Filtered Gaussian Processes for Learning with Large Data-Sets

Jian Qing Shi[1], Roderick Murray-Smith[2,3], D. Mike Titterington[4], and Barak A. Pearlmutter[3]

[1] School of Mathematics and Statistics, University of Newcastle, UK
j.q.shi@ncl.ac.uk
[2] Department of Computing Science, University of Glasgow, Scotland
rod@dcs.gla.ac.uk
[3] Hamilton Institute, NUI Maynooth, Co. Kildare, Ireland
barak@cs.may.ie
[4] Department of Statistics, University of Glasgow, Scotland
mike@stats.gla.ac.uk

Abstract. Kernel-based non-parametric models have been applied widely over recent years. However, the associated computational complexity imposes limitations on the applicability of those methods to problems with large data-sets. In this paper we develop a filtering approach based on a Gaussian process regression model. The idea is to generate a small-dimensional set of filtered data that keeps a high proportion of the information contained in the original large data-set. Model learning and prediction are based on the filtered data, thereby decreasing the computational burden dramatically.
Keywords: Filtering transformation, Gaussian process regression model, Karhunen-Loeve expansion, Kernel-based non-parametric models, Principal component analysis.

1 Introduction

Kernel-based non-parametric models such as Splines (Wahba, 1990), Support Vector Machines (Vapnik, 1995) and Gaussian process regression models (see for example O'Hagan (1978), and Williams and Rasmussen, (1996)) have become very popular in recent years. A major limiting factor with such methods is the computational effort associated with dealing with large training data-sets, as the complexity grows at rate $O(N^3)$, where N is the number of observations in the training set. A number of methods have been developed to overcome this problem. So far as the Gaussian process (GP) regression model is concerned, such methods include the use of mixtures of GPs (Shi, Murray-Smith and Titterington, 2002) for a large data-set with repeated measurements, and the use of approximation methods such as the Subset of Regressors method (Poggio and Girosi, 1990; Luo and Wahba, 1997), the iterative Lanczos method (Gibbs and Mackay, 1997), the Bayesian Committee Machine (Tresp, 2000), the Nyström Method (Williams and Seeger, 2001) and Selection Mechanisms (Seeger *et al.*, 2003).

R. Murray-Smith, R. Shorten (Eds.): Switching and Learning, LNCS 3355, pp. 128–139, 2005.

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. In Murray-Smith and Pearlmutter (2003), the case was considered where the measurements are instead noisy weighted sums of samples. Adapting the idea of the transformation of GPs described in Murray-Smith and Pearlmutter (2003), we describe here a specific filtering approach to deal with the modelling of large data-sets. The approach involves two stages. In the first stage, a set of filtered data of dimension n is generated, where usually $n \ll N$, the dimension of the original training data-set. The value of n can be selected such that the filtered data can represent a proportion of the information of the original whole data-set. This therefore amounts to a question of experiment design, involving specification of how to design physical filters to generate filtered data. In the second stage, we carry out model learning and prediction based on the filtered data. The approach is also extended to online learning where data arrive sequentially and training must be performed sequentially as well.

The paper is organized as follows. Section 2 discusses the details of the filtering approach. We first discuss an orthogonal expansion of a kernel covariance function of a GP model based on its eigenfunctions and eigenvalues in Section 2.1. Using the results, we develop a filtering approach, the details of which are given in Section 2.2. Section 2.3 discusses statistical inference based on the filtered data, including model learning and prediction. Section 2.4 extends the approach to online learning. A simulation study is given in Section 3 to illustrate the performance of the method, and some discussion is given in Section 4.

2 Filtering Approach for Large Data-Sets

2.1 Expansion of a Gaussian Process and Its Transformations

Consider a Gaussian process $y(\boldsymbol{x})$, which has a normal distribution with zero mean and kernel covariance function $k(\boldsymbol{x}, \boldsymbol{u})$, where $\boldsymbol{x}$ is a vector of input variables. The related observation is $t(\boldsymbol{x}) = y(\boldsymbol{x}) + \epsilon(\boldsymbol{x})$, where $\epsilon(\boldsymbol{x}) \sim N(0, \sigma^2)$ and $\epsilon(\boldsymbol{x})$'s for different $\boldsymbol{x}$'s are assumed independent. The Gaussian process $y(\boldsymbol{x})$ can be decomposed, according to the Karhunen-Loève orthogonal expansion, as

$$y(\boldsymbol{x}) = \sum_{i=1}^{\infty} \phi_i(\boldsymbol{x})\xi_i, \tag{1}$$

and the covariance kernel function $k(\boldsymbol{x}, \boldsymbol{u})$ can be expanded as

