To appear in *Neural Computation and Applications*

Bayesian Regression Filters and the Issue of Priors

Huaiyu Zhu and Richard Rohwer
Neural Computing Research Group
Department of Computer Science and Applied Mathematics,
Aston University, Birmingham B4 7ET, UK
Fax: 0121 333 3215, Email: `H.Zhu@aston.ac.uk` `R.J.Rohwer.ac.uk`

November 9, 1995

Abstract

We propose a Bayesian framework for regression problems, which covers areas which are usually dealt with by function approximation. An online learning algorithm is derived which solves regression problems with a Kalman filter. Its solution always improves with increasing model complexity, without the risk of over-fitting. In the infinite dimension limit it approaches the true Bayesian posterior. The issues of prior selection and over-fitting are also discussed, showing that some of the commonly held beliefs are misleading. The practical implementation is summarised. Simulations using 13 popular publicly available data sets are used to demonstrate the method and highlight important issues concerning the choice of priors.

Keywords: regression, Bayesian method, Kalman filter, approximation, prior selection, radial basis functions, on-line learning.

Running title: Bayesian Regression filter.

1 Introduction

Neural network models such as multi-layer perceptrons or radial basis function networks are used to approximate the regression of a data set. That is, the network output approximates the mean of the distribution of possible outputs, given its input. This function is not well-defined by the training data alone; prior information is also required to fully specify it.

It is a common practice to specify this prior implicitly by restricting the model order, which is a practical necessity to some extent anyway, or more explicitly by using a regulariser. Neither method is well-suited to expressing prior knowledge in a form likely to be familiar, such as the scales on which the regression is expected to vary. Furthermore it is not obvious how different models and regularisation terms should be compared.

These problems are addressed here by starting from the observation that the prior is best defined as a distribution over a space of functions in which the various models define subspaces. This prior can be projected into the model space where the calculations are carried out. Some proportion of the information in the data is neglected by this procedure, but this proportion is reduced to zero as the model complexity approaches that required to represent any function.

Because a fixed prior is projected into the model space, overfitting cannot occur as model order is increased.

The method is derived in §2. It is a basis function method using a Kalman filter algorithm, in which the prior enters via an initialisation procedure. Several useful classes of priors are introduced and discussed in §3. An illustration using synthetic data is presented in §4, and several tests on real data sets are presented in §5. The selection of priors for the real data sets would normally rely on domain knowledge. Data sets which are available over the Internet were used, and these are usually supplied with little domain knowledge. However, reasonable results were obtained by choosing priors in accord with the likely motives of people in general who make data sets available. This matter is discussed in §6. Conclusions are summarised in §7.

2 Theory and Algorithm

To avoid excessive notation we shall formulate the theory in the one dimensional case. The general case is quite similar, with more indices in the notation. The problem to be solved is to find a function $g : X \to Y$ as the regression of data set $z^k = [z_1, \ldots, z_k]$, where $X = Y = \mathbb{R}$ and $z_i = [x_i, y_i]$, such that the expected mean squared error $\langle (g(x_j) - y_j)^2 \rangle_k$ is minimised on any test data ($j > k$). In the above the notation $\langle \cdot \rangle$ and $\langle \cdot, \cdot \rangle$ denote mean and covariance, respectively, conditional on z^k when a subscript k is present, and unconditional when unsubscripted or with subscript 0.

In order that this problem has any solution at all, an underlying mechanism must be assumed. For this we assume that the data is generated by an unknown function $f : X \to Y$ and independent Gaussian noise η:

$$y_i = f(x_i) + \eta_i, \qquad \langle \eta_i \rangle = 0, \qquad \langle \eta_i, \eta_j \rangle = \sigma_2^2 \delta_{ij}, \tag{2.1}$$

where δ_{ij} is the Kronecker delta. The input x is assumed to be sampled from a distribution $p(x)$ which remains the same for both the training and test data.

Our algorithm will be based on the following decomposition of errors

$$\left\langle (y_j - g(x_j))^2 \right\rangle_k = \left\langle \| f - \widehat{f}_k \|^2 \right\rangle_k + \| \widehat{f}_k - g \|^2 + \left\langle \eta^2 \right\rangle, \tag{2.2}$$

where $\widehat{f}_k := \langle f \rangle_k$ is the posterior mean, and the norm $\| \cdot \|$ is defined by

$$\| f \|^2 := \int dx\, p(x) f(x)^2. \tag{2.3}$$

This can be verified by noting that

$$y - g(x) = (y - f(x)) + (f(x) - \widehat{f}_k(x)) + (\widehat{f}_k(x) - g(x)), \tag{2.4}$$

and that conditional on z_k, the three terms on the right hand side are independent. Therefore the generalisation error is the sum of three terms, representing the lack of knowledge of f (the variance), the deviation of g from the mean $\widehat{f}_k$ (the bias), and the intrinsic noise η. The minimisation problem is equivalent to minimising $\| \widehat{f}_k - g \|^2$ with respect to g. There is no bias/variance trade-off as discussed in [2], since the generalisation error is already defined as the expected error on test set.

The posterior of f also depends on the prior of f, ie. the unconditional distribution $\Pr(f)$. It is shown elsewhere [14, 15] that no learning rule can be independent of a prior, either explicitly or implicitly. We shall later explain how the explicit form of the prior can be written down for practical problems. Here we assume that f is a Gaussian random field [1] with a mean $\widehat{f}_0$ and a covariance kernel V_0. That is,

$$\langle f(\xi) \rangle_0 = \widehat{f}_0(\xi), \qquad \langle f(\xi_1), f(\xi_2) \rangle_0 = V_0(\xi_1, \xi_2). \tag{2.5}$$

After a data set z^k is observed we obtain a posterior $P(f|z^k)$, which is the distribution of f conditional on z^k. It is a Gaussian random field with mean $\widehat{f}_k$ and covariance kernel V_k.

If we actually compute $\widehat{f}_k$ and V_k, we shall be able to keep all the information in the posterior, which enables this process to carry on even if the data used to generate them is discarded [12]. They are called jointly sufficient statistics. Unfortunately, they can only be represented by vectors and matrices of dimension k, the size of the sample used to calculate them.

In our approach to be described here, we only keep an approximation of them, $\widehat{g}_k$ and S_k, within the model space M, which is a linear space of functions of fixed dimension m, spanned by a basis $\Phi = [\phi_1, \ldots, \phi_m]$, where each $\phi_i : X \to Y$. That is, $M = \{\sum_i \phi_i w_i : w \in \mathbb{R}^m\}$. The basis can be of any form, as long as on average the functions f in the prior $\Pr(f)$ can be approximated by members of M reasonably well.

If we knew the true $\widehat{f}_k$ and V_k, then the best approximation $\widehat{g}_k$ would be given by projecting them onto M, but unfortunately, to do this we need to keep all the information necessary to compute $\widehat{f}_k$. The projection operator is given by $\mathsf{P}_M = \Phi \Psi^T P$, where Ψ is the dual basis given by $\Psi = \Phi H^{-1}$, H is the Hessian given by $H = \Phi^T P \Phi$, and P is the inner product operator defined by $p(x)$. In greater detail, these are

$$\mathsf{P}_M f(\xi_1) = \sum_i \phi_i(\xi_1) \int d\xi_2 \, \psi_i(\xi_2) p(\xi_2) f(\xi_2), \tag{2.6}$$

$$\psi_j(\xi) = \sum_i \phi_i(\xi) (H^{-1})_{ij}, \tag{2.7}$$

$$H_{ij} = \int d\xi \, \phi_i(\xi) p(\xi) \phi_j(\xi), \tag{2.8}$$

$$P(\xi_1, \xi_2) = p(\xi_1) \delta(\xi_1 - \xi_2), \tag{2.9}$$

where $\delta(\xi)$ is the Dirac delta function.

