ARTICLE Communicated by Kechen Zhang

Efficient Learning and Feature Selection in High-Dimensional Regression

Jo-Anne Ting
jting@acm.org
University of Edinburgh, Edinburgh EH8 9AB, U.K.

Aaron D'Souza
adsouza@google.com
Google Inc., Mountain View, CA 94043, U.S.A.

Sethu Vijayakumar
sethu.vijayakumar@ed.ac.uk
University of Edinburgh, Edinburgh EH8 9AB, U.K.

Stefan Schaal
sschaal@usc.edu
University of Southern California, Los Angeles, CA 90089, U.S.A., and
ATR Computational Neuroscience Laboratories, Kyoto, Japan

We present a novel algorithm for efficient learning and feature selection in high-dimensional regression problems. We arrive at this model through a modification of the standard regression model, enabling us to derive a probabilistic version of the well-known statistical regression technique of backfitting. Using the expectation-maximization algorithm, along with variational approximation methods to overcome intractability, we extend our algorithm to include automatic relevance detection of the input features. This variational Bayesian least squares (VBLS) approach retains its simplicity as a linear model, but offers a novel statistically robust black-box approach to generalized linear regression with high-dimensional inputs. It can be easily extended to nonlinear regression and classification problems. In particular, we derive the framework of sparse Bayesian learning, the relevance vector machine, with VBLS at its core, offering significant computational and robustness advantages for this class of methods. The iterative nature of VBLS makes it most suitable for real-time incremental learning, which is crucial especially in the application domain of robotics, brain-machine interfaces, and neural prosthetics, where real-time learning of models for control is needed. We evaluate our algorithm on synthetic and neurophysiological data sets, as well as on standard regression and classification benchmark data sets, comparing it with other competitive statistical approaches and demonstrating its suitability as a drop-in replacement for other generalized linear regression techniques.

1 Introduction

Real-world data such as those obtained from neuroscience, chemometrics, data mining, or sensor-rich environments are often extremely high-dimensional, severely underconstrained (few data samples compared to the dimensionality of the data), and interspersed with a large number of irrelevant or redundant features. Combined with inevitable measurement noise, efficient learning from such data still poses a significant challenge to state-of-the-art supervised learning algorithms, even in linear settings. We are especially interested in scenarios where a large number of high-dimensional samples need to be processed, potentially in a real-time, incremental fashion (i.e., where d is the number of input dimensions, N is the number of samples in the data set, and $d < N$). While traditional statistical techniques for supervised learning (e.g., partial least squares regression, backfitting) are often efficient and robust for these problems, they lack a probabilistic interpretation and cannot easily provide measures needed for model selection such as the evidence of the data or predictive distributions. Yet while recent statistical learning algorithms in supervised learning compute such information, some lack computational efficiency as, for instance, in gaussian process regression or classical implementations of support vector learning, especially in the $d < N$ scenarios we are interested in.

Our article introduces a new algorithm, variational Bayesian least squares (VBLS), that possesses both efficiency and a sound probabilistic foundation. It is derived by developing a Bayesian formulation of a classical nonparametric, nonprobabilistic regression algorithm. We demonstrate that the algorithm can significantly improve the computational efficiency of sparse Bayesian learning while performing feature detection and automatic relevance determination. Additionally, the algorithm avoids any potentially expensive cross-validation or tuning of meta parameters by the user, offering a statistically robust, "automatic" method that can be applied across data sets from various systems and incorporated into more complex learning algorithms. In this way, we can apply this technique to very high-dimensional problems in both linear and nonlinear scenarios.

The algorithm can be interpreted as a Bayesian version of backfitting that does not require any sampling, making it suitable for implementation in incremental form for real-time applications (e.g., as in application domains such as robotics, brain-machine interfaces, and tracking systems) and for embedment in other iterative methods.

The iterative nature of VBLS is invaluable in real-time situations where decisions need to be made quickly, such that an approximate solution is acceptable. In these scenarios, waiting a longer time for a very accurate solution may not be an acceptable alternative. Additionally, VBLS is most advantageous when embedded in other iterative methods, offering a significant computational improvement.

