The Bayesian Backfitting Relevance Vector Machine

Aaron D'Souza ADSOUZA@USC.EDU
University of Southern California, Los Angeles, CA 90089, USA

Sethu Vijayakumar SETHU.VIJAYAKUMAR@ED.AC.UK
University of Edinburgh, Edinburgh EH9 3JZ, UK

Stefan Schaal SSCHAAL@USC.EDU
University of Southern California, Los Angeles, CA 90089, USA and ATR Computational Neuroscience Laboratory, Kyoto, Japan

Abstract

Traditional non-parametric statistical learning techniques are often computationally attractive, but lack the same generalization and model selection abilities as state-of-the-art Bayesian algorithms which, however, are usually computationally prohibitive. This paper makes several important contributions that allow Bayesian learning to scale to more complex, real-world learning scenarios. Firstly, we show that *backfitting* — a traditional non-parametric, yet highly efficient regression tool — can be derived in a novel formulation within an expectation maximization (EM) framework and thus can finally be given a probabilistic interpretation. Secondly, we show that the general framework of *sparse Bayesian learning* and in particular the relevance vector machine (RVM), can be derived as a highly efficient algorithm using a Bayesian version of backfitting at its core. As we demonstrate on several regression and classification benchmarks, Bayesian backfitting offers a compelling alternative to current regression methods, especially when the size and dimensionality of the data challenge computational resources.

1. Introduction

Real-world data, for instance obtained from neuroscience, chemometrics, data mining, or sensor-rich environments, is frequently extremely high-dimensional, severely underconstrained (few data points), even interspersed with large amounts of irrelevant and/or redundant features. Combined with the inevitable measurement noise, efficient learning from such data still poses significant challenges to state-of-the-art supervised learning algorithms, even in linear settings. While traditional statistical techniques (e.g. partial least squares (PLS) regression, backfitting) for supervised learning are often quite efficient and robust for these problems, they lack a probabilistic interpretation and cannot easily provide measures like predictive distributions or the evidence of data as needed for model selection. On the other hand, while recent algorithms in supervised learning compute such information, they lack computational efficiency as, for instance, in Gaussian process regression or support vector learning. The goal of this paper is to introduce a new algorithm that exploits the best of both worlds by developing a probabilistic formulation of a classical non-parametric non-probabilistic regression algorithm. As will be demonstrated, this algorithm can greatly improve the computational efficiency of the modern framework of sparse Bayesian learning, including feature detection and automatic relevance determination, and allow this technique to be applied for very high dimensional problems.

1.1. Sparse Bayesian Learning: The Relevance Vector Machine

The relevance vector machine was introduced by Bishop and Tipping (2000) as an alternative to the popular support vector regression (SVR) method. The RVM operates in a framework similar to generalized linear regression, but uses the following generative model:

$$y(\mathbf{x};\mathbf{b}) = \sum_{i=1}^{N} b_i k(\mathbf{x},\mathbf{x}_i) + \epsilon \qquad (1)$$

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

where $k(\mathbf{x}, \mathbf{x}_i)$ is a bivariate *kernel* function centered on each of the N training data points $\mathbf{x}_i$, and $\mathbf{b} = \begin{bmatrix} b_1 & \dots & b_N \end{bmatrix}^T$ is a vector of regression coefficients. As in SVR, the goal of the RVM is to accurately predict the target function, while retaining as few basis functions as possible in the linear combination. This is achieved through the framework of *sparse Bayesian learning* and the introduction of prior distributions over the precisions α_i of each element of $\mathbf{b}$:

$$p(\mathbf{b}, \boldsymbol{\alpha}) = \prod_{i=1}^{N} \text{Normal}\left(b_i; 0, \alpha_i^{-1}\right) \text{Gamma}\left(\alpha_i; a_\alpha, b_\alpha\right) \tag{2}$$

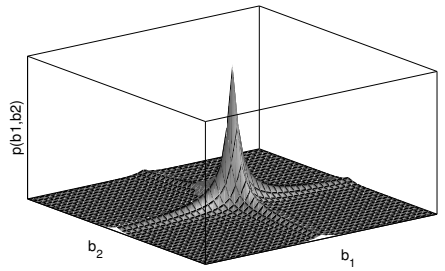


Figure 1. Marginal prior over regression coefficients

This form of prior results in the marginal distribution over $\mathbf{b}$ being a product of Student-t distributions as shown (for a 2-dimensional $\mathbf{b}$) in Fig. 1, and thus favors sparse solutions that lie along the *(hyper-)spines* of the distribution.

