Transformations of Gaussian Process Priors

Roderick Murray-Smith[12] and Barak A. Pearlmutter[2]

[1] Department of Computing Science, University of Glasgow, Scotland
[2] Hamilton Institute, NUI Maynooth, Co. Kildare, Ireland
rod@dcs.gla.ac.uk
barak@cs.nuim.ie

Abstract. Gaussian process 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; allows certain group invariances (ie symmetries) to be weakly enforced; and under certain conditions suitable application 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. It is also applied to nonlinear dynamic system identification applications where a nonlinear function is followed by a known linear dynamic system, and where observed data can be a mixture of irregularly sampled higher derivatives of the signal of interest.

1 Introduction

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, 1998b; Murray-Smith and Sbarbaro, 2002). In (Solak et al., 2003) we 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. Like the 'generalised observations' obtained from bounded linear functionals introduced in (Wahba, 1990). We show four major practical 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.

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 110–123, 2005.

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.

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

$$\mathrm{cov}(x_i, x_j) = v_0 \exp\Big(-\sum_k w_k (x_{i,k} - x_{j,k})^2\Big) + \sigma_y^2,$$

and 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 in discrete form is

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

In other words, for the vector of latents 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. Note that although the discretised form K is a linear transformation, the original kernel $K(t,x)$ could represent a nonlinear mapping.

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 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

$$\begin{aligned} \mu_{2|1} &= K_2 \Sigma_{12} K_1^T (K_1 \Sigma K_1^T)^{-1} M_1 \\ \Sigma_{2|1} &= \Sigma_2 - K_2 \Sigma_{12} K_1^T (K_1 \Sigma K_1^T)^{-1} K_1 \Sigma_{21} K_2 \end{aligned} \tag{3}$$

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.

2.1 Learning the Covariance Function Parameters

The log-likelihood, given the training data M_1 is

$$L = -\tfrac{1}{2}\log|K_1\Sigma_1K_1^T| - \tfrac{1}{2}M_1^T(K_1\Sigma_1K_1^T)^{-1}M_1 - \tfrac{1}{2}N_1\log 2\pi.$$

If we wish to maximise the likelihood, we use the derivative with respect to the hyperparameters θ,

$$\begin{aligned}\frac{\partial L}{\partial\theta} &= -\tfrac{1}{2}\operatorname{tr}\Big((K_1\Sigma_1K_1^T)^{-1}\frac{\partial(K_1\Sigma_1K_1^T)}{\partial\theta}\Big)\\ &\quad + \tfrac{1}{2}M_1^T(K_1\Sigma_1K_1^T)^{-1}\frac{\partial(K_1\Sigma_1K_1^T)}{\partial\theta}(K_1\Sigma_1K_1^T)^{-1}M_1 \qquad (4)\\ &= -\tfrac{1}{2}\operatorname{tr}\Big(K_1\frac{\partial\Sigma_1}{\partial\theta}\Big) + \tfrac{1}{2}M_1^T(K_1\Sigma_1K_1^T)^{-1}\frac{\partial(K_1\Sigma_1K_1^T)}{\partial\theta}(K_1\Sigma_1K_1^T)^{-1}M_1\end{aligned}$$

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, 1998b)—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:

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.

Enforcing Constraints. We can add new data points which enforce constraints, such that a weighted sum of outputs equals some constant. For example, symmetry 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

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

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.

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

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

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 fictive 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