$$k(\boldsymbol{x}, \boldsymbol{u}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{x})\phi_i(\boldsymbol{u}), \tag{2}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ denote the eigenvalues and $\phi_1, \phi_2, \cdots$ are the related eigenfunctions of the operator whose kernel is $k(\boldsymbol{x}, \boldsymbol{u})$, so that

$$\int k(\boldsymbol{u}, \boldsymbol{x})\phi_i(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \lambda_i \phi_i(\boldsymbol{u}), \tag{3}$$

where $p(\boldsymbol{x})$ is the density function of the input vector $\boldsymbol{x}$. The eigenfunctions are p-orthogonal, i.e.

$$\int \phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \delta_{ij}.$$

In (1) ξ_i is given by

$$\xi_i = \int \phi_i(\boldsymbol{x})y(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}. \tag{4}$$

Given a random sample $\{\boldsymbol{x}_i, i = 1, \cdots, N\}$ of inputs, independent and identically distributed according to $p(\boldsymbol{x})$, we have the discrete form of $y(\boldsymbol{x})$; that is, $\boldsymbol{Y}' = (y(\boldsymbol{x}_1), \cdots, y(\boldsymbol{x}_N))$. From (2), the covariance kernel $k(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{\theta})$ can be expanded into a feature space of dimension N as

$$k(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \lambda_i \phi_i(\boldsymbol{x})\phi_i(\boldsymbol{u}), \tag{5}$$

where $\boldsymbol{\theta}$ is a vector of unknown parameters of interest. Typically N is very large, so that the above expansion is a good approximation to (2). The discrete form of (4) is

$$\xi_i \approx \frac{1}{N} \sum_{j=1}^{N} \phi_i(\boldsymbol{x}_j)y(\boldsymbol{x}_j). \tag{6}$$

Let $\boldsymbol{\Sigma}^{(N)}$ be the covariance matrix of $\boldsymbol{Y}$, $\lambda_i^{(N)}$ be an eigenvalue of $\boldsymbol{\Sigma}^{(N)}$ and $\boldsymbol{\phi}_i^{(N)}$ be the related N-dimensional eigenvector, where $\lambda_1^{(N)} \geq \lambda_2^{(N)} \geq \cdots \geq 0$. Then $(\phi_i(\boldsymbol{x}_1), \cdots, \phi_i(\boldsymbol{x}_N)) \approx \sqrt{N}\boldsymbol{\phi}_i^{(N)}$ and $\lambda_i \approx \lambda_i^{(N)}/N$ for $i = 1, \cdots, N$; for details see Williams and Seeger (2001).

We will now assume that instead of observing the $\boldsymbol{Y}$'s directly, we observe a transformation z of the latent vector $\boldsymbol{Y}$, given by

$$z_k = \sum_{j=1}^{N} \boldsymbol{K}_{kj} y(\boldsymbol{x}_j) = \boldsymbol{K}_k \boldsymbol{Y} \tag{7}$$

for $k = 1, \cdots, n$. In other words, for the vector of latent variables $\boldsymbol{Y}$ we observe outputs $\boldsymbol{Z} = \boldsymbol{K}\boldsymbol{Y}$, where $\boldsymbol{K}$ is an $n \times N$ known matrix and $\boldsymbol{Z}^T = (z_1, \cdots, z_n)$. The above transformations define n data filters, and usually $n \ll N$. Each of n physical filters can be designed by the values of each row of $\boldsymbol{K}$.

A special case corresponds to constructing $\boldsymbol{K}$ from the first n eigenvectors of $\boldsymbol{\Sigma}^{(N)}$. When the kth row of $\boldsymbol{K}$ consists of the eigenvector $\boldsymbol{\phi}_k^{(N)}$, z_k is calculated by (7). Comparing (7) with (6), we have that $\xi_k \approx z_k/\sqrt{N}$. The n filtered observations z correspond to the n largest eigenvalues. Therefore, if we use the n-dimensional filtered data, we approximate the covariance kernel in (5) by

$$k(\boldsymbol{x}, \boldsymbol{u}) \approx \sum_{i=1}^{n} \lambda_i \phi_i(\boldsymbol{x})\phi_i(\boldsymbol{u}). \tag{8}$$

Then the subspace spanned by the n-dimensional transformed data contains the 'best' n-dimensional view of the original N-dimensional data. If the remaining eigenvalues are very small in comparison, (8) should be a good approximation to (5). This idea is used to develop a filtering approach, the details of which are given in the next subsection.