Approximate analytical solutions for the RVM can be obtained by the Laplace method (Tipping, 2001) or by using factorial variational approximations (Bishop & Tipping, 2000). However, in both these methods each update of the hyperparameters $\boldsymbol{\alpha}$ requires the re-estimation of the posterior distribution of $\mathbf{b}$ via an $O(N^3)$ Cholesky decomposition. As the number of data points increases, the RVM faces a similar explosion of computational requirements as that observed in Gaussian processes and support vector machines, since each new data point adds an extra "dimension" to the input vector. As will be shown in the next section, there are several alternatives for performing this regression step efficiently. In particular, our introduction of a probabilistic version of backfitting in this paper will show that we can achieve orders of magnitude improvement in the performance of the RVM, allowing the backfitting-RVM to tackle significantly larger data sets than previous methods.

1.2. High-Dimensional Regression

Algorithms for high-dimensional regression usually fall into one of two categories:

1. Those that try to find a low-dimensional representation of the data which captures the salient information required to perform the regression.

2. Those that deal with the complete dimensionality, but structure computations as efficiently as possible (such as performing successive inexpensive univariate regressions).

In the former category, methods like Principal Component Regression (PCR) and Factor Regression (FR) can be used to find a low-dimensional representation of the input data (Massey, 1965). Unfortunately, these methods are purely variance based, and do not take the output data into account when determining the relevant input dimensions. Thus, directions in input space which have large variance will be retained even if they have no influence on the prediction at all (Schaal et al., 1998). This drawback can be somewhat alleviated by performing the dimensionality reduction on the joint space of input and output data, and then conditioning on the observed input. Joint-space principal component regression (JPCR), and joint-space factor analysis for regression (JFR) are two such methods (Schaal et al., 1998), and more recently, the use of reproducing kernel Hilbert spaces (Fukumizu et al., 2004). The dimensionality reduction nevertheless typically requires expensive manipulation of covariance matrices of the data — an operation typically *cubic* in the assumed latent dimensionality.

Methods like partial least squares (PLS) (Wold, 1975) and backfitting (Hastie & Tibshirani, 1990) fall into the second category mentioned above (i.e. algorithms that structure computation efficiently). While PLS performs computationally inexpensive *univariate* regressions, along projection directions chosen according to *correlation* between input and output, backfitting creates *fake supervised targets* for successive inexpensive univariate regressions along each input dimension (see Algorithm 1). This effectively decouples inference in each individual dimension leading to a highly efficient (albeit iterative) algorithm which can be shown to be a generalized Gauss-Seidel procedure (Hastie & Tibshirani, 1990).

Although computationally extremely efficient, backfitting comes with a series of drawbacks, the most significant being that it has no probabilistic interpretation. This makes it difficult to insert into the framework of current research in Bayesian statistical learning which emphasizes model selection, and the estimation of confidence intervals. Another potential pitfall is that in even the simplest case of linear regression, backfitting provides no guarantees of convergence (Press et al., 1992). In Sec. 2, we will show that a simple modification to the standard graphical model for linear regres-

1: Init: $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T, \mathbf{y} = \left[y_1, \ldots, y_N\right], g_{m,i} = g_m(\mathbf{x}_i; \theta_m), \mathbf{g}_m = [g_{m,1}, \ldots, g_{m,N}]^T$
2: **repeat**
3: **for** $m = 1$ to d **do**
4: $\mathbf{r}_m \leftarrow \mathbf{y} - \sum_{k \neq m} \mathbf{g}_k$ *//fake target*
5: $\theta_m \leftarrow \arg\min_{\theta_m} \|\mathbf{g}_m - \mathbf{r}_m\|^2$
6: **end for**
7: **until** convergence of $\boldsymbol{\theta}_m$

Algorithm 1: The backfitting algorithm works with a linear combination of basis functions $g_m(\mathbf{x}_i; \theta_m)$ that are iteratively updated to fit fake targets formed by partial residuals.

sion allows us to derive a probabilistic version of backfitting which is guaranteed to converge by virtue of the convergence properties of the EM algorithm. Subsequently, this allows us to augment the model with appropriate prior distributions to enable an automatic determination of which input dimensions are relevant to the regression. This in turn gives us the foundation for our reformulation of sparse Bayesian learning in Sec. 4.

2. Probabilistic Backfitting

By introducing the notion of *fake supervised targets*, backfitting decouples the inference in each input dimension, creating an efficient regression algorithm. This section shows that by treating these supervised targets as hidden variables in an EM algorithm, we can derive a probabilistic version of backfitting, which provides the same computational advantages as traditional backfitting, but with the added bonus of a probabilistic interpretation, and convergence properties that stem from its EM formulation.