The regression filter approach is to also "project" the likelihood function to the subspace, so that all the calculations can be carried out directly within M. This has the advantage that the method can be implemented as a Kalman filter and is very efficient in terms of computing cost. The drawback is that there is an inevitable loss of information, in the sense of Fisher efficiency. In practical terms, an efficiency $\epsilon < 1$ means that, given a data set of size k, the estimate given by this approximate method will be, on average, as accurate as that given by the best method possible from a data set of size ϵk. The efficiency depends on how well the functions in the prior are modelled by M on average. It increases towards unity as M becomes larger.

The actual algorithm is simply a standard Kalman filter algorithm [1] with a particular method for setting the initial mean and variance.

[1] In one dimensional cases it is usually called a random process.

1. Use prior knowledge to select a mean $\langle f \rangle_0 = \widehat{f}_0$, and covariance kernel $\left\langle f, f^T \right\rangle_0 = V_0$.
2. Define model space M by selecting basis Φ. Compute its dual basis Ψ.
3. Project the prior onto the weight space, represented by the mean $\widehat{w}_0 = \Psi^T P \widehat{f}_0$ and weight space covariance $S_0 = \Psi^T P V P \Psi$.
4. Given a new data point $z_k = [x_k, y_k]$, update the posterior by the Kalman filter, which is equivalent to

$$S_k^{-1} = S_{k-1}^{-1} + \sigma_2^{-2} \Phi(x_k)^T \Phi(x_k), \tag{2.10}$$

$$S_k^{-1} \widehat{w}_k = S_{k-1}^{-1} \widehat{w}_{k-1} + \sigma_2^{-2} \Phi(x_k)^T y_k, \tag{2.11}$$

   where $\Phi(x_k) = [\phi_1(x_k), \ldots, \phi_m(x_k)]$. [2] The standard implementation of a Kalman filter uses a more computationally efficient version of these equations.

3 Specifying the Prior

One requirement of the algorithm is the explicit specification of priors. This should represent the knowledge or requirement of the smoothness of the function f *before any data is observed.* This therefore acts as a definition of "over-fitting". In practice, one would like to specify the prior which actually describes the distribution of problems the algorithm is to be applied to. Therefore it is of interest to supply some generally applicable forms of prior and study their properties. The effect of the prior mean is intuitively clear, so here we shall only consider the effect of the prior covariance kernel.

Covariance kernels can generally be represented as $V_0(\xi_1, \xi_2)$. In practice, however, the most important ones are "translation invariant",

$$V_0(\xi_1, \xi_2) = V_0(\xi_1 - \xi_2). \tag{3.1}$$

Invariance means that the X space is parameterised so that before the arrival of data, we would not expect the variability of functions drawn from the prior at one point of X to be statistically different from that at another point. This property is only preserved under linear transforms. It can be shown that

$$\left\langle \|f\|^2 \right\rangle = \|\widehat{f}_0\|^2 + V_0(0). \tag{3.2}$$

Denoting $\sigma_1^2 := V_0(0)$, and using a scaling parameter σ_0 on X space, we have

$$V_0(\xi) = \sigma_1^2 V_0^0(\xi / \sigma_0), \tag{3.3}$$

where V_0^0 is normalised so that $V_0^0(0) = 1$. Here σ_0 specifies the X space correlation length of f, and σ_1 specifies the Y space variability of f. For multidimensional X and Y, σ_0 and σ_1 are in general matrices instead of scalars.

The following are some of the most important covariance kernels: [3]

[2] The points x_k are not random at the time it is used.

[3] The kernel, which might or might not be of the form of a Gaussian function, parameterises the prior, which is always Gaussian random filed.

- The Gaussian kernel (Figure 1)

$$V_0^0(\xi) = \exp(-\xi^2/2). \tag{3.4}$$

- The Laplace kernel (Figure 2)

$$V_0^0(\xi) = \exp(-|\xi|). \tag{3.5}$$

- The Bachelier-Wiener Process [4] (Figure 3) has a covariance function which is usually called the "hat" function

$$V_0^0(\xi) = (1 - |\xi|)_+ = \max\{1 - |\xi|, 0\}. \tag{3.6}$$

The prior can also be defined in terms of filtered white noise [4], which provides a convenient way to generate sample functions, thus assisting intuition about the prior. The filter F is related to the covariance kernel V by

$$V(\xi_1, \xi_2) = \int d\xi_3 F(\xi_1, \xi_3) F(\xi_2, \xi_3). \tag{3.7}$$

A sample from this prior can be generated from a white Gaussian noise process u by

$$f(\xi_1) = \int d\xi_2 F(\xi_1, \xi_2) u(\xi_2). \tag{3.8}$$

A translation invariant F always corresponds to a translation invariant V, and for any translation invariant V a corresponding translation invariant F can always be found. Either of the above implies that the Fourier transform of V is the square of the modulus of the Fourier transform of F. This fact can be used to generate a suitable F from a given V.

As a further example, a sample from a prior with Laplacian filter (not kernel) $F(\xi) = \sigma_1 \exp(-|\xi|/\sigma_0)$ are shown in Figure 4. This provides one of many possible intermediate levels of smoothness between the Gaussian and Laplacian kernels.

4 Illustration using synthetic data

A one dimensional example illustrates the main features of the the method (Figure 5), particularly the role of the parameters in the prior and the model order (Figure 6).

The prior $\Pr(f)$ is a Gaussian random process on $\mathcal{F}$ with zero mean $\langle f \rangle = 0$, and a Laplacian covariance kernel $V(\xi, \xi') = \sigma_1^2 \exp(|\xi - \xi'|/\sigma_0)$ with $\sigma_0 = .5$ and $\sigma_1 = 1$. Five randomly drawn sample functions from this prior are shown in Figure 5(a). The general variability reflected by these sample functions is all the prior knowledge.

Further suppose that all the training data and test data are contained in the interval $[-4, 4]$, with uniform distribution $p(x)$, and that the noise η is an independent zero-mean Gaussian process with $\sigma_2 = .5$. The training data is then generated by $y = f(x) + \eta$, using a particular f drawn from $\Pr(f)$, and is shown in Figure 5(b). This is all that is seen by the filter.

The approximation space M is chosen to be S_0^1-splines, ie., continuous piecewise-linear functions. [4] The basis functions are shown in Figure 5(c).

[4] Other basis functions such as harmonics or Gaussians, can also be used. They have been tested and give no substantial difference. The present choice makes the effect of basis functions more visible on the plots in Figure 6 for small m.

The dynamic progress of the regression filter is shown in Figure 5(d)–5(f), together with three error bars, indicating the cumulative effects of the uncertainty about $\widehat{g}_k$, the approximation, and the additive noise in the test data, in that order. [5]

These results can be compared with those obtained by the method of MacKay [5]. It is important to note that in that method, the model size and smoothness are implicitly related, because a diagonal prior on the weight space is used with a basis function width inversely proportional to the number of basis functions. Our results are tabulated for different m and σ_0 in Figure 6. The diagonal sub-plots correspond to those allowed in MacKay's framework. It is obvious from these plots that the parameter concerned with under- or over-fitting is σ_0, instead of m.