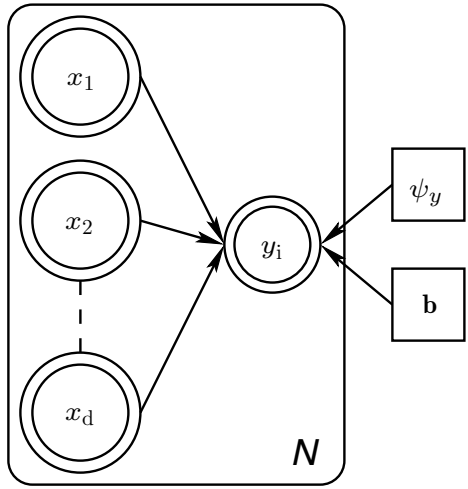


Figure 1: Graphical model for linear regression. Note the fan-in, which causes the estimates of the individual regression coefficients b_m to become coupled in the posterior.

We start by discussing some of the popular approaches for supervised learning of high-dimensional, underconstrained data, examining methods such as principal component regression, backfitting, partial least-squares regression, and least absolute shrinkage and selection operator regression, to name a few. Then, in section 3, we derive backfitting from a probabilistic model and solve it within the expectation-maximization (EM) framework. Third, we extend the model to incorporate structured priors, as described in section 4, allowing us to exploit Bayesian model selection. Using a variational approximation technique, we arrive at a new algorithm, VBLS, that can be applied to both regression and classification problems. Section 5 shows that while VBLS is derived within the context of a linear model, it is also a powerful tool for supervised learning in nonlinear settings. Finally, we evaluate VBLS on high-dimensional synthetic and real data sets, demonstrating its significant computational advantages over other competitive statistical learning methods.

2 Computationally Tractable Linear Regression

We begin by examining the graphical model for linear regression, as shown in Figure 1, which corresponds to the following generative model:

$$y = \mathbf{b}^T \mathbf{x} + \epsilon, \tag{2.1}$$

where, for successive samples from this model, we assume the ϵ are independent and identically distributed (i.i.d.) as $\epsilon \sim \text{Normal}(\epsilon; 0, \psi_y)$. Given a data set of observed tuples $\mathbf{x}_D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, our goal is to estimate the optimal linear coefficients $\mathbf{b} = [b_1\ b_2\ \cdots\ b_d]^T$, which combine the input dimensions to produce the output y.

It is easy to see that under our current noise model, the optimal estimate of the regression parameters (in a least squares or maximum likelihood sense) is given by

$$\mathbf{b}_{\text{OLS}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{2.2}$$

where $\mathbf{X}$ denotes a matrix whose rows contain the $\mathbf{x}_i$ and $\mathbf{y}$ is a column vector containing the corresponding y_i. Equation 2.2 is also known as the ordinary least squares (OLS) solution. A fan-in of the type observed from $\mathbf{x}$ to y in Figure 1 couples all the regression coefficients in the posterior inference—a fact reflected in the need to evaluate the covariance matrix $\mathbf{X}^T\mathbf{X}$ in equation 2.2. With an increasing number of fan-in variables in the graphical model (or, equivalently, an increasing input dimensionality d), evaluation of the solution in equation 2.2 becomes increasingly computationally expensive (approximately $O(d^3)$) and numerically brittle. While one can attempt to reduce the complexity to $O(d^2)$ with efficient matrix inversion techniques (Belsley, Kuh, & Welsch, 1980), solutions to this problem typically fall into one of two categories:

1. *Dimensionality reduction for regression*: Those that try to find a low-dimensional, full-rank representation of the data that captures the salient information required to perform the regression.
2. *Data structures for fast statistics*: Those that deal with the complete dimensionality but structure computations as efficiently and robustly as possible (e.g., by performing successive inexpensive univariate regressions).

In the following subsections, assume we are given a data set $\{\mathbf{X}, \mathbf{y}\}$ with inputs $\mathbf{X} = [\mathbf{x}_1 \ldots \mathbf{x}_N]^T$ consisting of d-dimensional vectors $\mathbf{x}_i$ (where $i = 1, 2, \ldots, N$ and N is the number of data samples) and outputs $\mathbf{y} = [y_1 \ldots y_N]^T$ consisting of scalars y_i. Without any loss of generality, we shall assume that both $\mathbf{X}$ and $\mathbf{y}$ are mean zero. We want to find the vector $\mathbf{b}$ of regression coefficients that linearly combine the input $\mathbf{x}$ to predict y.