Fig. 2(a) shows the graphical model for generalized linear regression, according to the following equation:

$$y(\mathbf{x}) = \sum_{m=1}^{d} b_m f_m(\mathbf{x}; \boldsymbol{\theta}_m) + \epsilon$$

i.e., multiple predictors $f_m(\mathbf{x}; \boldsymbol{\theta}_m)$ (where $1 \leq m \leq d$) that are generated by an adjustable non-linear transformation with parameters $\boldsymbol{\theta}_m$ and are fed linearly to an output y by an inner product with a regression vector $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \cdots & b_d \end{bmatrix}^T$ plus additive noise ϵ. It is easy to see that the optimal estimate of the regression parameters (in the least-squares or maximum-likelihood sense) is given by the Ordinary Least Squares (OLS) solution $\mathbf{b}_{\text{OLS}} = \left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T\mathbf{y}$, where $\mathbf{F}$ denotes a matrix whose rows contain the $f_m(\mathbf{x}_i)$ of all the training data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. With a growing number of fan-in variables in the graphical model (or equivalently, an increasing input dimensionality d), evaluation of the OLS solution becomes increasingly computationally expensive (approximately $O(d^3)$) and numerically brittle.

Consider the introduction of a random variable z_{im} which is analogous to the output of the g_m function of Algorithm 1, where we define $g_m(\mathbf{x}; \theta_m) = b_m f_m(\mathbf{x}; \theta_m)$. For the derivation of our algorithm, we assume that z_{im} is conditionally normally distributed, $z_{im}|\mathbf{x}_i \sim \text{Normal}\left(z_{im}; g_m(\mathbf{x}_i), \psi_{zm}\right)$. The introduction of the z_{im} variables modifies the graphical model to that in Fig. 2(b), which we can formally describe for every data point i as follows:

$$y_i|\mathbf{z}_i \sim \text{Normal}\left(y_i; \mathbf{1}^T\mathbf{z}_i, \psi_y\right)$$
$$z_{im}|\mathbf{x}_i \sim \text{Normal}\left(z_{im}; b_m f_m(\mathbf{x}_i), \psi_{zm}\right)$$

where $\mathbf{1} = [1, 1, \ldots, 1]^T$. It needs to be emphasized that now, the regression coefficients b_m are *behind* the fan-in of the graphical model.

Given the data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, and the graphical model of Fig. 2(b), we wish to estimate the parameters b_m and (possibly) optimize the individual functions $f_m(\mathbf{x}; \theta_m)$ with respect to the parameters θ_m. This is easily formulated as an EM algorithm, which maximizes the *incomplete* log likelihood $\log p(\mathbf{y}|\mathbf{X})$:

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{N}{2}\log\psi_y - \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{b}^T\mathbf{f}(\mathbf{x}_i)\right)^2 + const \quad (3)$$

by maximizing the expected *complete* log likelihood $\langle \log p(\mathbf{y}, \mathbf{Z}|\mathbf{X}) \rangle$, where:

$$\log p(\mathbf{y}, \mathbf{Z}|\mathbf{X}) = -\frac{N}{2}\log\psi_y - \frac{1}{2\psi_y}\sum_{i=1}^{N}\left(y_i - \mathbf{1}^T\mathbf{z}_i\right)^2 - \sum_{m=1}^{d}\left[\frac{N}{2}\log\psi_{zm} + \frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\left(z_{im} - b_m f_m(\mathbf{x}_i; \boldsymbol{\theta}_m)\right)^2\right] + const \quad (4)$$

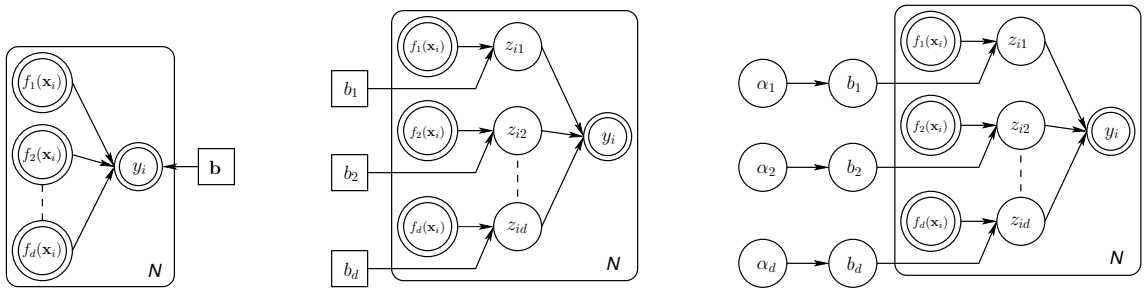
As this maximization is solely based on standard manipulations of normal distributions, we omit derivations and just summarize the EM update equations for b_m and the noise variances ψ_y and ψ_{zm} as follows:

M-Step :

$$b_m = \frac{\sum_{i=1}^{N}\langle z_{im}\rangle f_m(\mathbf{x}_i)}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2}$$

$$\psi_y = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \mathbf{1}^T\langle \mathbf{z}_i\rangle\right)^2 + \mathbf{1}^T\boldsymbol{\Sigma}_{\mathbf{z}}\mathbf{1}$$

$$\psi_{zm} = \frac{1}{N}\sum_{i=1}^{N}\left(\langle z_{im}\rangle - b_m f_m(\mathbf{x}_i)\right)^2 + \sigma_{zm}^2$$



(a) Standard linear regression

(b) Probabilistic backfitting

(c) Backfitting with ARD

Figure 2. Graphical models for backfitting. Circular nodes represent random variables, with a double circle denoting observed variables. Square nodes denote point estimated parameters.

E-Step :

$$\mathbf{1}^T\mathbf{\Sigma_z}\mathbf{1} = \left(\sum_{m=1}^{d}\psi_{zm}\right)\left[1-\frac{1}{s}\left(\sum_{m=1}^{d}\psi_{zm}\right)\right]$$

$$\sigma_{zm}^2 = \psi_{zm}\left(1-\frac{1}{s}\psi_{zm}\right)$$

$$\langle z_{im}\rangle = b_m f_m(\mathbf{x}_i) + \frac{1}{s}\psi_{zm}\left(y_i - \mathbf{b}^T\mathbf{f}(\mathbf{x}_i)\right)$$

where we define $s = \psi_y + \sum_{m=1}^{d}\psi_{zm}$, and $\mathbf{\Sigma_z} = \mathrm{Cov}(\mathbf{z}|\mathbf{y},\mathbf{X})$. In addition, the parameters θ_m of each function f_m can be updated by setting $\sum_{i=1}^{N}\left(\langle z_{im}\rangle - b_m f_m\left(\mathbf{x}_i;\theta_m\right)\right)\frac{\partial f_m(\mathbf{x}_i;\theta_m)}{\partial\theta_m} = 0$ and solving for θ_m. As this step depends on the particular choice of f_m, e.g., splines, kernel smoothers, parametric models, etc., we will not pursue it any further in this paper and just note that *any* statistical approximation mechanism could be used.

Two items in the above EM algorithm are of special interest. First, all equations are algorithmically $O(d)$ where d is the number of predictor functions f_m. Second, if we substitute the expression for $\langle z_{im}\rangle$ in the maximization equation for b_m we get the following update equation:

$$b_m^{(n+1)} = b_m^{(n)} + \frac{\psi_{zm}}{s}\frac{\sum_{i=1}^{N}\left(y_i - \sum_{k=1}^{d} b_k^{(n)} f_k(\mathbf{x}_i)\right) f_m(\mathbf{x}_i)}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2} \tag{5}$$

Thus each EM cycle updates the mth regression coefficient by an amount proportional to the correlation between the mth predictor and the residual error. Hence the residual can be interpreted as forming a "fake target" for the mth branch of the fan-in, which is similar to the way PLS regresses residual errors against individual input projections — indeed, our algorithm can also be interpreted as a probabilistic version of PLS. As the next section shows, this enables us to place this algorithm in the context of *backfitting*.

2.1. Interpreting the EM Solution as Probabilistic Backfitting

In the context of understanding Eq. (5) as Probabilistic Backfitting, we note that backfitting can be viewed as a formal Gauss-Seidel algorithm; an equivalence that becomes exact in the special case of linear models (Hastie & Tibshirani, 1990). For the linear system $\mathbf{F}^T\mathbf{F}\mathbf{b} = \mathbf{F}^T\mathbf{y}$, the Gauss-Seidel updates for the individual b_m are:

$$b_m = \frac{\sum_{i=1}^{N}\left(y_i - \sum_{k\neq m}^{d} b_k f_k(\mathbf{x}_i)\right) f_m(\mathbf{x}_i)}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2} \tag{6}$$

A well-known extension to the Gauss-Seidel algorithm called *successive relaxation* adds a fraction $(1-\omega)$ of b_m to the update and giving us:

$$b_m^{(n+1)} = (1-\omega)b_m^{(n)} + \omega\frac{\sum_{i=1}^{N}\left(y_i - \sum_{k\neq m}^{d} b_k f_k(\mathbf{x}_i)\right) f_m(\mathbf{x}_i)}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2} \tag{7}$$

which has improved convergence rates for *overrelaxation* ($1 < \omega < 2$), or improved stability for *underrelaxation* ($0 < \omega < 1$). For $\omega = 1$, the standard Gauss-Seidel/backfitting of Eq. (6) is recovered. Setting $\omega = \omega_m = \psi_{zm}/s$ in Eq. (7), it can be shown that (after some algebraic rearrangement,) we obtain exactly our EM update in Eq. (5), i.e., we indeed derive a probabilistic version of backfitting as an underrelaxation method.

