of the *Multi-Agent Control* Research Training Network by EC TMR grant HPRN-CT-1999-00107, support from EPSRC grant *Modern statistical approaches to off-equilibrium modelling for nonlinear system control* GR/M76379/01, support from EPSRC grant GR/R15863/01, and Science Foundation Ireland grant 00/PI.1/C067.

References

Hansen, P. C. (1997). *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM. SIAM Monographs on Mathematical Modeling and Computation 4.

Leith, D. and Leithead, W. (1999). Analytic framework for blended multiple model systems using linear local models. *International Journal of Control*, 72(7/8):605–619.

Murray-Smith, R. and Sbarbaro, D. (2002). Nonlinear adaptive control using non-parametric Gaussian process prior models. In *15th IFAC World Congress on Automatic Control, Barcelona*.

O'Hagan, A. (1978). On curve fitting and optimal design for regression (with discussion). *Journal of the Royal Statistical Society B*, 40:1–42.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression*. PhD thesis, Graduate department of Computer Science, University of Toronto.

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

Shi, J. Q., Murray-Smith, R., and Titterington, D. M. (2002). Hierarchical Gaussian process mixtures for regression. Technical Report TR-2002-107, University of Glasgow, Scotland, UK.

Solak, E., Murray-Smith, R., Leithead, W. E., Leith, D. J., and Rasmussen, C. E. (2003). Derivative observations in Gaussian process models of dynamic systems. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information processing Systems 15*.

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

Williams, C. K. I. (1998). Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Jordan, M. I., editor, *Learning and Inference in Graphical Models*, pages 599–621. Kluwer.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Diettrich, V. T., editor, *Advances in Neural Information Processing Systems 13, MIT Press*.