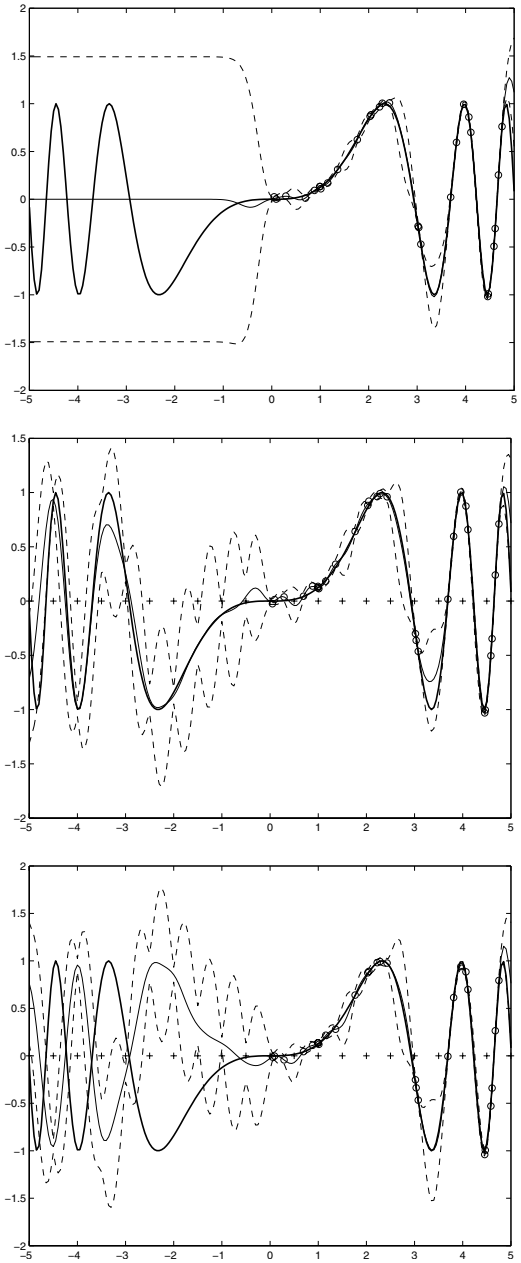


Fig. 1. Artificial data points used to locally enforce symmetry. Top: no symmetry constraints. Centre: odd symmetry constraints. Bottom: even symmetry constraint. Circles are normal observed outputs, crosses are points on x-axis where symmetry constraint has been added. Plots show model mean $\pm$ 2σ as thin solid line and dashed contours. Note that due to the sparse enforcement of symmetry, the error region about the inferred symmetric portion of the curve is looser than on the side with the data.

$Y = K^+M = (K^TK)^{-1}K^TM$, 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. 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

Tipping and Bishop (2002) presented a Gaussian process approach to super-resolution in images, with uniform sampling from a series of low-resolution images. Here we 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 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 2, with varying levels of sparsity, $k = 41, m = 625$ and 1009 pixels in image (60% of the original pixels) available.

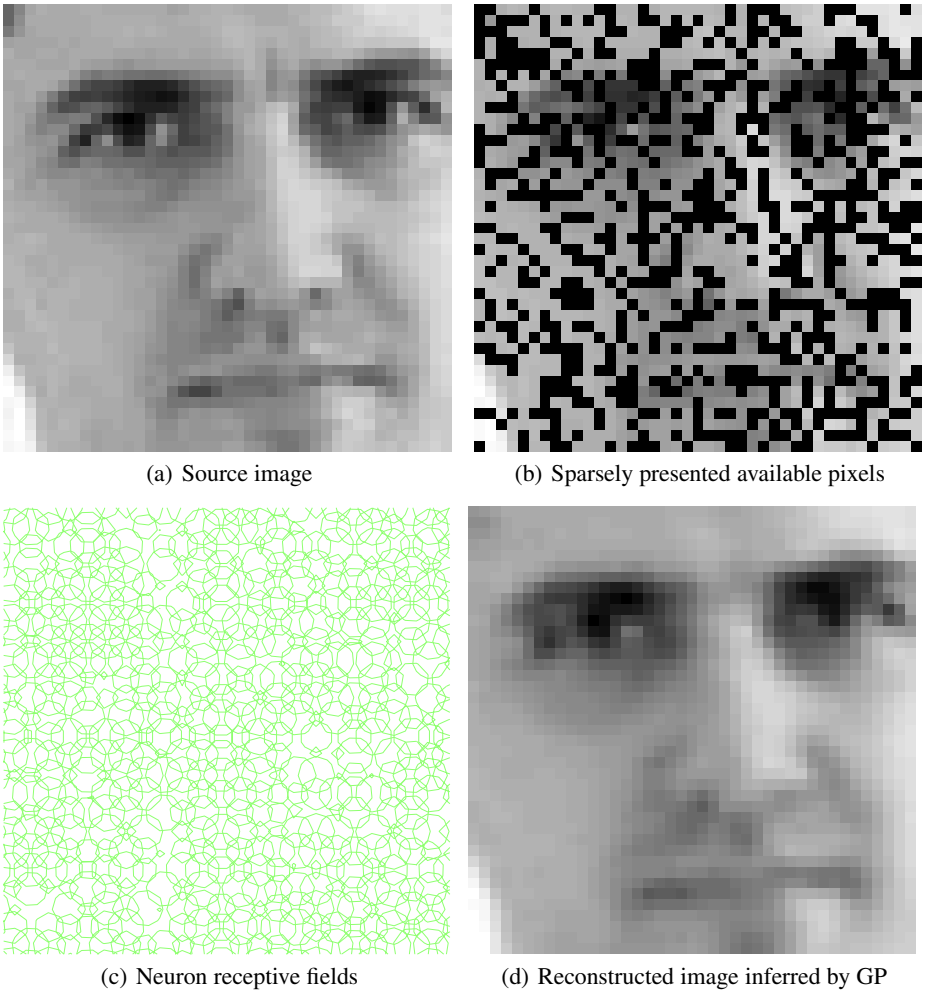
Inspection of natural images such as that shown in Figure 2 suggests that the use of a stationary covariance function is inappropriate. Instead we use a nonstationary one,