Notably, the original backfitting procedure makes no guarantees about convergence. However, it is easy to show that due to the convergence properties of EM, the probabilistic backfitting procedure is guaranteed to converge to an OLS solution.

We note that in general, there exist other algorithms that can iteratively arrive at a solution to a linear sys-

tem of equations such as the method of conjugate gradients, which can also be related algorithmically to Jacobi iterations and Gauss-Seidel relaxation methods. Importantly, our formulation shows that the rich family of methods that can be related to the backfitting algorithm, and that until now did not have a probabilistic derivation, can now be represented within the probabilistic framework of an iterative EM algorithm. This is an important stepping stone, since — as the next section shows — these algorithms may now benefit from the model regularizing features of Bayesian inference.

3. Bayesian Backfitting

Modifying Fig. 2(b) slightly, we now place individual precision variables α_m over each of the regression parameters b_m, resulting in Fig. 2(c). This model structure can be captured by the following set of prior distributions (c.f. Eq. (2)):

$$\begin{aligned} p(\mathbf{b}|\boldsymbol{\alpha}) &= \prod_{m=1}^{d} \left(\frac{\alpha_m}{2\pi}\right)^{1/2} \exp\left\{-\frac{\alpha_m}{2} b_m^2\right\} \\ p(\boldsymbol{\alpha}) &= \prod_{m=1}^{d} \frac{b_\alpha^{a_\alpha}}{\Gamma(a_\alpha)} \alpha_m^{(a_\alpha - 1)} \exp\left(-b_\alpha \alpha_m\right) \end{aligned} \tag{8}$$

Using a factorial variational approximation (e.g. Ghahramani & Beal, 2000), we can derive the modified update equations for the variables in the model. Due to space constraints, we omit the derivation, and only summarize the update equations for the mean of $\mathbf{b}$ and $\boldsymbol{\alpha}$:

$$\begin{aligned} \langle b_m \rangle^{(n+1)} = &\left(\frac{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \langle \alpha_m \rangle}\right) \langle b_m \rangle^{(n)} \\ &+ \frac{\psi_{zm} \sum_{i=1}^{N} \left(y_i - \langle \mathbf{b} \rangle^{(n)^T} \mathbf{f}(\mathbf{x}_i)\right) f_m(\mathbf{x}_i)}{s\left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \langle \alpha_m \rangle\right)} \end{aligned} \tag{9}$$

$$\langle \alpha_m \rangle = \frac{2a_\alpha + 1}{2b_\alpha + \langle b_m^2 \rangle} \tag{10}$$

Comparing Eqs. (9) and (5) we see that in the absence of a correlation between the current input dimension and the residual error, the first term of Eq. (9) causes the current regression coefficient to decay. This results in a regression solution which regularizes over the *number of retained input dimensions* in the final regression vector, similar to Automatic Relevance Determination (ARD) (Neal, 1994). As an aside, it is useful to note that if we chose to put a *single* precision variable over the entire regression vector $\mathbf{b}$ then our model reduces to *ridge regression*, with the Bayesian model selection process providing an automatic tuning of the ridge parameter.

3.1. Bayesian Backfitting Evaluation

Rather than immediately derive the backfitting RVM, we will momentarily digress to underscore the efficacy of Bayesian backfitting as a robust and efficient linear regression procedure. We compare the use of PLS and Bayesian backfitting as described in Sec. 3 to analyze the following real-world data set collected from neuroscience. Our choice of PLS for comparison was motivated by the fact that this is a well-studied algorithm that also has $O(d)$ complexity, and is widely used on data sets in chemometrics with similar properties. The data set consists of simultaneous recordings (2400 data points) of firing-rate coded activity in 71 motor cortical neurons and the EMG of 11 muscles. The goal is to determine which neurons are responsible for the activity of each muscle. The relationship between neural and muscle activity is assumed to be linear, such that the basis functions in backfitting are simply a copy of the respective input dimensions, i.e. $f_m(\mathbf{x}) = x_m$.