It might appear from a casual inspection of Figure 6 that increasing the dimensionality m will not enhance the results beyond a point determined by σ_0. This is not true in general, since all the plots in Figure 6 are obtained from a fixed-sized sample (ie. training set). As the sample size increases, the models with larger m will do better, even when σ_0 is fixed. The smoothness specified by σ_0 is only a preference. The final result is allowed to have any non-smoothness expressible in the approximation space, should the data provide strong evidence. On the other hand, the smoothness specified by using only smooth approximation functions will be such that only smooth curves are allowed, even when the data indicates otherwise.

If one really believes that the underlying function f is always smooth, it is advisable to assume a prior with Gaussian covariance kernel, which will assign vanishingly small prior variance to high frequency components [4]. This has the effect that it requires a higher than exponentially increasing amount of data to force the regression to have higher frequency components. This indeed makes a high-dimensional model redundant.

5 Tests on real data

It is widely held that the only objective way to measure the performance of a method is to train and test the algorithm on data sets from a large number of applications. This type of testing is not as "objective" as it might appear, since it risks blaming the method for what is actually an inappropriate (often implicit) choice of prior. However, even in this situation we can still apply the regression filter by determining an appropriate prior for the ensemble of tasks.

The method was tested on a selection of 13 publicly available data sets obtainable over the Internet. The choice of data sets was restricted by various considerations. For example, this algorithm is not designed for missing values, or non-Gaussian processes. We also want data sets for which the training sets and test sets can be mixed and resampled, partly because the algorithm is derived under the assumption that the training set and test set come form the same distribution, and partly because we need many simulation runs to measure an average performance. The selected datasets are shown in Table 3.

For this "Internet game" we make the following observations:

[5] The first two error bars are often practically indistinguishable, especially for plots with small numbers of data points and high model dimension.

- It seems reasonable to choose a prior which consists of a linear component which dominates the non-linear terms. [6]
- The data sets are usually supplied because there are some regularities contaminated by random noise, the ratio of which may be captured by σ_2/σ_1;
- The data sets are usually generated by researchers with some understanding of the application so that the number of sample points is about right for describing the regularities;
- The true regression is usually quite smooth so that a Gaussian kernel can be assumed;
- We perform an affine transform on the input space so that the training data inputs have zero mean and unit variance. [7] We assume that after this input preprocessing all the input dimensions are of comparable scale.

Therefore we write

$$V(\xi_1, \xi_2) = \sigma_1^2 \exp(-|\xi_1 - \xi_2|^2/2\sigma_0^2), \tag{5.1}$$

in terms of hyperparameters C_0 and C_2 where

$$C_0^d = N\sigma_0^d, \qquad \sigma_2 = C_2\sigma_1, \tag{5.2}$$

where d is the dimension of the input space X, and N is the sample size. The value of C_0 describe roughly how many sample points there are for each "oscillation" of the true regression. The value of C_2 describes roughly the "noise/signal ratio".

Table 1 shows the effect of C_0 and C_2 on generalisation performance for a particular data set, the "Boston house price" data. We choose to analyse this data set carefully partly because there are several results of other researchers which we can compare with, and partly because it is relatively small [8] so we could perform the following simulations requiring $5 \times 5 \times 20 = 500$ runs in reasonable time (several hours). We ran the algorithm on each combination of $C_0 = \sigma_0 N^{1/d} \in \{.75, 1.5, 3, 6, 12\}$ and $C_2 = \sigma_2/\sigma_1 \in \{.025, .05, .1, .2, .4, \}$. For each combination, we ran the algorithm 20 times, each time drawing a new partition into a training set and test set, and randomly selecting the basis centres from the training set. This produces 20 test set mean squared errors for each pair (C_0, C_2), for which the table gives the averages. Figure 7 shows an interpolation of this table.

A selection of results of other methods on the same data set is given in Table 2. These methods make various assumptions at various levels of explicitness about priors. It is clear by comparing Table 1 and Table 2 that the regression filter can perform better or worse than any of these, depending on the choice of prior. However, by the following more extensive experiments, we are able to enter a particular MSE, 13, for regression filter in Table 2.

Assuming that Internet data sets are drawn from one grandiose distribution, we can measure the hyperparameters of this distribution, C_1 and C_2, by inspecting an ensemble of data

[6] For computational simplicity, an alternative but equivalent approach is adopted. We first perform a linear regression on the training data, and then only work on the residuals. We assume that it has a zero mean.

[7] These operations involve little computational cost, and is always implicitly assumed in the sequel.

[8] It has 13 inputs and 1 output, with 506 samples partitioned to 253 training samples and 253 test samples. Each run takes about half a minute.

	C_2				
C_0	.02	.05	.1	.2	.4
.75	24.7428	21.8528	21.4950	21.3076	21.2822
1.5	18.0275	17.9603	17.9672	17.9990	18.1351
3	13.9183	13.7582	13.5115	13.3436	13.8677
6	13.7740	12.7685	11.8671	12.8733	15.8249
12	14.0007	11.8116	14.2269	17.4118	19.7492

Table 1: MSE for Boston housing data versus hyperparameters

Model & learning rule,	MSE on test sets
Regression, constant function	83.4
Regression, linear functions	28.9
Monte Carlo, network with 8 hidden units	13.7
Monte Carlo, network with two hidden layers	12.4
Gaussian process regression	11.9
Bayesian Regression filter	13

Table 2: MSE of different methods on the Boston housing data

The data in the first five rows are reproduced from [12]. The origin of the top four rows is [9].

sets. If the above assumptions about Internet data sets are correct, then this exercise should reveal a consistent range of values for these hyperparameters which could be expected to give reasonably good results for a randomly selected data set.

We trained the network on $K = 12$ other data sets $D_k, k = 1, \ldots, K$, with various number of runs. We always consider a data set as a whole, with the training sets and test sets being randomly sampled subsets. Since we know that this method always performs better with larger m, we choose m as large as possible within reasonable computation constraints. [9] The results are shown in Table 3. For each data set D_k, the algorithm was run several times on a selection of hyperparameter values C_i in order to obtain a crude estimate of the optimal values C_{ik}. Since these are scale parameters, we work with their logarithms. The estimates and their uncertainties are given in the form $\log_{10} C_{ik} \sim N(l_{ik}, {s_{ik}}^2)$. The means $\bar{l}_i$, uncertainty of the mean $\bar{s}_i$, and the standard deviation of the means among the data sets $\{D_i\}$ are calculated as

$$\bar{l}_i = \sum_k l_{ik} {s_{ik}}^{-2} / \sum_k {s_{ik}}^{-2}, \qquad {\bar{s}_i}^{-2} = \sum_k {s_{ik}}^{-2}, \tag{5.3}$$

$$ {S_i}^2 = \sum (l_{ik} - \bar{l}_i)^2 / (K-1). \tag{5.4}$$

[9] We usually choose m between 20 and 200. The data sets usually have dozens of input dimensions, a few output dimensions, and have sample sizes ranging from a few hundred to a tens of thousands. A single run usually takes several minutes on a Sun Sparc 10 workstation. The algorithm is implemented as several `matlab` programs. The basis functions are defined by choosing the centres randomly from the training inputs, and setting the width to $1/m^{1/d}$ on the transformed input.