2.2 Filtering Approach

If we have N observations, the related $N \times N$ covariance matrix is calculated by the covariance kernel function $\boldsymbol{\Sigma}^{(N)} = (k(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}))$. Following the discussion in the above section, $\boldsymbol{K}$ is constructed from the n eigenvectors of $\boldsymbol{\Sigma}^{(N)}(\boldsymbol{\theta})$ which are associated with the first n largest eigenvalues. Since $\boldsymbol{\theta}$ is unknown, we need to use an estimate $\hat{\boldsymbol{\theta}}$ based on those N observations. A standard method (see for example Williams and Rasmussen, 1996, and Shi, Murray-Smith and Titterington, 2002) can be used to calculate $\hat{\boldsymbol{\theta}}$. Then $\boldsymbol{\Sigma}^{(N)}$ is approximated by $\boldsymbol{\Sigma}^{(N)}(\hat{\boldsymbol{\theta}})$. The related eigenvalues and eigenvectors are calculated from $\boldsymbol{\Sigma}^{(N)}$ and are used to construct filtered data. Since the complexity of obtaining the estimate $\hat{\boldsymbol{\theta}}$ and calculating eigenvalues is $O(N^3)$, it is very time consuming for large N. Fortunately, the Nyström method (Williams and Seeger, 2001) can be used to calculate the n largest eigenvalues and the associated eigenvectors approximately. It is a very efficient approach, especially when $n \ll N$.

The procedure for generating a filtered data-set is as follows.

Step 1. Select a subset of training data of size m at random from the N observations. This m may be much less than N. We use a standard method to calculate an estimate $\hat{\boldsymbol{\theta}}^{(m)}$ using those m observations. The covariance matrix of those m observations is estimated by the covariance kernel function $\boldsymbol{\Sigma}^{(m)} = \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j; \hat{\boldsymbol{\theta}}^{(m)})\right)$, which is an $m \times m$ matrix. We calculate its eigenvalues $\lambda_1 \geq \cdots \geq \lambda_m \geq 0$ and the related eigenvectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m$.

Step 2. By the Nyström method, the first m largest eigenvalues of $\boldsymbol{\Sigma}^{(N)}$ can be approximated by

$$\frac{N}{m}\lambda_i,$$

for $i = 1, \cdots, m$, and their associated eigenvectors are

$$\sqrt{\frac{m}{N}}\frac{1}{\lambda_i}\boldsymbol{\Sigma}_{N,m}\boldsymbol{v}_i, \tag{9}$$

where $\boldsymbol{\Sigma}_{N,m}$ is the appropriate $N \times m$ submatrix of $\boldsymbol{\Sigma}^{(N)}$.

Step 3. We select the first n ($\leq m$) eigenvectors in order to construct the transformation matrix $\boldsymbol{K}$ in (7), and thereby generate an n-dimensional filtered data-set.

In the above procedure, we need to select m and n. We first discuss how to select n. The basic idea of the filtering approach is to use (8) to approximate (5). In the extreme case where $\lambda_i = 0$ for all $i > n$, the filtered data are equivalent

to the original data, in terms of the covariance kernel. This typically does not happen in practice. However, if the values of λ_i for all $i > n$ are very small compared to the first n eigenvalues, (8) is a good approximation of (5). Though it is difficult to compare (8) and (5) directly, we can compare the values of eigenvalues and choose n such that the remaining eigenvalues are very small in comparison to the largest eigenvalue. Alternatively, we might select n such that

$$\frac{\sum_{i=1}^{n} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geq c,$$

where c is a constant, such as $c = 0.99$. More discussion will be given in Section 3 in the context of the simulation study.

The other problem is how to select m. In Step 1, we select m observations and use them to learn the eigen-structure of the covariance kernel $k(\boldsymbol{x}, \boldsymbol{u})$. It is obvious that a larger value of m should lead to a more accurate approximation of eigenvalues and eigenvectors. However, we usually just need to learn the eigen-structure once. It can then be used repeatedly in similar systems. It will not increase the computational burden very much if we select a relatively large value of m. On the other hand, since the eigenvectors are used to generate a 'best' n-dimensional view of the original data, the accuracy of the 'design' in the first stage will not have much influence on carried out in the second stage. Some numerical results will be presented in the next section.

2.3 Model Learning and Prediction Using Filtered Data

The procedure proposed in the last subsection is used to generate a filtered data-set. Here we discuss how to carry out inference based on the filtered data.