$$\mathrm{cov}(x_i, x_j) = v_0 \sin^{-1} \frac{\mathbf{x}_i^T \Sigma \mathbf{x}_j}{\sqrt{(1 + 2\mathbf{x}_i^T \Sigma \mathbf{x}_i)(1 + \mathbf{x}_j^T \Sigma \mathbf{x}_j)}} + \delta_{i,j}\sigma_y^2 \tag{5}$$

as described in Williams (1998a), using Rasmussen's MATLAB implementation.[1] This covariance function corresponds to that of a GP describing a single hidden-layer neural network of sigmoidal neurons with infinite neurons (Williams, 1998a). The Σ is a diagonal matrix with positive entries, weighting each input (and an additional constant one acting as a bias term).

The idea can be extended to colour images. The 'ganglion' cells are now made to be receptive to one colour only, with the allocation of cells to colours done randomly. A further area of interest is that we can use the GP to learn covariances between colours in a natural image, such that if we interpolate between observations we are less likely to generate spurious artifacts. We added an identifier to the inputs, indicating whether the pixel was red (1), green (0) or blue (-1). This was compared to the result of training three independent Gaussian process prior models on the red, green and blue components of the image, independently. Informal inspection of a number of test images showed more frequent colour artefacts in the independent GPs model, than in the dependent one.

[1] `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/gp/`



(a) Source image (b) Sparsely presented available pixels

(c) Neuron receptive fields (d) Reconstructed image inferred by GP

Fig. 2. Inverse problem solved using a GP. 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 Dynamic Systems Applications

In many applications we will have a learning task which involves identifying a nonlinear subsystem $f(x)$, where we do not have direct access to the outputs y of that subsystem, but to a transformation of them through another dynamic system $m = g(y, z)$, where z is the internal state of $g()$. This transformation may be another subsystem or it could represent the sensor dynamics. We assume that the dynamics of $g(y, z)$ are known and investigate identification of $f(x)$ from observed m and x.

As an example we simulate a system with

$$\begin{aligned} y(t) &= f(x(t)) = 0.3\,x(t)^3 + \sin(5\,x(t)) + \mathcal{N}(0, 0.05) \\ m(t) &= g(y, z) \end{aligned} \tag{6}$$

where there is an unknown initial condition $z(0) = z_0$. The simple discrete-time state-space system $g(y, z)$ has an output matrix C.

$$\begin{aligned} z(t+1) &= Az(t) + By(t) \\ m(t) &= Cz(t) \end{aligned} \tag{7}$$

$$M = C \begin{bmatrix} B & & & & & \\ AB & B & & & & \\ A^2B & AB & B & & & \\ A^3B & A^2B & AB & B & & \\ & & & & \ddots & \\ A^{N-1}B & A^{N-2}B & \cdots & A^2B & AB & B \end{bmatrix} Y + \begin{bmatrix} A \\ A^2 \\ A^3 \\ A^4 \\ \vdots \\ A^N \end{bmatrix} z_0, \tag{8}$$

where $M = \begin{bmatrix} m(1) \cdots m(N) \end{bmatrix}^T$, $Y = \begin{bmatrix} y(1) \cdots y(N) \end{bmatrix}^T$. If we subtract the contribution of the initial condition z_0 from M, we have an effective filter matrix K_d

$$K = C \begin{bmatrix} B & & & & & \\ AB & B & & & & \\ A^2B & AB & B & & & \\ A^3B & A^2B & AB & B & & \\ & & & & \ddots & \\ A^{N-1}B & A^{N-2}B & \cdots & A^2B & AB & B \end{bmatrix} \tag{9}$$

and we can apply the filtered Gaussian process approach in a straightforward manner.