They are given in Table 4, with and without including the Boston house price data.

k	Data sets D_k	C_0		C_2	
		l_{0k}	s_{0k}	l_{2k}	s_{2k}
	StatLib (CMU) data sets [a]				
0	boston	0.8	0.5	-1.2	0.5
1	bodyfat	1.0	1.0	-1.0	1.5
2	tecator	0.5	1.5	-1.5	1.5
3	polution	1.0	1.0	-0.5	1.0
4	cloud	0.0	3.0	0.0	2.0
	Data sets used in [10] [b]				
5	Mpg	0.5	2.0	-1.0	3.0
6	Cpu	1.0	1.0	-1.0	1.5
7	Aprce	0.0	1.5	-1.0	2.0
	UCI data sets [c]				
8	satim	0.5	1.0	-0.5	1.0
9	dna	1.0	1.5	-1.5	3.0
10	shuttle	1.0	2.0	-2.0	1.5
	Santa Fe data sets [d]				
11	santafeA	0.0	1.5	-1.0	1.5
12	santafeD	0.0	2.0	0.0	2.0

[a] These data sets can be obtained from `ftp://lib.stat.cmu.edu/datasets`.

[b] We obtained these data sets from J. Quinlan indirectly via G. Hinton in Toronto.

[c] These data sets can be obtained from `ftp://ics.uci.edu/pub/machinelearning`. They are also used in the StatLog Project [7].

[d] These data sets can be obtained from `ftp://ftp.santafe.edu/pub/TimeSeries`. We use them to construct learning problems of predicting the next step from a window of 50 time steps.

Table 3: Estimates of C_0 and C_2 for several data sets retrieved from Internet

It emerges that there is indeed a range of reasonable values for C_0 and C_2, for which the algorithm performs well on most data sets. These are $C_0 \approx 5$, $C_2 \approx .2$, within a factor of about 2.5 up or down. Translating back to the familiar language of training radial basis functions, this means that the regulariser should be derived from a covariance kernel having width of about 5 sample points in each dimension, and the prior should be about as important as $1/.2^2 = 25$ sample points.

Choosing these averages (omitting the Boston house price results) for the hyperparameters would give an MSE for the Boston housing data set of about 13, as entered for the regression filter algorithm in Table 2. Considering the uncertainties in the hyperparameters, this MSE might have been as little as 11.9 or as much as 19.7.

6 Discussion

Figure 7 shows clearly that the choice of prior, or its "hyperparameters" is important. In a fully consistent theory, neither the data nor the learning rule play any role in this, although proposals of this nature has been studied [5, 13, 6, 14]. The Bayesian regression filter has the advantage of using hyperparameters that express scales likely to be estimated from domain

	$\bar{l}_0$	$\bar{s}_0$	S_0	$\bar{l}_2$	$\bar{s}_2$	S_2
Average without Boston data						
$\bar{l}_i, \bar{s}_i, S_i$	0.68	0.39	0.47	-0.73	0.43	0.66
$10^{\bar{l}_i}, 10^{\bar{s}_i}, 10^{S_i}$	4.83	2.44	2.98	0.19	2.70	4.60
Average with Boston data						
$\bar{l}_i, \bar{s}_i, S_i$	0.73	0.31	0.47	-0.87	0.33	0.62
$10^{\bar{l}_i}, 10^{\bar{s}_i}, 10^{S_i}$	5.34	2.03	2.95	0.12	2.12	4.21

Table 4: Average of estimates of optimal hyperparameters

knowledge. This proved to be the case, to a reasonable extent, even in the domain of Internet data sets. It would be interesting to attempt to confirm this with a larger experiment.

It can be seen that the best performance of the regression filter algorithm with the "optimal hyperparameter" performs better than any of the other methods, on an average case bases. Such comparisons should be put in perspective, however. It is theoretically clear that with identical conditions the regression filter algorithm presented here cannot outperform the "regression with Gaussian processes" algorithm [12], since the latter keeps a sufficient statistic which is of the same size as the training set, and is always the optimal algorithm for any given prior. Considering the variability of the mean squared error expressed by Table 1, it can be said that to the precision of these experiments, the regression filter, which is a fixed finite dimensional method, performs comparably with exact Bayesian algorithms which either sample the posterior by a Monte Carlo method [8] or compute the posterior explicitly [12].

It is important to point out that the "Internet game" is quite artificial, since in applications one really wants the algorithm to perform well on the problems which would arise in a particular application area, rather than on average over many irrelevant problems. This shows a deficiency in common practice for evaluating learning rules. Since our method is asymptotically optimal (as m increases), the competition is really on whether one finds a good prior, which depends on the available computing resources and real world data sets, but not on the algorithm itself. From this point of view, tests on synthetic data with an explicitly stated prior are more important, even to real-world users of the algorithm, because they circumvent this issue. Real-world users would normally have better domain knowledge than in the "Internet game", and could therefore choose a prior which is better concentrated on problems from the particular application. This process is assisted by the intuitively clear interpretation of the hyperparameters of the regression filter algorithm, and figures such as Figure 1-4, which can be compared to domain-based intuition to select an appropriate kernel.

7 Conclusions

The regression problem is analysed in a Bayesian framework, and approximately solved within a finite-dimensional model space. By defining the prior directly in the function space and projecting it into the model space, we are able to show that the accuracy of the algorithm always increases with the complexity of the model, and approaches the true Bayesian posterior. The resulting algorithm is a straightforward Kalman filter with the initial covariance matrix set by the prior. The prior is specified in a form which is easy to relate to typical problem domains.

Using a real-world data set, the algorithm was shown to perform comparably to other Bayesian methods which require either Monte Carlo simulations or the maintenance of a sufficient statistic.

Acknowledgements:

This work was inspired by earlier joint work with C. Bishop, C. Qazaz, and C. Williams [11]. We would like to thank C. Williams for many interesting discussions and providing comparable data. The work of H. Zhu is supported by EPSRC Grant GR/J17814.

References

[1] C. K. Chui and G. Chen. *Kalman Filtering with Real-Time Applications.* Springer Series in Information Sciences. Springer-Verlag, Berlin, 1987.

[2] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[3] S. J. Hanson, J. D. Cowan, and C. Lee Giles, editors. *Advances in Neural Information Processing Systems*, volume 5, San Mateo, CA, 1993. Morgan Kaufmann.

[4] G. M. Jenkins and D. G. Watts. *Spectral Analysis and its Applications.* Holden-Day, San Francisco, 1968.

[5] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[6] D. J. C. MacKay. Hyperparameters: Optimize, or integrate out? In G. Heidbreder, editor, *Maximum Entropy and Bayesian Methods, Santa Barbara 1993*, Dordrecht, 1995. Kluwer.

[7] D. Michie, D. J. Speigelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[8] R. M. Neal. Bayesian learning via stochastic dynamics. In Hanson et al. [3], pages 475–482.

[9] R. M. Neal. *Bayesian Learning for Neural Networks.* PhD thesis, Dept. of Computer Science, University of Toronto, 1995. `ftp://ftp.cs.utoronto.ca/pub/radford/thesis.ps.Z`.

[10] Quinlan J. R. Combining instance-based and model-based learning. In P. E. Utgoff, editor, *Proceedings of the Machine Learning Conference '93*, San Mateo, CA, 1993. Morgan Kaufmann.

[11] C. K. I. Williams, C. Qazaz, C. M. Bishop, and H. Zhu. On the relationship between Bayesian error bars and the input data density. In *Fourth International Conference on Artificial Neural Networks. IEE Conference Publications no. 409*, pages 160–165, 1995.

[12] C. K. I. Williams and C. E. Rasmussen. Regression with gaussian processes. In M. Mozer D. Touretzky and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, 1996. (To appear).

[13] D. H. Wolpert. On the use of evidence in neural neworks. In Hanson et al. [3], pages 539–546.

[14] H. Zhu and R. Rohwer. Information geometric measurements of generalisation. Technical Report NCRG/4350, Aston University, 1995. `ftp://cs.aston.ac.uk/neural/zhuh/generalisation.ps.Z`.