The filtered data are defined via a linear transformation $\boldsymbol{Z} = \boldsymbol{KY}$, which can be used to design a set of filters. The observed filtered data may be obtained through those filters, so for generality we can consider observed errors. The observed filtered data are assumed to be

$$s_k = z_k + e_i, \quad \text{for } i = 1, \cdots, n,$$

where $\boldsymbol{Z} = (z_k) = \boldsymbol{KY}$ and the e_i are independent and identically distributed as $N(0, \sigma_s^2)$ which is the random error when the filtered data are observed. In matrix form, the filtered data $\boldsymbol{S} = (s_1, \cdots, s_n)^T$ are distributed as

$$\boldsymbol{S} \sim N(0,\ \boldsymbol{\Sigma}_s),$$

where $\boldsymbol{\Sigma}_s = \boldsymbol{K\Sigma K}^T + \sigma_s^2 \boldsymbol{I}_n$, and $\boldsymbol{K}$ is a known matrix which is designed in the first stage. We still use $\boldsymbol{\theta}$ to denote the unknown parameters involved in $\boldsymbol{\Sigma}_s$, which includes σ_s^2 and the unknown parameters in kernel covariance function. Then the log-likelihood of $\boldsymbol{\theta}$ is

$$L(\boldsymbol{\theta}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}_s| - \frac{1}{2} \boldsymbol{S}^T \boldsymbol{\Sigma}_s^{-1} \boldsymbol{S} - \frac{n}{2} \log(2\pi).$$

Maximizing $L(\boldsymbol{\theta})$ leads to a maximum likelihood estimate of θ.

Suppose that we wish to predict $z^* = \boldsymbol{K}^{*T}\boldsymbol{Y}^*$, where $\boldsymbol{Y}^* = \boldsymbol{Y}(\boldsymbol{X}^*)$ is q-dimensional, and $\boldsymbol{X}^*$ represents q test data points of input. $\boldsymbol{K}^*$ is a known q-dimensional vector, which can be thought of as a filter. Given the filtered data $\boldsymbol{S}$, the conditional mean and variance of z^* are

$$\hat{\mu}^* = \boldsymbol{K}^{*T}\boldsymbol{\Sigma}_{X^*X}\boldsymbol{K}^T\boldsymbol{\Sigma}_s^{-1}\boldsymbol{S}$$
$$\hat{\sigma}^{*2} = \boldsymbol{K}^{*T}\boldsymbol{\Sigma}_{X^*X^*}\boldsymbol{K}^* - \boldsymbol{K}^{*T}\boldsymbol{\Sigma}_{X^*X}\boldsymbol{K}^T\boldsymbol{\Sigma}_s^{-1}\boldsymbol{K}\boldsymbol{\Sigma}_{XX^*}\boldsymbol{K}^*,$$

where $\boldsymbol{\Sigma}_{X^*X} = \left(k(\boldsymbol{x}_i^*, \boldsymbol{x}_j; \hat{\boldsymbol{\theta}})\right)$ is the $q \times N$ covariance matrix between $\boldsymbol{X}^*$ and $\boldsymbol{X}$ evaluated at $\hat{\boldsymbol{\theta}}$ which is an estimate using $\boldsymbol{S}$, and so are the other similar notations.

If we want to make a single prediction at a new $y^* = y(\boldsymbol{x}^*)$, we just need to take $q = 1$ and $K^* = 1$. Bayesian inference can also be used. The implementation is similar to the methods discussed in Rasmussen (1996) and Shi, Murray-Smith and Titterington (2002).

2.4 Online Filtering Approach

We assume that data arrive sequentially. Let $\boldsymbol{D}_a = (\boldsymbol{Y}_a, \boldsymbol{X}_a)$ denote the data collected between time $t(a)$ and $t(a-1)$. We can apply the filtering approach online and adapt the predictive distribution for test data point. For each subset of data $\boldsymbol{D}_a$, we have a set of filtered data sets,

$$\boldsymbol{S}_a = \boldsymbol{Z}_a + \boldsymbol{e}_a, \quad \boldsymbol{Z}_a = \boldsymbol{K}_a\boldsymbol{Y}_a$$