In an attempt to perform a comprehensive (although not exhaustive) review of the literature, we discuss some examples of algorithms that are representative of the two categories described above.

2.1 Dimensionality Reduction for Regression. Often the information relevant to predicting the output y can be localized to a low-dimensional manifold within the domain of $\mathbf{x}$. The methods discussed in this section rely

on the assumption that when a dimensionality reduction on the input space is performed, the resulting lower-dimensional manifold captures sufficient information to accurately predict the output.

2.1.1 Principal Component Regression. The underlying basis of principal component regression (PCR) (Massey, 1965) is that the low-dimensional subspace that explains the most variance in the $\mathbf{x}$ also captures the most essential information required to predict y. Starting with the empirical covariance matrix $\mathbf{\Sigma}_{\text{PCR}}$ of the input data,

$$\mathbf{\Sigma}_{\text{PCR}} = \frac{1}{N-1} \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T, \tag{2.3}$$

we compute its eigendecomposition,

$$\mathbf{\Sigma}_{\text{PCR}} \mathbf{v}_j = \lambda_j \mathbf{u}_j, \tag{2.4}$$

where $\mathbf{u}_j$ is the jth eigenvector and λ_j the corresponding eigenvalue. By projecting the input $\mathbf{x}$ onto the principal K eigenvectors using the projection matrix $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2, \ldots, \mathbf{u}_K]$, we can compute the regression solution as follows:

$$\mathbf{b}_{\text{PCR}} = \left(\mathbf{U}^T \mathbf{X}^T \mathbf{X} \mathbf{U}\right)^{-1} \mathbf{U}^T \mathbf{X}^T \mathbf{y}. \tag{2.5}$$

Note that as a result of the projection onto the orthogonal eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$, the matrix $\left(\mathbf{U}^T \mathbf{X}^T \mathbf{X} \mathbf{U}\right)$ in equation 2.5 is diagonal, and hence, trivial to invert—the brunt of the computation having already been expended in the eigendecomposition step. As a result, PCR essentially reduces the multivariate regression to a set of independent univariate regressions along each of the orthogonal principal component directions.

A serious drawback of PCR is that it is based purely on variance in the input data (Schaal, Vijayakumar, & Atkeson, 1998). The regression solution is therefore highly sensitive to preprocessing operations such as sphering, which modify the perceived variance of each input dimension. Hence, low-variance input dimensions that are nevertheless important predictors of the output may be discarded in favor of high-variance but irrelevant dimensions. If, however, we operate on the joint space $\mathbf{z} = [\mathbf{x}^T y]^T$ of the data, we can take the output into consideration when determining the appropriate lower-dimensional manifold.

2.1.2 Joint-Space Factor Analysis for Regression. Factor analysis (Everitt, 1984; Ghahramani & Hinton, 1997) is a density estimation technique that assumes that the observed data $\mathbf{z}$ are generated from a lower-dimensional

process characterized by K latent or hidden variables $\mathbf{v}$ as follows:

$$\mathbf{z}_i = \mathbf{W}\mathbf{v}_i + \boldsymbol{\epsilon}_i \qquad \text{where } 1 \leq i \leq N. \tag{2.6}$$

If we assume that the latent variables are independently distributed as

$$\mathbf{v}_i \sim \text{Normal}(\mathbf{v}_i; \mathbf{0}, \mathbf{I})^1$$
$$\boldsymbol{\epsilon}_i \sim \text{Normal}(\boldsymbol{\epsilon}_i; \mathbf{0}, \boldsymbol{\Psi}),$$

then the parameters $\mathbf{W}$ and $\boldsymbol{\Psi}$ can be easily estimated using maximum likelihood (Ghahramani & Hinton, 1997) or Bayesian (Ghahramani & Beal, 2000a) techniques. In joint-space factor analysis for regression (JFR), we define

$$\mathbf{z} \equiv \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix} \quad \text{and} \quad \mathbf{W} \equiv \begin{bmatrix} \mathbf{W}_\mathbf{x} \\ \mathbf{W}_y \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Psi} \equiv \begin{bmatrix} \boldsymbol{\Psi}_\mathbf{x} & \mathbf{0} \\ \mathbf{0}^T & \psi_y \end{bmatrix}. \tag{2.7}$$