[15] H. Zhu and R. Rohwer. Measurements of generalisation based on information geometry. Mathematics of Neural Networks and Applications Conference (MANNA), Oxford. Ann. Math. Artif. Intell.(to appear) `ftp://cs.aston.ac.uk/neural/zhuh/generalisation-manna.ps.Z`., 1995.

A Figure Legends

Figure 1: Effect of different prior parameters for Gaussian covariance kernels

> Five sample functions are drawn from each of the nine priors with Gaussian covariance function and parameters given in the titles of sub-plots.

Figure 2: Effect of different prior parameters for Laplace covariance kernels

> Five sample functions are drawn from each of the nine priors with Laplacian covariance function and parameters given in the titles of sub-plots.

Figure 3: Effect of different prior parameters for Bachelier-Winer covariance kernel

> Five sample functions are chosen from each of the nine priors with "hat" covariance function and parameters given in the titles of sub-plots.

Figure 4: Effect of different prior parameters for Laplace filters

> Five sample functions are drawn from each of the nine priors with Laplacian filters and parameters given in the titles of sub-plots.

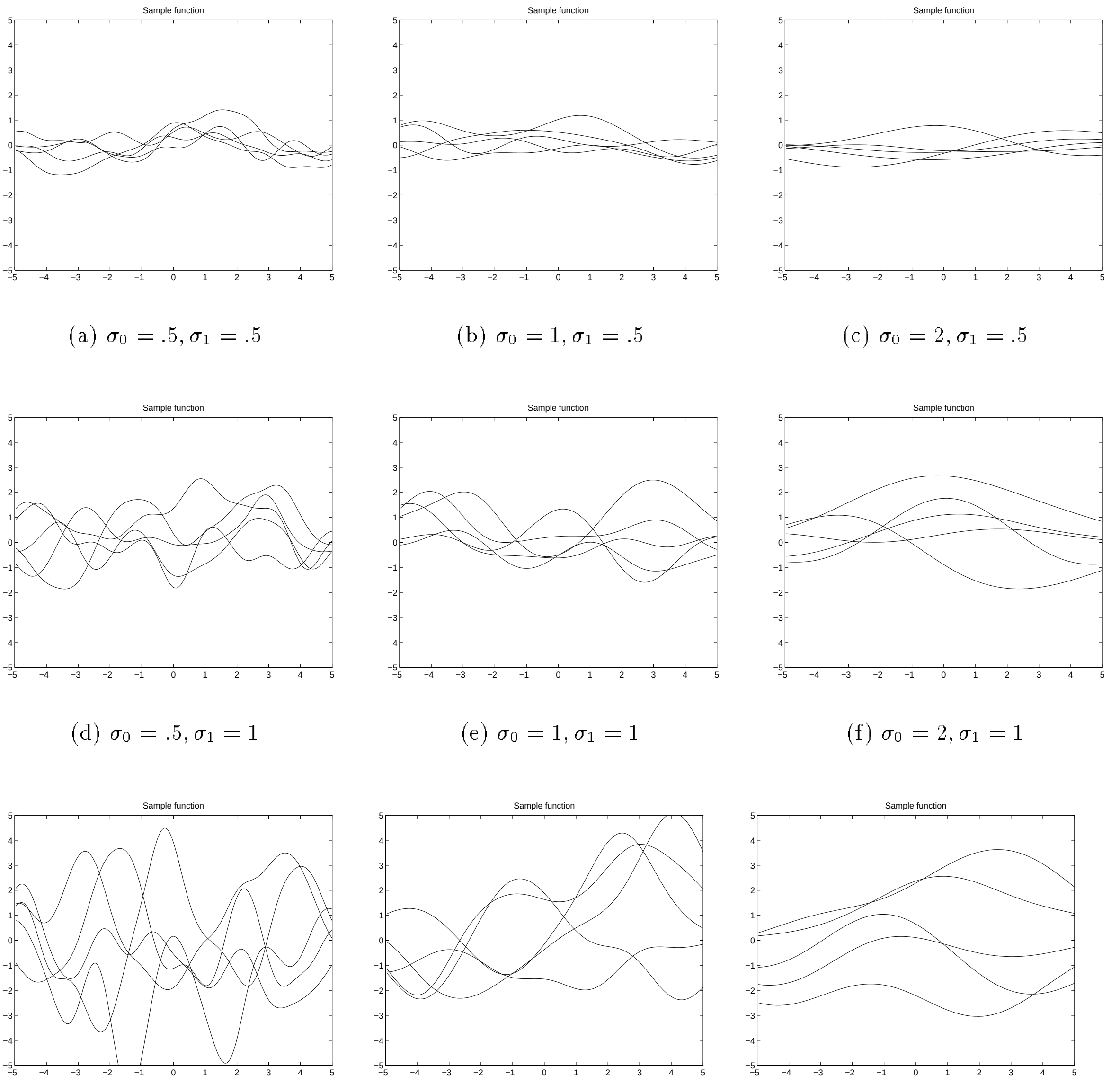
Figure 5: Numerical example of regression filter

> Prior: $V_0^0 = \exp(-|\xi|), \sigma_0 = .5, \sigma_1 = 1$. Sampling distribution: $p(x) = \text{uniform}[-4, 4]$. Sampling noise: $\sigma_2 = .5$. Sample size: $n = 160$. Approximation dimension: $m = 40$. Basis: continuous piecewise-linear functions with nodes uniformly on $[-4, 4]$.
>
> True function f: dashed line. Fitted curve: solid line. Error bars: dotted lines, from innermost outwards, that of $\mathsf{P}_M f$, of f and of y, respectively.

Figure 6: Effect of different model complexity and prior smoothness

> The true prior, the noise and the model basis form are the same as in Figure 5. The sample size $n = 100$. The different sub-plot uses different model dimension m and different assumed σ_0, which may not be the same as that of the true prior.

Figure 7: Mean squared error for Boston housing data versus prior parameters (interpolated with cubic spline)