for $a = 1, 2, \cdots, A$, where A is the number of data sets up to time $t(A)$. The transformation matrix $\boldsymbol{K}_a$ can be constructed by Step 2 of the filtering approach discussed in Subsection 2.2. We assume that the eigenstructure of the covariance kernel for the new data is similar to the previous data, so we just need to learn the eigenstructure once. It can be used repeatedly for new data sets, so the computation to generate a new filtered data-set is therefore very efficient. An estimate of $\boldsymbol{\theta}$ based on filtered data $\boldsymbol{S}_a$ is obtained by the method discussed in the last subsection, and is denoted by $\hat{\boldsymbol{\theta}}_a$. If we are interested in prediction at a new $y^* = y(\boldsymbol{x}^*)$, the predictive mean and variance based on the filtered data are

$$\hat{\mu}_a^* = \boldsymbol{\Sigma}_a^*\boldsymbol{K}_a^T\boldsymbol{\Sigma}_{s,a}^{-1}\boldsymbol{S}_a \tag{10}$$
$$\hat{\sigma}_a^{*2} = k(\boldsymbol{x}^*, \boldsymbol{x}^*; \hat{\boldsymbol{\theta}}_a) - \boldsymbol{\Sigma}_a^*\boldsymbol{K}_a^T\boldsymbol{\Sigma}_{s,a}^{-1}\boldsymbol{K}_a\boldsymbol{\Sigma}_a^{*T} \tag{11}$$

where $\boldsymbol{\Sigma}_a^* = \left(k(\boldsymbol{x}^*, \boldsymbol{x}_{j,a}; \hat{\boldsymbol{\theta}}_a)\right)$, and all the other quantities are defined in the last subsection but evaluated at $\hat{\boldsymbol{\theta}}_a$.

Therefore, we have

$$\hat{\mu}_a^* \sim N(\mu^*, \hat{\sigma}_a^{*2})$$

for $a = 1, \cdots, A$. Here, $\hat{\mu}_a^*$'s are correlated with each other with covariance

$$\begin{aligned}\hat{\sigma}_{ab}^* &= \mathrm{cov}(\hat{\mu}_a^*, \hat{\mu}_b^*) \\ &= \boldsymbol{\Sigma}_a^*\boldsymbol{K}_a^T\boldsymbol{\Sigma}_{s,a}^{-1}(\boldsymbol{K}_a\boldsymbol{\Sigma}_s^{ab}\boldsymbol{K}_b^T)\boldsymbol{\Sigma}_{s,b}^{-1}\boldsymbol{K}_a\boldsymbol{\Sigma}_b^{*T},\end{aligned} \tag{12}$$

where $\boldsymbol{\Sigma}_s^{ab}$ are the covariance matrix between $\boldsymbol{Y}_a$ and $\boldsymbol{Y}_b$. The correlation is calculated by $\rho_{ab}^* = \hat{\sigma}_{ab}^*/\hat{\sigma}_a^*\hat{\sigma}_b^*$. The overall mean of prediction based on A data-sets can be calculated by

$$\mu_* = \mathbf{1}^T \Omega^{-1} \hat{\boldsymbol{\mu}}_* / (\mathbf{1}^T \Omega^{-1} \mathbf{1})$$

and the variance is

$$\sigma_*^2 = (\mathbf{1}^T \Omega^{-1} \mathbf{1})^{-1},$$

where Ω is the covariance matrix of $\hat{\boldsymbol{\mu}}^* = (\hat{\mu}_1^*, \cdots, \hat{\mu}_A^*)^T$, with the diagonal element $\hat{\sigma}_a^{*2}$ and off-diagonal element $\hat{\sigma}_{ab}^*$, and $\mathbf{1} = (1, \cdots, 1)^T$.

If the correlation ρ_{ab}^* is not very large, we may approximate the predictive mean by

$$\mu_* = \frac{\sum_a \hat{\mu}_a^*/\hat{\sigma}_a^{*2}}{\sum_a 1/\hat{\sigma}_a^{*2}},$$

and the variance is

$$\sigma_*^2 = \frac{\sum_a 1/\hat{\sigma}_a^{*2} + 2\sum_{a \neq b} \rho_{ab}^*/(\hat{\sigma}_a^* \hat{\sigma}_b^*)}{(\sum_a 1/\hat{\sigma}_a^{*2})^2}.$$

This approximate method can be replaced by an iterative method. Each time we have a new data-set, we calculate the predictive mean (10), variance (11) and the covariance (12). Then, we can define the following iterative method:

$$\begin{aligned}
u^{(a)} &= u^{(a-1)} + \hat{\mu}_a^*/\hat{\sigma}_a^{*2}, \\
v^{(a)} &= v^{(a-1)} + 1/\hat{\sigma}_a^{*2}, \\
w^{(a)} &= w^{(a-1)} + 2\sum_{j=1}^{a-1} \rho_{aj}^*/(\hat{\sigma}_a^* \hat{\sigma}_j^*), \\
\hat{\mu}^{*(a)} &= u^{(a)}/v^{(a)}, \\
\hat{\sigma}^{*(a)2} &= (v^{(a)} + w^{(a)})/(v^{(a)})^2.
\end{aligned}$$

We can therefore maintain a much smaller set of training data, and can subsequently update the predictions online, as new data becomes available.