Once we estimate $\mathbf{W}$ and $\boldsymbol{\Psi}$ for the joint data space of $\mathbf{z}$, we can condition $\mathbf{y}$ on $\mathbf{x}$ and marginalize out the latent variables $\mathbf{v}$ to obtain

$$\langle y|\mathbf{x}\rangle = \underbrace{\mathbf{W}_y \left(\mathbf{I} + \mathbf{W}_\mathbf{x}^T \boldsymbol{\Psi}_\mathbf{x}^{-1} \mathbf{W}_\mathbf{x}\right)^{-1} \mathbf{W}_\mathbf{x}^T \boldsymbol{\Psi}_\mathbf{x}^{-1}}_{\mathbf{b}_{\text{JFR}}^T} \mathbf{x}, \tag{2.8}$$

where $\langle\cdot\rangle$ indicates expectation. Equation 2.8 is equivalent to

$$\mathbf{b}_{\text{JFR}} = \boldsymbol{\Psi}_\mathbf{x}^{-1} \mathbf{W}_\mathbf{x} \left(\mathbf{I} + \mathbf{W}_\mathbf{x}^T \boldsymbol{\Psi}_\mathbf{x}^{-1} \mathbf{W}_\mathbf{x}\right)^{-1} \mathbf{W}_y^T. \tag{2.9}$$

Note that the required matrix inversion of $\left(\mathbf{I} + \mathbf{W}_\mathbf{x}^T \boldsymbol{\Psi}_\mathbf{x}^{-1} \mathbf{W}_\mathbf{x}\right)$ is of the order of the latent dimensionality K, which makes joint space factor analysis for regression computationally attractive for problems in which the underlying latent variable manifold is known to be relatively low dimensional (i.e., $K \ll d$).

2.1.3 Joint Space Principal Component Regression. Tipping and Bishop (1999) show the relationship between factor analysis and principal component analysis. In particular, they show that factor analysis reduces to PCA if isotropic output noise is assumed (i.e., $\boldsymbol{\Psi} = \sigma^2 \mathbf{I}$). Taking the factor

[1] The notation Normal($\mathbf{x}$; $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$) denotes a Normal distribution over $\mathbf{x}$ with mean μ and covariance $\boldsymbol{\Sigma}$.

analysis solution for regression in equation 2.9 and assuming that $\mathbf{\Psi} = \sigma^2\mathbf{I}$, we can simplify the regression solution so that it is

$$\begin{aligned}\mathbf{b} &= \left(\sigma^2\mathbf{I}\right)^{-1}\mathbf{W}_\mathbf{x}\left(\mathbf{I} + \mathbf{W}_\mathbf{x}^T\left(\sigma^2\mathbf{I}\right)^{-1}\mathbf{W}_\mathbf{x}\right)^{-1}\mathbf{W}_y^T \\ &= \mathbf{W}_\mathbf{x}\left(\sigma^2\mathbf{I} + \mathbf{W}_\mathbf{x}^T\mathbf{W}_\mathbf{x}\right)^{-1}\mathbf{W}_y^T. \end{aligned} \tag{2.10}$$

Additionally, in PCA for regression, if we follow the assumption that $\mathbf{v} \sim \text{Normal}(0, \mathbf{I})$, then each column of $\mathbf{W}$ is an eigenvector scaled by its corresponding eigenvalue:

$$\mathbf{W} = [\lambda_1\mathbf{u}_1\ \lambda_2\mathbf{u}_2, \ldots, \lambda_K\mathbf{u}_K] = \mathbf{U}\mathbf{\Lambda}, \tag{2.11}$$

where, as defined in section 2.1.1, $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2, \ldots, \mathbf{u}_K]$ is a $d \times K$ matrix of principal eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of corresponding eigenvalues (i.e., $[\lambda_1\ \lambda_2, \ldots, \lambda_K]$ in its diagonal). Making use of the orthogonality of the eigenvectors, we can write

$$\begin{aligned}\mathbf{W}_\mathbf{x}^T\mathbf{W}_\mathbf{x} &= \mathbf{W}^T\mathbf{W} - \mathbf{W}_y^T\mathbf{W}_y \\ &= \mathbf{\Lambda}^2 - \mathbf{W}_y^T\mathbf{W}_y.\end{aligned}$$