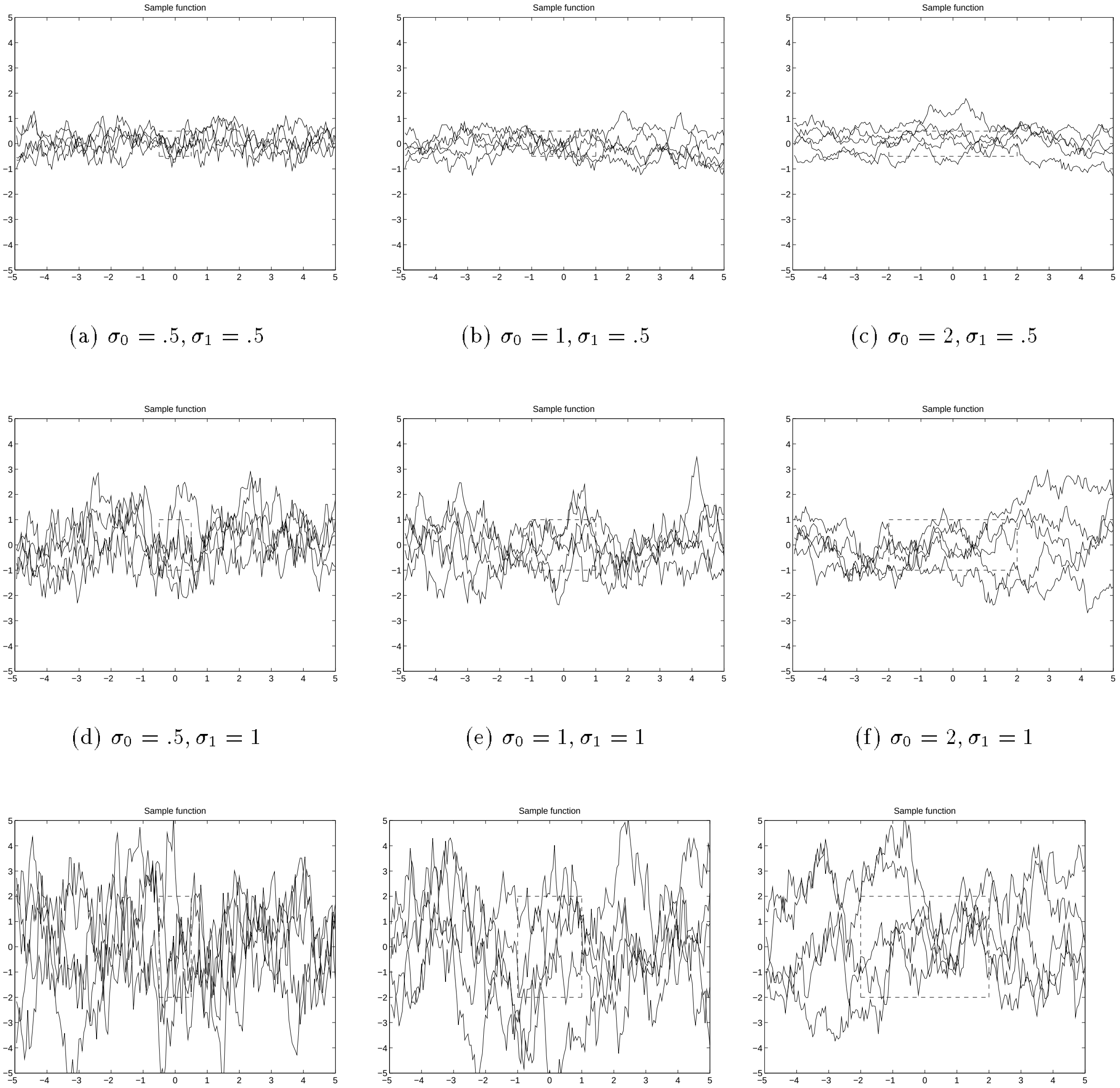


(a) $\sigma_0 = .5, \sigma_1 = .5$ (b) $\sigma_0 = 1, \sigma_1 = .5$ (c) $\sigma_0 = 2, \sigma_1 = .5$

(d) $\sigma_0 = .5, \sigma_1 = 1$ (e) $\sigma_0 = 1, \sigma_1 = 1$ (f) $\sigma_0 = 2, \sigma_1 = 1$

(g) $\sigma_0 = .5, \sigma_1 = 2$ (h) $\sigma_0 = 1, \sigma_1 = 2$ (i) $\sigma_0 = 2, \sigma_1 = 2$