3 Applications

3.1 Learning with Large Data-Sets

As we have discussed in Section 1, a major limiting factor in the use of Gaussian process models is the heavy computational burden associated with large training data-sets, as the complexity grows at rate $O(N^3)$. Some methods have been proposed for overcoming this. Murray-Smith and Pearlmutter (2003) argued that the complexity is $O(n^3) + O(N^2 n)$ for the model learning and prediction based on the filtered data, which corresponding to the second stage in this paper. In our first stage, the complexity associated with generating filtered data is $O(m^3)$,

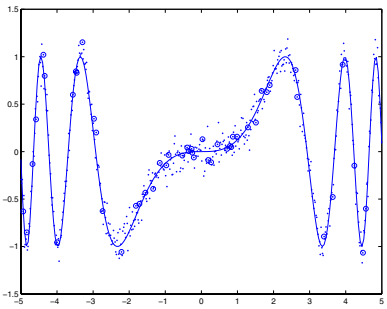
and therefore the overall complexity is $O(m^3) + O(n^3) + O(N^2 n)$. Since $n \leq m$ and usually $m \ll N$, the complexity is generally dominated by $O(N^2 n)$, and thus the filtering approach results in substantially decreased computational burden.

An example is used here to illustrate the filtering approach discussed in this paper. The original 500 training data (dots) and the $m = 50$ randomly selected data points (circles) are presented in Figure 1(a). The true model used to generate the data is $y_i = \sin((0.5x_i)^3) + \epsilon_i$, where the ϵ_i's are independent and identical distributed as $N(0, 0.01^2)$ and $x_i \in (-5, 5)$. The 50 selected data points are used to calculate the eigenvalues, and the related eigenfunctions and eigenvectors using the method described in Step 1 in Section 2.2. We take the values of c as 0.99, 0.999 and 0.9999, obtaining the values of n as 27, 33 and 39 respectively. The predictions and 95% confidence intervals for a test data set are presented in Figures 1(d) to 1(f).

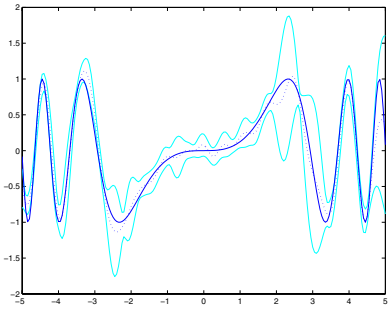
There are several interesting findings from these figures. Figure 1(e) gives the best results in terms of the value of root of mean squared error between the true test values and the predictions. Though it involves just $n = 33$ filtered data, the results are better than the results in Figure 1(b), obtained from 50 randomly selected data points. Figure 1(f) gives the results obtained from $n = 39$ filtered data points. The performance is slightly worse than Figure 1(e), based on only 33 filtered data points, though the difference is very small. This in fact coincides with the theory we discussed in Section 2.1. The filtering approach always chooses the largest eigenvalues and the related transformed data. It will not add much information to add more filtered data associated with relatively small eigenvalues. Comparing Figures 1(e) and 1(f), six more filtered data points are added. The associated eigenvalues are range from $\lambda_{34} = 0.0056$ to $\lambda_{39} = 0.0009$ and, relative to the largest eigenvalue $\lambda_1 = 2.5365$, the valuess ranged from 0.0022 to 0.0003, which are extremely small. In contrast, the numerical error may increase because the covariance matrix deteriorates due to those small eigenvalues. Thus it is not surprising that the performance of 1(f) is slightly worse than Figure 1(e). It shows that only a certain small number of filtered data points are needed to provide a good representation of the whole data set of N observations.

Figure 1(d) gives the results based on only $n = 27$ filtered data points. If we just use a randomly selected subset of 27 data points, the results are presented in Figure 1(c). The former is obviously much better than the latter. The other problem of using subset of data points is the sensitivity of the method to the choice of the data points. Our simulation study shows that the performance may be improved if those 27 data points are advantageously distributed over the whole range. For this, the training set must be located in all parts of the range, and there must be enough training data in regions where the mean response changes rapidly, such as near the two ends of the range in this example. Obviously, it is not easy to guarantee this. The performance will be poor when the data points are concentrated in certain areas. However, the filtering approach is quite robust.