A brute-force study (conducted by our research collaborators) painstakingly considered every possible combination of neurons (up to groups of 20 for computational reasons, i.e. even this reduced analysis required several weeks of computation on a 30-node cluster computer), to determine the optimal neuron-muscle correlation as measured on various validation sets. This study provided us with a baseline neuron-muscle correlation matrix that we hoped to duplicate with PLS and Bayesian backfitting, although with much reduced computational effort.

	Bayes. back.	**PLS**	**baseline**
neuron match	93.6%	18%	—
nMSE	0.8446	1.77	0.84

Table 1. Results on the neuron-muscle data set

The results shown in Table 1 demonstrate two points:

- The relevant neurons found by Bayesian backfitting contained over 93% of the neurons found by the baseline study, while PLS fails to find comparable correlations. The neuron match in backfitting is easily inferred from the resulting magnitude of the precision parameters α, while for PLS, the neuron match was inferred based on the subspace spanned by the projections that PLS employed.
- The regression accuracy of Bayesian backfitting (as determined by 8-fold cross-validation), is com-

parable to that of the baseline study, while PLS' failure to find the correct correlations causes it to have significantly higher generalization errors. The analysis for both backfitting and PLS was carried out using the same validation sets as those used for the baseline analysis.

The performance of Bayesian backfitting on this particularly difficult data set shows that it is a viable alternative to traditional generalized linear regression tools. Even with the additional Bayesian inference for ARD, it maintains its algorithmic efficiency since no matrix inversion is required.

As an aside it is useful to note that Bayesian backfitting and PLS required of the order of 8 hours of computation on a standard PC[1] (compared with several weeks on a cluster for the brute-force study), and evaluated the contributions of all 71 neurons.

4. Bayesian Backfitting RVM

Until now, we have chosen not to comment on the nature of the basis functions $f_m(\mathbf{x})$ in our model. Let us now switch to the RVM framework in which we create N basis functions by centering a bivariate kernel function $k(\mathbf{x}, \mathbf{x}')$ on each individual data point. This implies:

$$f_m(\cdot) = k(\cdot, \mathbf{x}_m)$$

for $1 \leq m \leq d$ and where we now have $d = N$. Notice that this transformation makes our backfitting model of Fig. 2(c) equivalent to the RVM model discussed in Sec. 1.1, with the notable difference that backfitting allows a significant advantage over the standard RVM in computational complexity. Note however, that while the computational complexity of a backfitting update is linear in the dimensionality of the problem, it is also linear in the number of data points i.e. $O(Nd)$. When cast into the RVM framework, setting $d = N$ makes this complexity $O(N^2)$. In particular we would like to stress the following:

- At *each* update of the α_m hyperparameters, the RVM requires an $O(N^3)$ Cholesky decomposition to re-estimate the regression parameters, while discarding the estimate at the previous iteration. In the backfitting-RVM however, the existing estimate of the regression parameters provides a good starting estimate, allowing the update to complete in just a handful of $O(N^2)$ iterations (~ 10 iterations were sufficient in our simulations). The saving in computation is especially evident when the number of data points (and hence the effective dimensionality) is large, and in situations where the hyperparameters require many updates before convergence.
- In the initial computations within the graphical model, it seems wasteful to spend large amounts of computation on estimating parameters accurately, when surrounding parameters (and hyperparameters) have not converged. One can structure the backfitting updates to work with partially converged estimates, such that the brunt of computation is only expended to accurately estimate a variable when one is more confident about the variables in its Markov blanket.

[1]Pentium IV class machine, 1.7GHz

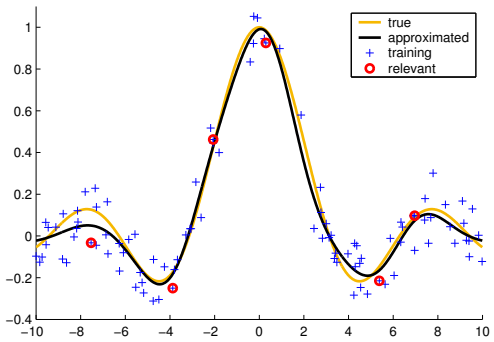


Figure 3. Fitting the sinc function using backfitting-RVM.