Figure 1: Effect of different prior parameters for Gaussian covariance kernel



(a) $\sigma_0 = .5, \sigma_1 = .5$

(b) $\sigma_0 = 1, \sigma_1 = .5$

(c) $\sigma_0 = 2, \sigma_1 = .5$

(d) $\sigma_0 = .5, \sigma_1 = 1$

(e) $\sigma_0 = 1, \sigma_1 = 1$

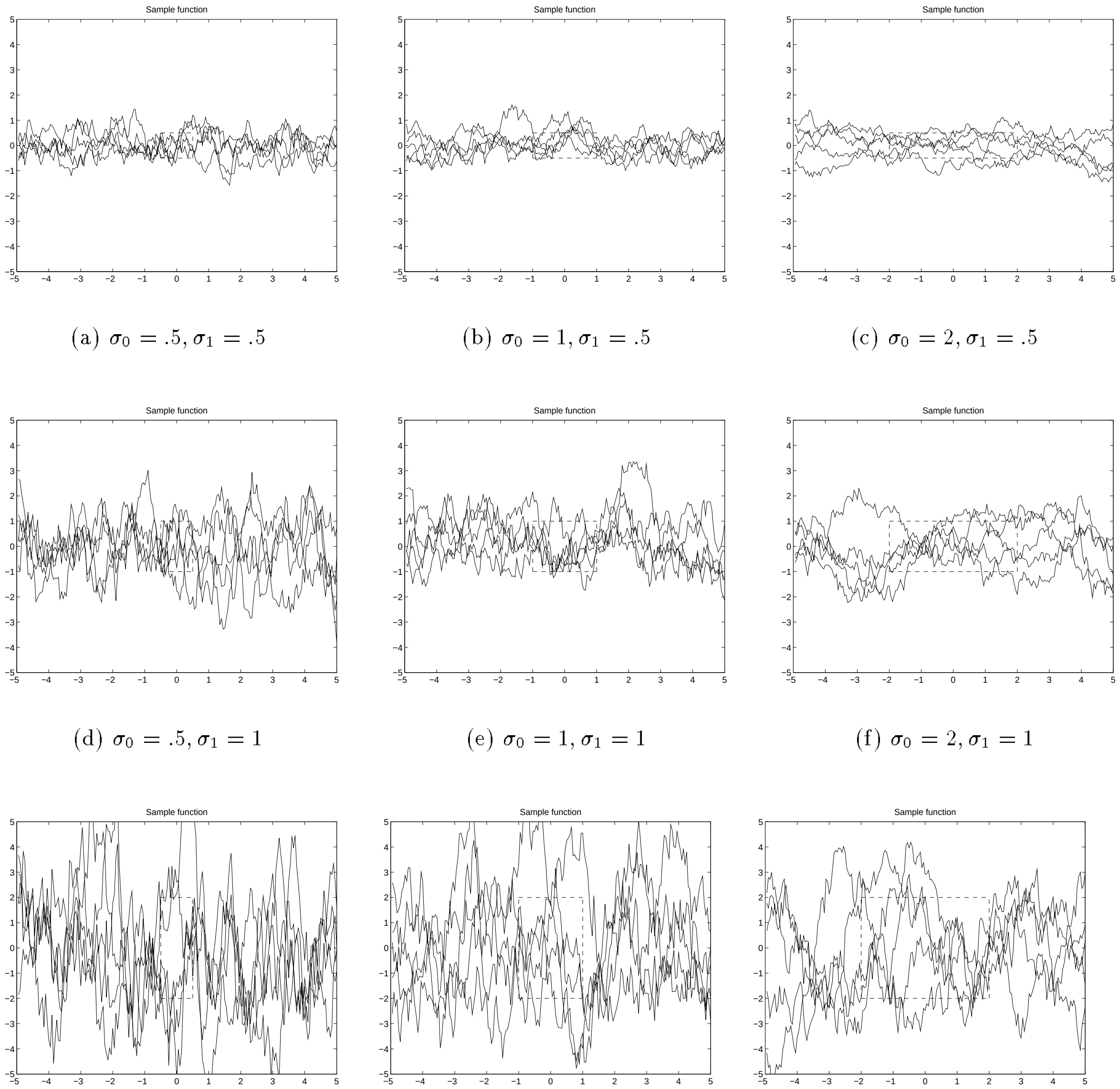
(f) $\sigma_0 = 2, \sigma_1 = 1$

(g) $\sigma_0 = .5, \sigma_1 = 2$

(h) $\sigma_0 = 1, \sigma_1 = 2$

(i) $\sigma_0 = 2, \sigma_1 = 2$

Figure 2: Effect of different prior parameters for Laplacian covariance kernels



(a) $\sigma_0 = .5, \sigma_1 = .5$

(b) $\sigma_0 = 1, \sigma_1 = .5$

(c) $\sigma_0 = 2, \sigma_1 = .5$

(d) $\sigma_0 = .5, \sigma_1 = 1$

(e) $\sigma_0 = 1, \sigma_1 = 1$

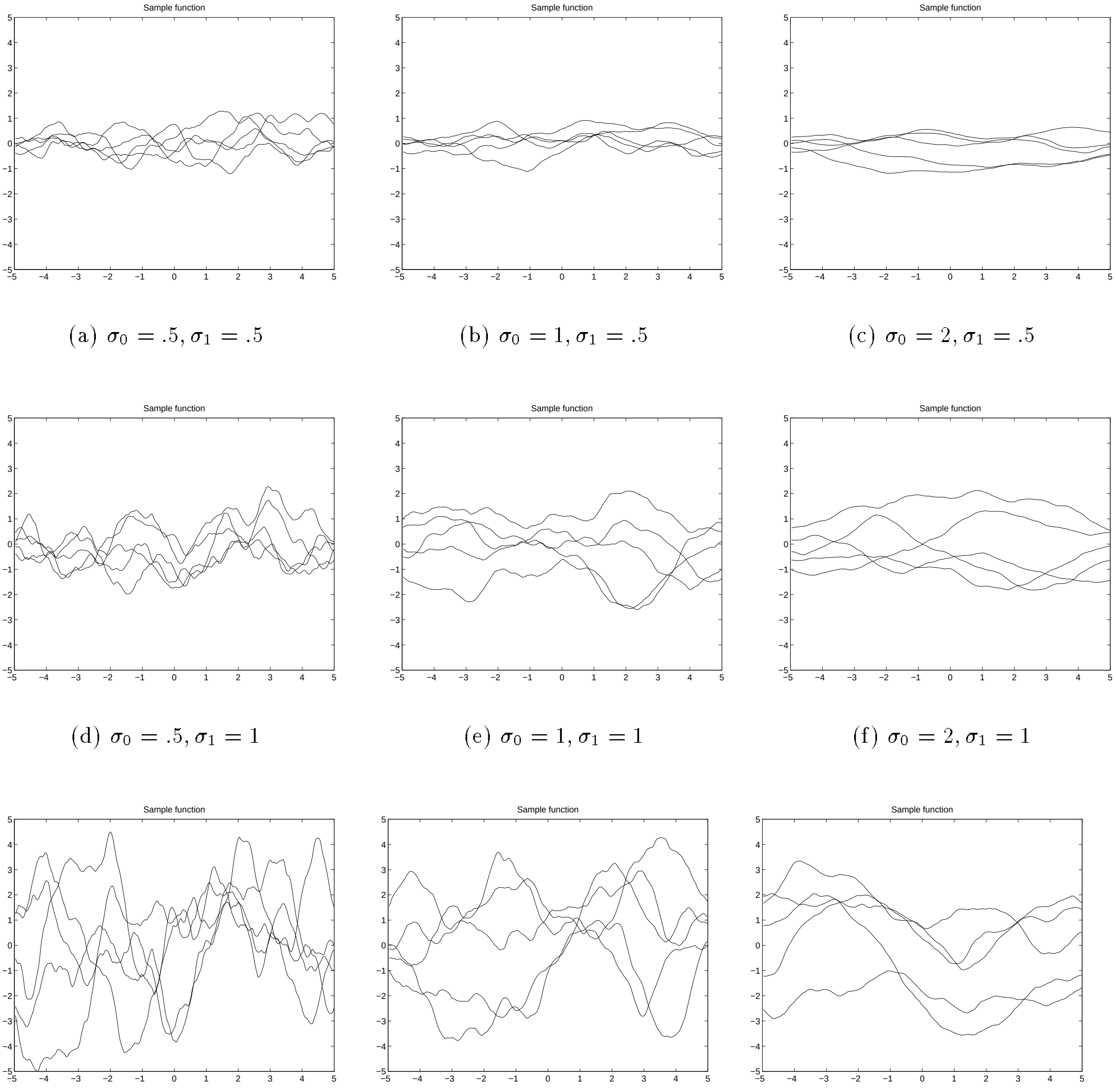
(f) $\sigma_0 = 2, \sigma_1 = 1$

(g) $\sigma_0 = .5, \sigma_1 = 2$

(h) $\sigma_0 = 1, \sigma_1 = 2$

(i) $\sigma_0 = 2, \sigma_1 = 2$

Figure 3: Effect of different prior parameters for Bachelier-Winer covariance kernels