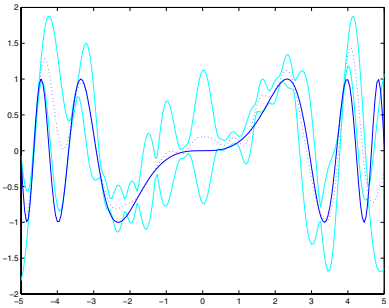
In our Step 1, an m-dimensional subset is selected for the calculation of the eigenvalues and eigenvectors. The accuracy depends on the value of m. For more



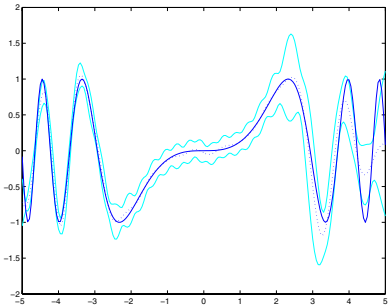
(a) 500 original data points (dots) and $m = 50$ selected data points (circles)



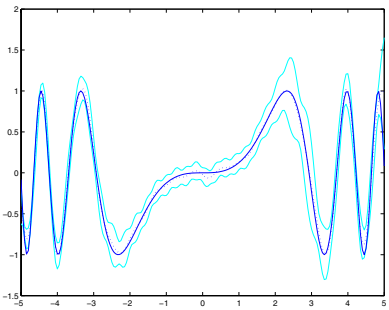
(b) with 50 random selected points; rmse=.1438



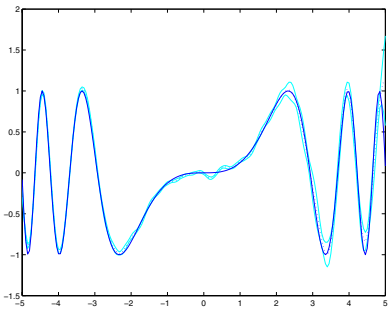
(c) with 27 random selected data points, rmse=.4263



(d) with $n = 27$ filtered data, rmse=.2063

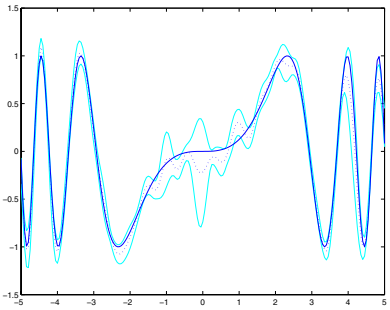


(e) with $n = 33$ filtered data, rmse=.0945

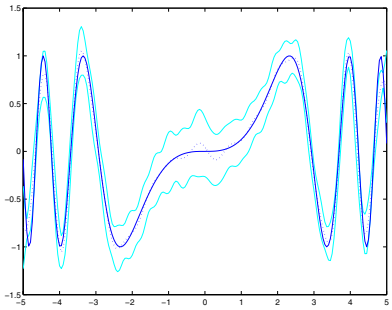


(f) with $n = 39$ filtered data, rmse=.1019

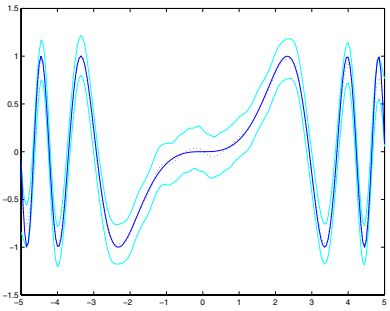
Fig. 1. Simulation study with $m = 50$: plot of true curve (solid line), prediction (dotted line) and 95% confidence intervals.



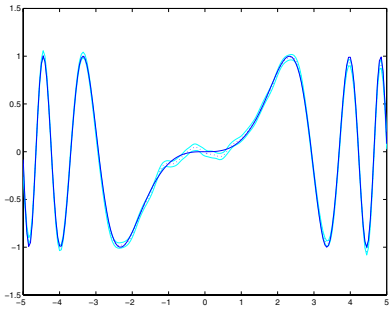
(a) with 100 random selected points; rmse=.0695



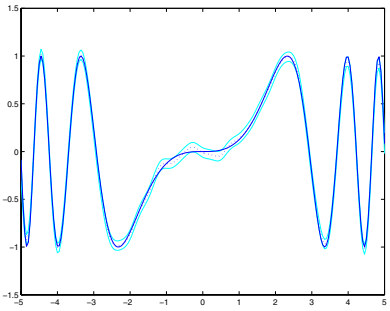
(b) with $n = 30$ filtered data, rmse=.1139