4.1 Observing Multiple Derivatives of Time-Series

In many practical engineering applications observations are made with multiple sensors which measure, e.g. position, velocity and acceleration of state variables of interest. These sensors will often have different noise characteristics and different sampling rates. We would like to be able to combine these variables in a consistent manner. In Solak et al. (2003) we presented methods which perform inference on derivatives with respect to the inputs by analytically differentiating the covariance function. The approach described in this paper allows us to take derivatives with respect to variables not used by the covariance function.

As a simple example, we can infer the distribution of position from observed higher derivatives, and occasional noisy observations of position. Using the filters described in section 2.2,

$$K_v = \frac{1}{\Delta t_v} \begin{bmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \\ & & & -1\ 1 \end{bmatrix}, K_a = \frac{1}{\Delta t_a} \begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2\ 1 \end{bmatrix}, \tag{10}$$

where Δt_v and Δt_a are the sampling times of the velocity and acceleration signals. Similarly using the transposes K_v^T or K_a^T would allow us to infer a Gaussian process model of acceleration or velocity signals from position observations. This approach also guarantees an internally consistent set of higher derivatives, and could therefore be used as pre-processing technique for system identification and analysis tasks.

This can be combined with other filters associated with the system dynamics as described in equation (9),

$$K = K_d \left[I_p \; K_v \; K_a \right] \tag{11}$$

where I_p is the identify matrix of size p for p position observations. General differential operators could be composed of weighted combinations of the basic matrices, $K = w_1 K_v + w_2 K_a$. This lets us use Gaussian processes as an alternative to the Functional Data Analysis methods suggested by Ramsay and Silverman (1997) for inferring higher derivatives of a function without numerical problems.

4.2 Simulation of Dynamic System

Here we generate a time-series of $N = 1000$ points from a second order, discrete-time, linear system, following a nonlinear transformation of the inputs x, as described in equations (6) and (7), with $A = \begin{bmatrix} 1 & 1-\exp(-T_s) \\ 0 & \exp(-T_s) \end{bmatrix}$, $B = \begin{bmatrix} T_s - 1 + \exp(-T_s) \\ 1 - \exp(-T_s) \end{bmatrix}$, $z_0 = 0$, and $T_s = 0.1\,\mathrm{s}$ is the sample time. Furthermore, in the simulation we include a velocity 'sensor', i.e. the observations also include observations of dz/dt. The observations were subsampled to every 20th position observation, and every fourth velocity observation.

In Figure 3 we show the training data inputs x, the unobserved outputs y, and the observed filtered outputs m. The observed outputs are corrupted by white noise $\mathcal{N}(0, 0.001)$.

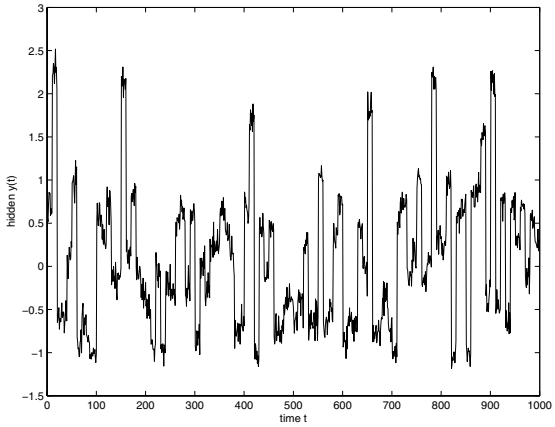
In Figure 4(b) we show the estimate of $f(x)$ over the range of interest, along with the true function, and in Figure 4(a) the estimate of the hidden states y, compared to the actual values. To show the accuracy of the velocity predictions, Figure 5 compares the inferred, observed and true velocities, and Figure 4(a) shows the estimate of the hidden states y, compared to the actual values. To show the accuracy of the velocity predictions, Figure 5 compares the inferred, observed, and true velocities.