(a) $\sigma_0 = .5, \sigma_1 = .5$

(b) $\sigma_0 = 1, \sigma_1 = .5$

(c) $\sigma_0 = 2, \sigma_1 = .5$

(d) $\sigma_0 = .5, \sigma_1 = 1$

(e) $\sigma_0 = 1, \sigma_1 = 1$

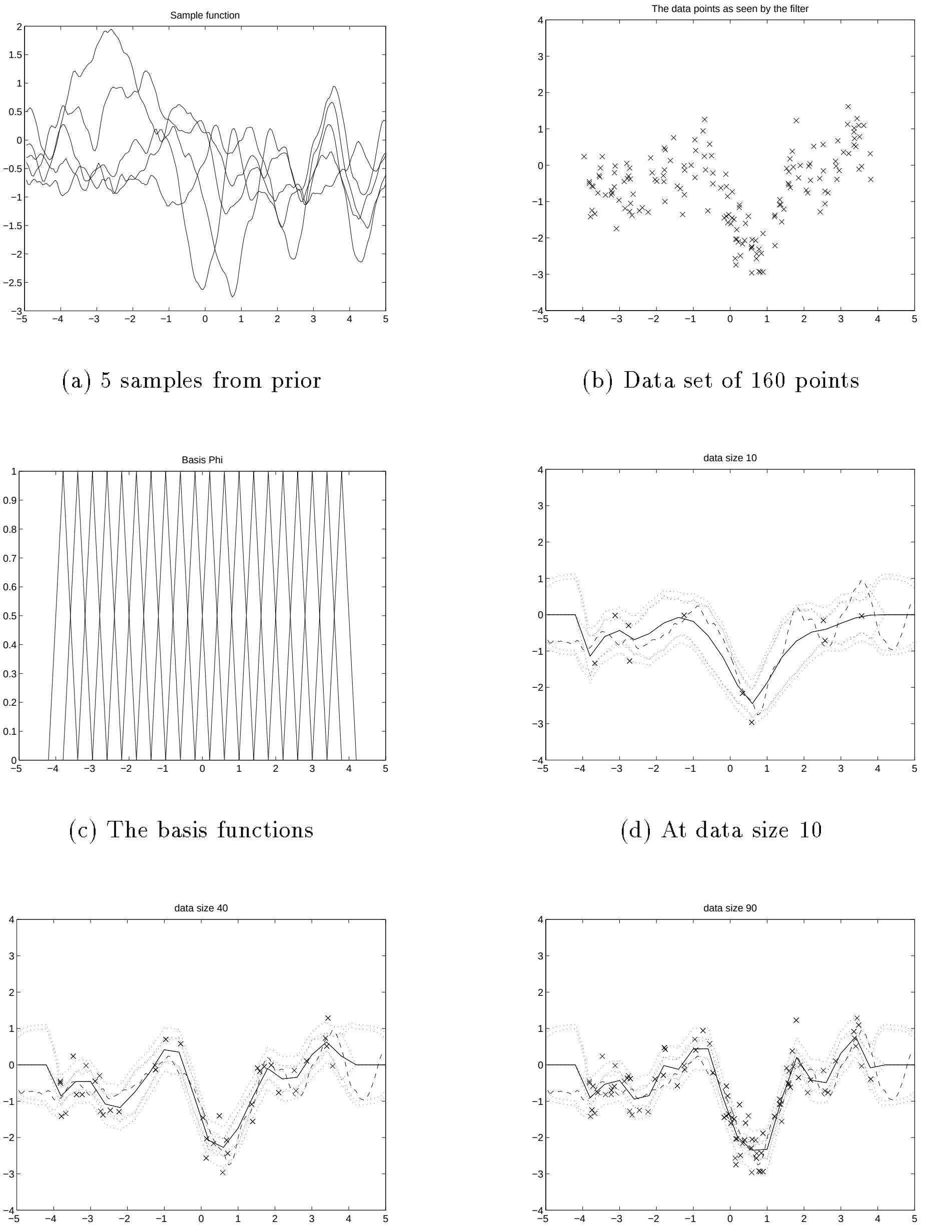
(f) $\sigma_0 = 2, \sigma_1 = 1$

(g) $\sigma_0 = .5, \sigma_1 = 2$

(h) $\sigma_0 = 1, \sigma_1 = 2$

(i) $\sigma_0 = 2, \sigma_1 = 2$

Figure 4: Effect of different prior parameters for Laplacian filters



(a) 5 samples from prior

(b) Data set of 160 points

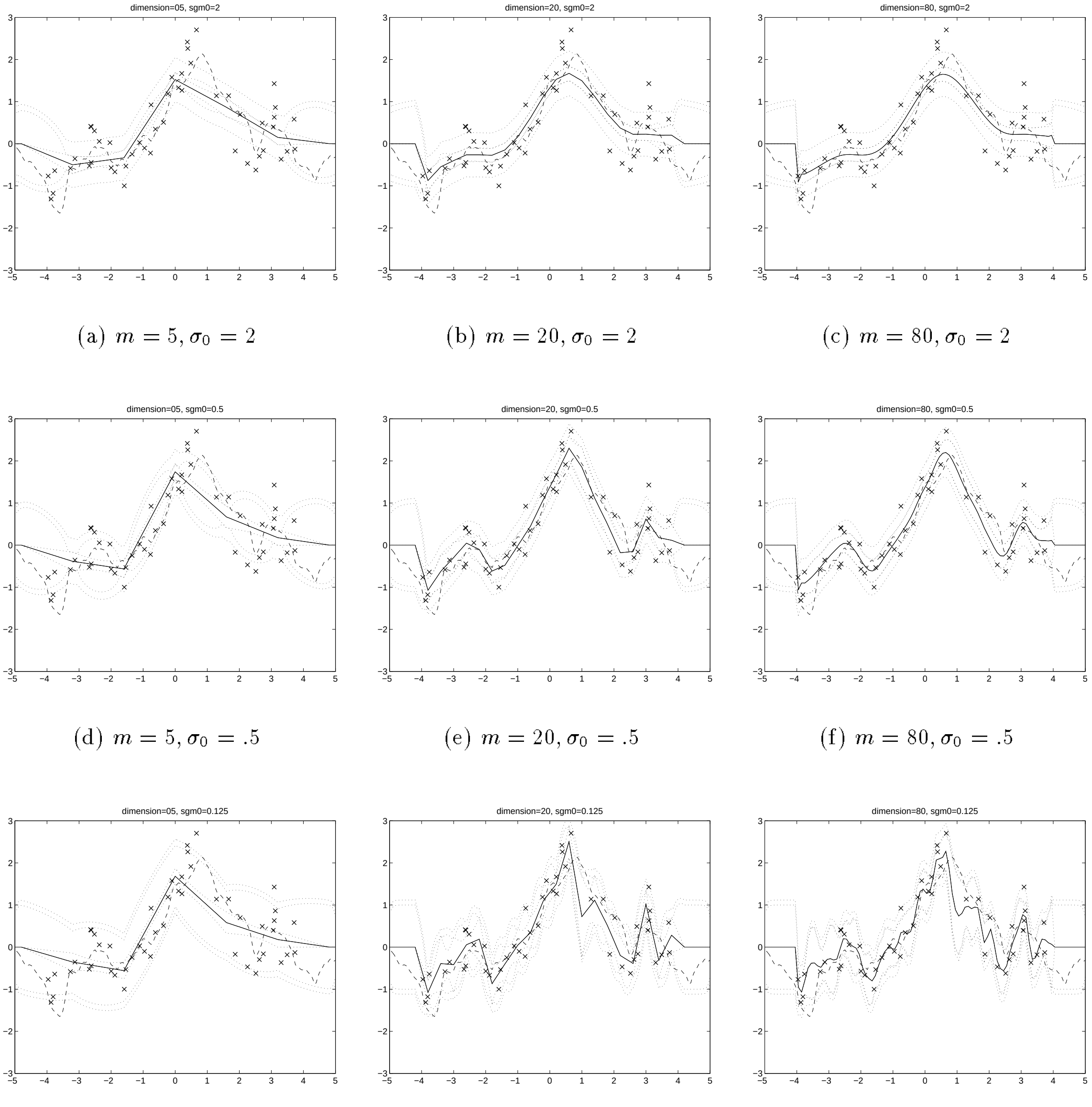
(c) The basis functions

(d) At data size 10

(e) At data size 40

(f) At data size 90

Figure 5: Numerical example of regression filter



(a) $m = 5, \sigma_0 = 2$

(b) $m = 20, \sigma_0 = 2$

(c) $m = 80, \sigma_0 = 2$

(d) $m = 5, \sigma_0 = .5$

(e) $m = 20, \sigma_0 = .5$

(f) $m = 80, \sigma_0 = .5$

(g) $m = 5, \sigma_0 = .125$

(h) $m = 20, \sigma_0 = .125$

(i) $m = 80, \sigma_0 = .125$

Figure 6: Effect of different model complexity and prior smoothness

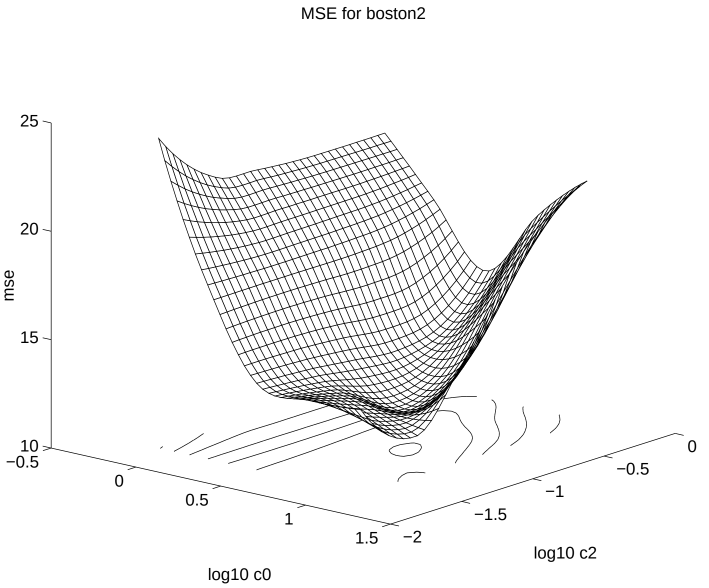


Figure 7: Mean squared error for Boston housing data versus prior parameters (interpolated with cubic spline)