Fig. 3 shows backfitting-RVM used to fit a toy data set generated using the 1-dimensional sinc function $\sin(x)/x$, using the Gaussian kernel:

$$k(x_i, x_j) = \exp\left\{-\lambda\,(x_i - x_j)^2\right\}$$

for $\lambda > 0$. Even though backfitting-RVM is an order of magnitude faster than the standard RVM, it suffers no penalty in generalization error or its ability to sparsify the set of basis functions. We note that Tipping (2001) proposes an optimization of the distance metric λ that is based on gradient ascent in the log likelihood. Such a gradient can also be computed for backfitting as:

$$\frac{\partial \langle \log p(\mathbf{y}, \mathbf{Z}|\mathbf{X}) \rangle}{\partial \lambda} = \sum_{j=1}^{N} \frac{b_j}{\psi_{zj}} \sum_{i=1}^{N} \left(\langle z_{ij} \rangle - b_j k_{ij}\right) (x_i - x_j)^2 \, k_{ij}$$

where we have abbreviated $k_{ij} = k(x_i, x_j)$. Based on our experience however, we would like to caution against unconstrained maximization of the likelihood, especially over distance metrics. Instead, we would recommend the route taken in the Gaussian process community, which is to treat these variables as hyperparameters, and place prior distributions over them.

Exact solutions being typically intractable, we can either optimize them by using *maximum a posteriori* estimates (MacKay, 1999), or by Monte Carlo techniques (Williams & Rasmussen, 1996).

We note that there are several "optimizations" that are suggested in (Tipping, 2001; Tipping & Faul, 2003). These include pruning the basis functions when their precision variables dictate that they are unneeded, as well as adopting a *greedy* (but potentially suboptimal) strategy in which the algorithm starts with a single basis function and adds candidates as necessary. We would like to emphasize that our implementation of the backfitting-RVM performs neither of these optimizations, although it is trivial to introduce them into our framework as well.

4.1. Backfitting-RVM Evaluation

To evaluate the generalization ability of backfitting-RVM, we compared it to other state-of-the art regression tools on the popular benchmark Boston housing and Abalone data sets[2]. For each data set, a randomly selected 20% of the data set was used as test data and the remainder for training.

	RVM	SVR	GP	LWPR	bRVM
Sinc	0.0132	0.0178	0.0136	0.0124	0.0130
Boston	0.0882	0.1115	0.0806	0.0846	0.0837
Abalone	0.4591	0.4830	0.4440	0.4056	0.4473

Table 2. nMSE on benchmark data sets

Table 2 shows the normalized mean squared errors on the test sets averaged over 100 experiments. The algorithms compared were the standard relevance vector machine (RVM), support vector regression (SVR)[3], Gaussian process (GP) regression, locally weighted projection regression (LWPR) (Vijayakumar & Schaal, 2000), and our backfitting-RVM (bRVM). Both backfitting-RVM and its standard counterpart used Gaussian kernels with distance metrics optimized by 5-fold cross-validation. The Gaussian process algorithm used RBF covariance function with automatic hyperparameter optimization. As Table 2 shows, backfitting-RVM provides an extremely competitive solution in terms of generalization ability when compared to other popular regression methods.

	RVM	SVR	bRVM
Sinc	6.7	45.2	4.8
Boston	39	142.8	57.4
Abalone	437	1320	368

Table 3. "Relevant" vectors retained

For the 3 methods (RVM, SVR, and bRVM) that focus on a "sparsification" of the set of basis functions, we compared the average number of basis functions retained on two data sets: the Boston housing, and sinc data sets. To aid comparison, data for the sinc benchmark was generated using a method identical to that specified in (Tipping, 2001). Table 3 shows the average number of vectors retained in the final solution on these data sets.

	RVM	bRVM	N	d
Sinc	18.71s	6.24s	100	1
Boston	372s	155s	481	13
Abalone	2767s	428s	3341	10

Table 4. Relative computation time

The above experiments demonstrate that backfitting-RVM is a competitive regression solution when compared to other current state-of-the-art statistical methods, both in its generalization ability, and in its efficacy as a sparse Bayesian learning algorithm. However, the main advantage of backfitting-RVM is apparent only when we examine its relative computation time. Table 4 gives the average execution time (in seconds) required by the RVM, and backfitting-RVM for convergence of their regression parameter estimates (to 5 significant digits) on the sinc, Boston housing, and Abalone data sets. The table also shows the number of *training* data points, and their dimensionality. Note that the number of $O(N^2)$ updates to $\mathbf{b}$ per update cycle of the hyperparameters is very small (~ 10), since the solution from the previous update cycle is a very good starting point for the iterations of the next cycle. The results demonstrate that the backfitting-RVM can significantly gain from the iterative nature of the Bayesian backfitting generalized linear regression procedure.