5 Discussion

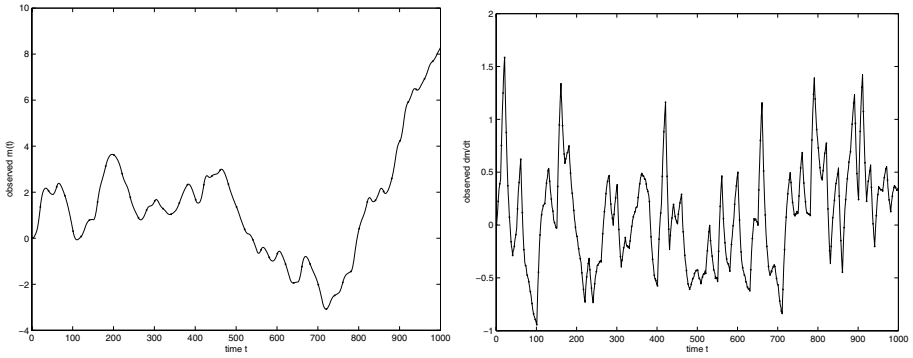
5.1 Learning with Large Data-Sets

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



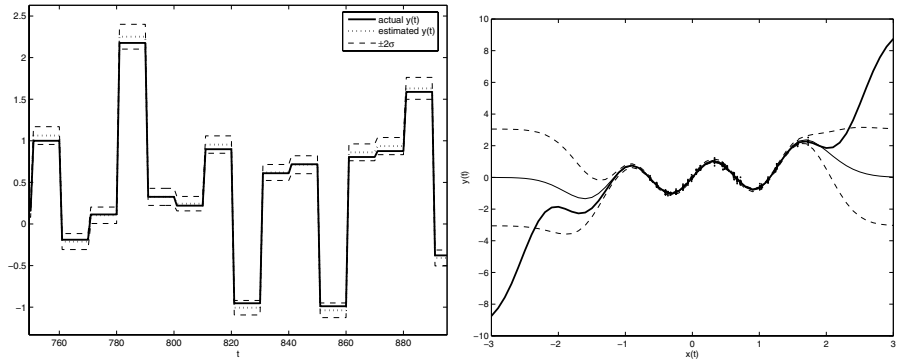
(a) Time-series of the hidden y time-series associated with the training data.



(b) Time-series of observations used by the GP to infer the nonlinear function associated with the hidden y time-series in (a).

Fig. 3. Time-series of observed and hidden states from the simulation used to generate the training data

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. In (Shi et al., 2005) we used a Karhunen-Loeve expansion to choose a subset of points. In (Solak et al., 2003) we compressed large amounts of observed data close to equilibria into local linear models, and used these very few parameters as estimated derivative observations.



(a) Comparison of hidden and inferred y as time-series

(b) Learned & true nonlinearities. (Thick line is true function, thin line mean prediction, and dashed lines ± 2 std. deviation contours.)

Fig. 4. Learning the unknown nonlinearity. The identified nonlinear function, compared to the ideal function $y(t) = 0.3\,x(t)^3 + \sin(5\,x(t))$, with $\pm 2\sigma$ contours. The actual value of y at training data is indicated by the points plotted in Figure 4(b), but the GP did not have access to this information

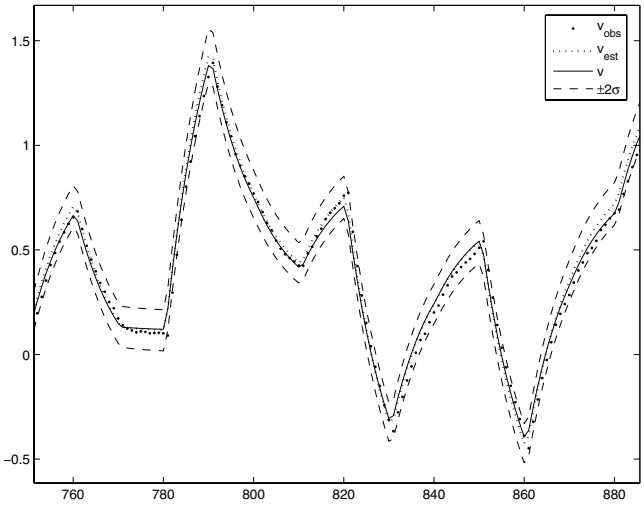


Fig. 5. The GP-inferred, observed (including effects of additive noise on the y's, and true (noise-free) velocities for a segment of the training time-series