If we denote $\bar{\mathbf{\Lambda}}^2 = \sigma^2\mathbf{I} + \mathbf{\Lambda}^2$, then we can rewrite equation 2.10 to get:

$$\begin{aligned}\mathbf{b} &= \mathbf{W}_\mathbf{x}\left(\bar{\mathbf{\Lambda}}^2 - \mathbf{W}_y^T\mathbf{W}_y\right)^{-1}\mathbf{W}_y^T \\ &= \mathbf{W}_\mathbf{x}\left[\bar{\mathbf{\Lambda}}^{-2} - \bar{\mathbf{\Lambda}}^{-2}\mathbf{W}_y^T\left(\mathbf{W}_y\bar{\mathbf{\Lambda}}^{-2}\mathbf{W}_y^T - \mathbf{I}\right)^{-1}\mathbf{W}_y\bar{\mathbf{\Lambda}}^{-2}\right]\mathbf{W}_y^T.\end{aligned} \tag{2.12}$$

Since we have a scalar output y, the matrix inversion $(\mathbf{W}_y\bar{\mathbf{\Lambda}}^{-2}\mathbf{W}_y - \mathbf{I})^{-1}$ reduces to a scalar division, making equation 2.12 now

$$\mathbf{b} = \mathbf{W}_\mathbf{x}\left[\bar{\mathbf{\Lambda}}^{-2} - \frac{\bar{\mathbf{\Lambda}}^{-2}\mathbf{W}_y^T\mathbf{W}_y\bar{\mathbf{\Lambda}}^{-2}}{\mathbf{W}_y\bar{\mathbf{\Lambda}}^{-2}\mathbf{W}_y^T - 1}\right]\mathbf{W}_y^T. \tag{2.13}$$

If we take the limit of the quantity above as σ^2 approaches 0, then $\bar{\mathbf{\Lambda}}^2 = \mathbf{\Lambda}^2$—exactly the matrix of squared eigenvalues. In this case, we can further simplify equation 2.13 as

$$\mathbf{b}_{\text{JPCR}} = \mathbf{U}_\mathbf{x}\left[\mathbf{I} - \frac{\mathbf{U}_y^T\mathbf{U}_y}{\mathbf{U}_y\mathbf{U}_y^T - 1}\right]\mathbf{U}_y^T, \tag{2.14}$$

where $\mathbf{U} = [\mathbf{U}_x^T \ \mathbf{U}_y^T]^T$ is the matrix containing the eigenvectors of the joint data $\mathbf{z}$ in its columns.

2.1.4 Kernel Dimensionality Reduction for Regression. Fukumizu, Bach, and Jordan (2006) have suggested the following method to achieve dimensionality reduction for regression. Assume that $[\mathbf{U}\ \mathbf{V}]$ is the d-dimensional orthogonal matrix, where $\mathbf{U}$ spans the subspace of $\mathbf{x}$ "relevant" to predicting y and $\mathbf{V}$ spans the orthogonal "irrelevant" subspace.

If we define $\mathbf{x}_R = \mathbf{U}^T\mathbf{x}$ and $\mathbf{x}_{\bar{R}} = \mathbf{V}^T\mathbf{x}$, then kernel dimensionality reduction seeks to find the subspace that minimizes $I(y \mid \mathbf{x}_R, \mathbf{x}_{\bar{R}} \mid \mathbf{x}_R)$, where $I(x_1, x_2)$ denotes mutual information defined by

$$I(x_1, x_2) = \int\int p(x_1, x_2) \log \frac{p(x_1, x_2)}{p(x_1)p(x_2)} dx_1 dx_2.$$

This concept is extended to the more general case of reproducing kernel Hilbert spaces on the domains of y, $\mathbf{x}_R$ and $\mathbf{x}_R$ endowed with gaussian kernels.

It should be emphasized that as with the other methods described in this section, kernel dimensionality reduction requires that the latent dimensionality K be a known quantity. In general, however, unless explicit meta-level knowledge of the data is known beforehand, the estimation of this quantity would require expensive cross-validation to avoid overfitting.