5. Discussion

Given the form $y = \sum_{i=1}^{N} b_i k(\cdot, x_i)$ of the RVM solution, it is natural to make a connection to Gaussian processes, which also express the solution as a linear combination of basis functions centered at the training data points. Indeed, retaining only the relevant vectors amounts to a *sparsification* of the Gaussian process. Our algorithm is equivalent to *pruning* the set of basis functions, as is also achieved using the Nyström method (Williams & Seeger, 2001). Other variants exist such as the *growing/replacement* solution of Csató and Opper (2001), which generalizes well to online learning scenarios.

This paper makes two essential contributions. Firstly, we have demonstrated that the class of traditionally non-Bayesian, yet highly efficient iterative linear regression methods like backfitting, can be derived from

[2]Both available from the UCI repository

[3]RVM and SVR results adapted from (Tipping, 2001)

the framework of the EM algorithm. We have derived Bayesian backfitting, which retains *linear* complexity in the dimensionality of the input data, even while performing ARD-like model selection. On its own, Bayesian backfitting provides a very general framework for generalized linear regression, which is numerically robust and is able to handle extremely high-dimensional datasets. At the expense of being an iterative algorithm, it is a viable drop-in replacement for algorithms such as PLS, stepwise regression, singular value decomposition regression, and others mentioned in Sec. 1.2. While requiring the assumption of Gaussian distributions at certain steps of the derivation of Bayesian backfitting, our experience with its use on data sets that violate these assumptions has shown no significant degradation in performance or generalization ability.

Secondly, we have shown that the framework of sparse Bayesian learning can benefit immensely from a probabilistic formulation of this iterative class of methods. In particular, the popular relevance vector machine can be derived from the framework of Bayesian backfitting. This backfitting-RVM has significant computational advantages over its conventional counterpart, without sacrificing generalization and model regularization ability. Although the examples presented here focus on regression, it is easy to see that by using similar variational extensions (Jaakkola & Jordan, 2000) as those used in (Bishop & Tipping, 2000), the applicability of the backfitting-RVM can be extended to classification tasks as well.

Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, a NASA grant AC#98-516, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, and the ATR Computational Neuroscience Laboratories.

References

Bishop, C. M., & Tipping, M. E. (2000). Variational relevance vector machine. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence* (pp. 46–53). Morgan Kaufmann Publishers.

Csató, L., & Opper, M. (2001). Sparse representation for Gaussian process models. In (Leen et al., 2001), 444–450.

Fukumizu, K., Bach, F. R., & Jordan, M. I. (2004). Dimensionality reduction for supervised learning using reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, *5*, 73–99.

Ghahramani, Z., & Beal, M. J. (2000). Variational inference for Bayesian mixtures of factor analysers. *Advances in Neural Information Processing Systems 12* (pp. 509–514). Cambridge, MA: MIT Press.

Hastie, T. J., & Tibshirani, R. J. (1990). *Generalized additive models*. No. 43 in Monographs on Statistics and Applied Probability. Chapman & Hall.

Jaakkola, T. S., & Jordan, M. I. (2000). Bayesian parameter estimation via variational methods. *Statistics and Computing*, *10*, 25–37.

Leen, T. K., Diettrich, T. G., & Tresp, V. (Eds.). (2001). *Advances in neural information processing systems 13*, vol. 13. Cambridge, MA: MIT Press.

MacKay, D. J. C. (1999). Comparison of approximate methods for handling hyperparameters. *Neural Computation*, *11*, 1035–1068.

Massey, W. F. (1965). Principal component regression in exploratory statistical research. *Journal of the American Statistical Association*, *60*, 234–246.

Neal, R. M. (1994). *Bayesian learning for neural networks*. Doctoral dissertation, Dept. of Computer Science, University of Toronto.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: The art of scientific computing*. Cambridge University Press. 2 edition.

Schaal, S., Vijayakumar, S., & Atkeson, C. G. (1998). Local dimensionality reduction. *Advances in Neural Information Processing Systems 10* (pp. 633–639). Cambridge, MA: MIT Press.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–244.

Tipping, M. E., & Faul, A. C. (2003). Fast marginal likelihood maximization for sparse Bayesian models. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*.

Vijayakumar, S., & Schaal, S. (2000). An $O(n)$ algorithm for incremental real time learning in high dimensional space. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000)* (pp. 1079–1086). Stanford, CA.

Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. *Advances in Neural Information Processing Systems 8* (pp. 514–520). Cambridge, MA: MIT Press.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In (Leen et al., 2001), 682–688.

Wold, H. (1975). Soft modeling by latent variables: The nonlinear iterative partial least squares approach. In J. Gani (Ed.), *Perspectives in probability and statistics, papers in honour of M. S. Bartlett*, 520–540. London: Academic Press.