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

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

Bias and Variance. In most machine learning approaches to data-dependent control of model complexity, the size of the model is reduced in order to trade off model bias against model variance. For instance, it is common to fit a large neural network to a large corpus of data, and then prune the network gradually removing weights. As weights are removed, the computational burden is reduced; the variance decreases; and the bias increases. This results in a tradeoff between bias and variance, meaning there is a point of best generalisation after some amount of pruning. With less than this amount of pruning, it is possible to both improve generalisation performance and reduce computational burden by further pruning. Below it, there is a tradeoff between generalisation and efficiency.

Here we have a superficially similar situation, in which exemplars can be removed from the data. As they are gradually removed, the computational burden is reduced, but the bias and variance *both* increase. As a result, the tradeoff is solely between computational burden and generalisation performance, with no need to find the point of best generalisation. Furthermore, since we would expect the exemplars removed first to contribute least to generalisation, we might expect dramatic computational savings at a very modest cost in generalisation during the initial phases.

Reusing *Effective Kernels* from Earlier Models. A practical approach for finding a suitable K, with $n \ll N$, is the use of 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 from weighted combinations of the observed data. This might provide a useful way to bring Gaussian Processes to the attention of a broader user base, as modellers who currently use a linear-in-the-parameters model could 'wrap' a Gaussian process around their current best model, getting potentially better lower model bias, and the benefit of conditional variance estimates.

5.2 Differential Forms and Ease of Implementation

The above derivations assume that measurements correspond to the outputs of known linear filters applied to the underlying function, and that these linear filters are sim-

ply weighted sums. The derivation is reasonably straightforward to extend to a broader class of linear filters corresponding to weighted sums of not just the function itself but also its derivatives, including potentially higher-order derivatives. This would combine the derivation above with that of Solak et al. (2003) yielding a theory that treats both as special cases. However in practice, as we have seen above, it is simple to approximate derivatives simple weighted sums of nearby points, which fits naturally into the framework here. The main advantage of this arguably less elegant approach to handling derivatives is three-fold. First, it avoids the cumbersome notational complexity required for referring to derivatives as well as the corresponding additional matrices. Second, it allows standard GP tools to be easily pressed into service for data of this type, making the approach more accessible to the casual practitioner. And thirdly, it makes it easy for the exploratory practitioner to constrain their models *locally* by introducing virtual data points representing some constraint.

6 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. 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 significant in 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.

The application of the method to a toy dynamic system indicates the promise this approach has to real-world system identification tasks where a number of different sensors, each with its own dynamics and noise level, need to be integrated.

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.

Ramsay, J. O. and Silverman, B. W. (1997). *Functional Data Analysis*. Springer-Verlag.

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., Murray-Smith, R., Titterington, D., and Pearlmutter, B. (2005). Learning with large data sets using filtered gaussian process priors. In Murray-Smith, R. and Shorten, R., editors, *Proceedings of the Hamilton Summer School on Switching and Learning in Feedback systems*, volume 3355 of *Lecture Notes in Computing Science*, pages 128–139. Springer-Verlag.

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 S. Becker, S. T. and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1033–1040. MIT Press, Cambridge, MA.

Tipping, M. and Bishop, C. M. (2002). Bayesian image super-resolution. In Becker, S., Thrun, S., and Obermeyer, K., editors, *Neural Information Processing Systems*, volume 12, pages 1303–1310.

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

Wahba, G. (1990). Spline models for observation data. In *Regional Conference Series in Applied Mathematics*, Philadelphia, PA. SIAM.

Williams, C. K. I. (1998a). Computation with infinite neural networks. *Neural Computation*, 10:1203–1216.

Williams, C. K. I. (1998b). 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*.