2.1.5 Stepwise Regression. Stepwise regression (Derksen & Keselman, 1992) is a popular statistical technique for large data sets that chooses dimensions to include in a regression model. The selection of dimensions for the model can be in a forward or backward manner. For example, forward stepwise regression starts with no terms in the model and at each step adds the most statistically significant dimension (using either the highest f-statistic or lowest p-value) until none are left. In contrast, backward stepwise regression starts with all dimensions in the model and removes the least significant until the remaining dimensions are statistically significant. Unfortunately, there are several issues with stepwise regression. These include its inability to cope with redundant dimensions (it deteriorates in the presence of collinearity) and its inability to shrink regression coefficients (Tibshirani, 1996), resulting in regression coefficients that are too large. These properties, among others, make it problematic for high-dimensional data sets.

2.1.6 Partial Least Squares Regression. Instead of seeking a low-dimensional version of the problem, some methods seek to structure the computation in such a way that the problem is decomposed into computationally efficient subproblems. For example, by decomposing the

multivariate regression problem into successive univariate regressions, one can create robust, iterative methods that do not suffer from the difficulties of matrix inversion for underconstrained data sets. Partial least squares regression (PLS) (Wold, 1975) is one example of such a method:

Algorithm 1: Partial Least Squares Regression

1: Initialize: $\mathbf{X}_{\text{res}} = \mathbf{X}$, $\mathbf{y}_{\text{res}} = \mathbf{y}$
2: **for** $k = 1$ to K **do** *//$K \leq d$ where d is max. input dim*
3: $\quad \mathbf{v}_k \leftarrow \mathbf{X}_{\text{res}}^T \mathbf{y}_{\text{res}}$ *//correlation direction*
4: $\quad \mathbf{s}_k \leftarrow \mathbf{X}_{\text{res}} \mathbf{v}_k$ *//project input*
5: $\quad b_k \leftarrow \mathbf{s}_k^T \mathbf{y}_{\text{res}} / \left(\mathbf{s}_k^T \mathbf{s}_k\right)$ *//univariate regression*
6: $\quad \mathbf{y}_{\text{res}} \leftarrow \mathbf{y}_{\text{res}} - b_k \mathbf{s}_k$ *//compute residual output*
7: $\quad \mathbf{X}_{\text{res}} \leftarrow \mathbf{X}_{\text{res}} - \mathbf{s}_k \mathbf{p}_k^T$ where $\mathbf{p}_k \equiv \mathbf{X}_{\text{res}}^T \mathbf{s}_k / (\mathbf{s}_k^T \mathbf{s}_k)$ *//compute residual input*
8: **end for**

In section 2.1.1, we noted that PCR projected the input data onto a very specific set of directions: the principal eigenvectors. As a direct result, the coefficients of the optimal regression vector $\mathbf{b}_{\text{PCR}}$ fall out of inexpensive univariate regressions along each projection direction. However, obtaining the eigenvectors is an (expensive for large d) $O(d^3)$ operation that can be reduced to a faster $O(d^2)$. It is here that PCR must expend the bulk of its computation.

PLS regression is a technique that is extensively used in high-dimensional and severely underconstrained domains such as in chemometrics. Rather than compute the covariance structure of the input space, as is done in PCR, PLS iteratively chooses its projection directions $\mathbf{v}_k$ (at the kth iteration) according to the direction of maximum correlation between the (current residual) input and the output. Computation of each projection direction is $O(d)$ (*linear*) in the dimensionality of the data, making PLS a highly efficient algorithm. As shown in algorithm 1, successive iterations create orthogonal projection directions by removing the subspace of the input data used in the last projection. PLS requires no expensive matrix inversion or eigendecomposition and thus is well suited to the high-dimensional yet severely underconstrained data sets in applications such as near infrared (NIR) spectrometry (Frank & Friedman, 1993).

The number of projection directions found by PLS is bound only by the dimensionality of the data, with each univariate regression on successive projection components further serving to reduce the residual error. Using all d projections is equivalent to performing ordinary least squares (OLS) regression. Hence, to avoid overfitting, the algorithm is typically stopped after K projection components are found, where K is determined empirically using cross-validation. It can be shown that if the distribution of the input data is spherical (i.e., has covariance structure $\sigma^2\mathbf{I}$), then PLS requires only a single projection to optimally reconstruct the output.