(c) with $n = 39$ filtered data rmse=.0611



(d) with $n = 46$ filtered data, rmse=.0293



(e) with $n = 56$ filtered data, rmse=.0291

Fig. 2. Simulation study with $m = 100$: plot of true curve (solid line), prediction (dotted line) and 95% confidence intervals.

accurate results, we should obviously select a relatively larger value of m. Figure 2 presents results when we take $m = 100$. We get quite similar results to those in Figure 1. For example, the value of rmse for $n = 30$ with $m = 100$ is between the values of rmse for $n = 27$ and $n = 33$ with $m = 50$ in Figure 1. When we added more filtered data, moving from the case of $n = 46$ in Figure 2(d) to $n = 56$ in Figure 2(e), the performance did not improve further. Of course, there is no surprise that we obtain more accurate results in 2(d) with $n = 46$ and $m = 100$, compared to Figure 1(e) with $m = 50$.

3.2 Inverse Problems

Suppose we want to transfer an image, which typically corresponds to a very large data-set, across a communication channel. One method is to compress the image into a data-set of much smaller size. On receipt of the compressed data-set, the original image is estimated. We can use the method discussed in Murray-Smith and Pearlmutter (2003). If the filtered data are represented by $\boldsymbol{Z}$, the transformation matrix used to construct the filtered data is $\boldsymbol{K}$, and the the original data-set $\boldsymbol{Y}$ can be estimated by

$$\boldsymbol{Y} = \boldsymbol{\Sigma}\boldsymbol{K}^T(\boldsymbol{K}\boldsymbol{\Sigma}\boldsymbol{K}^T)^{-1}\boldsymbol{Z}.$$

However, here we construct $\boldsymbol{K}$ in advance by selecting m data points from the original N data points using the method discussed in Section 2.2. There are two distinguishing features of this method. First, the filtered data provide approximately the 'best' n-dimensional view of the original N-dimensional data-set. Secondly, $\boldsymbol{K}\boldsymbol{\Sigma}\boldsymbol{K}^T$ is approximately a diagonal matrix $\mathrm{diag}(\lambda_1, \cdots, \lambda_n)$, so that numerically the inversion of $\boldsymbol{K}\boldsymbol{\Sigma}\boldsymbol{K}^T$ is well conditioned.

4 Conclusions

In this paper we have developed the work in Murray-Smith & Pearlmutter (2003), and have proposed a filtering approach based on approximate eigendecompositions of the covariance matrix, for dealing with large data-sets. There are two stages. The first stage is to generate a small-sized filtered data-set, which is a good representation of the original data-set so far as the covariance kernel is concerned. The second stage carries out model learning and prediction based on the filtered data. The method can be used in multi-scale learning, the solution of inverse problems and other areas.

References

1. Gibbs, Mark and MacKay, D. J. C. (1997). Efficient implementation of Gaussian processes.
2. Luo, Z and Wahba, G. (1997). Hybrid adaptive splines. *J. Amer. Statist. Assoc.*, **92**, 107-116.

3. Murray-Smith, R. and Pearlmutter, B. A. (2003). Transformations of Gaussian process priors. TR-2003-149, Department of Computing Science, University of Glasgow, Scotland, June.
4. O'Hagan, A. (1978). On curve fitting and optimal design for regression (with discussion). *Journal of the Royal Statistical Society B*, **40**, 1-42.
5. Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of IEEE*, **78**, 1481-1497.
6. Seeger, M., Willians, 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*.
7. Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*. PhD Thesis. University of Toronto. (Available from http: //bayes.imm.dtu.dk)
8. Shi, J. Q., Murray-Smith, R. and Titterington, D. M. (2002). Hierarchical Gaussian Process Mixtures for Regression, DCS Technical Report TR-2002-107/Dept. Statistics Tech. Report 02-7, University of Glasgow, Scotland.
9. Tresp, V. (2000). The Bayesian committee machine. *Neural Computation*, **12**, 2719-2741.
10. Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer Verlag, New York.
11. Wahba, G. (1990). *Spline Models for Observational Data*. SIAM, Philadelphia, PA. CBMS_NSF Regional Conference series in applied mathematics.
12. Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian process for regression. in D.S. Touretzky *et al* (eds), *Advances in Neural Information Processing Systems 8*, MIT Press.
13. Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, **13**. Eds T. K. Leen, T. G. Diettrich and V. Tresp. MIT Press.