2.1.7 Backfitting. Another very general framework for estimating additive models of the form $y(\mathbf{x}) = \sum_{m=1}^{d} g_m(\mathbf{x}; \theta_m)$ is backfitting (Hastie & Tibshirani, 1990), where the functions g_m are adjustable basis functions (e.g., splines), parameterized by θ_m. As shown in algorithm 2, backfitting decomposes the statistical estimation problem into d individual estimation problems by creating "fake supervised targets" for each function g_m:

Algorithm 2: Backfitting

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

At the cost of an iterative procedure, this strategy effectively reduces the computational complexity of fan-ins in graphical models and allows easier numerical robustness control since no matrix inversion is involved.

For all its computational attractiveness, backfitting presents two serious drawbacks. First, there is no guarantee that the iterative procedure outlined in algorithm 2, will converge as it is heavily dependent on the nature of the functions g_m. Second, the updates have no probabilistic interpretation, making backfitting difficult to insert into the current framework of statistical learning, which emphasizes confidence measures, model selection, and predictive distributions. Note that Hastie & Tibshirani (2000) have proposed a Bayesian version of backfitting. Their algorithm, however, relies on Gibbs sampling, which is more suitable when dealing with the nonparametric spline models discussed there and is quite useful for generating samples from the posterior additive model. We instead focus on developing a Bayesian version of backfitting that does not require any sampling and, hence, can be implemented in incremental form for use in real-time applications such as real-time brain-machine interfaces or robotics.

In practice, a large class of methods can be traced to have similar computational underpinnings. For example, in the case of linear regression ($\mathbf{X}^T\mathbf{X}\mathbf{b} = \mathbf{X}^T\mathbf{y}$), Gauss-Seidel/Jacobi updates are a natural specialization of the general backfitting algorithm:

$$b_m = \frac{\overbrace{(\mathbf{y} - \mathbf{X}_{\bar{m}}\mathbf{b}_{\bar{m}})^T}^{\text{partial residual}} \mathbf{x}_m}{\mathbf{x}_m^T \mathbf{x}_m} \tag{2.15}$$

where $\mathbf{x}_m = [x_{1m} \cdots x_{Nm}]^T$, the vector of mth dimension entries, while $\mathbf{X}_{\bar{m}}$ denotes the data matrix with the mth dimension removed and $\mathbf{b}_{\bar{m}}$ denotes the regression coefficient vector with the mth coefficient removed. The well-known cascade correlation neural network architecture (Fahlman & Lebiere, 1989) can also be seen to have similar algorithmic underpinnings; the addition of each new hidden unit can be considered to be the tuning of an additional basis function in the sequence, with the previous basis functions being locked to their previously tuned forms.

2.1.8 Least Absolute Shrinkage and Selection Operator Regression. Least absolute shrinkage and selection operator (LASSO) regression (Tibshirani, 1996) shrinks certain regression coefficients to zero, giving interpretable models that are sparse. It minimizes the sum of squared errors, given a fixed bound on the sum of absolute value of the regression coefficients. However, LASSO regression and a wealth of other L1-regularized regression methods have an open parameter, typically a regularization parameter, that needs to be set. Some of the methods for solving L1-regularized regression problems (especially large-scale problems) include convex optimization techniques such as sequential quadratic programming or interior point methods (Kim, Koh, Lustig, Boyd, & Gorinevsky, 2007), coordinate descent methods (Friedman, Hastie, & Tibshirani, 2007), the Gauss-Seidel method (Shevade & Keerthi, 2003), generalized iterative scaling (Goodman, 2004), and iterative reweighted least squares (Lokhorst, 1999; Lee, Lee, Abbeel, & Ng, 2006).

The LASSO estimate $\hat{\mathbf{b}}_{lasso}$ is then defined as

$$\hat{\mathbf{b}}_{lasso} = \arg\min \left\{ \sum_{i=1}^{N} \left(y_i - \sum_j b_j x_{ij} \right)^2 \right\} \text{ subject to } \sum_j \mid b_j \mid \leq t,$$

where $t \geq 0$ is a tuning parameter that can be set using n-fold cross-validation or manual hand-tuning. For smaller values of t, LASSO regression gives solutions that are sparse estimates of the least squares estimates. For larger values of t, the above constraint has little effect, resulting in a solution similar to ridge regression. The main difference between LASSO regression and ridge regression is that LASSO attempts to shrink the solution by using L1 penalty norm (i.e., $\sum \mathbf{b}$), while ridge regression uses L2 penalty norm (i.e., $\sum \mathbf{b}^2$). Ng (2004) shows that this contributes to LASSO being an effective algorithm suitable for high-dimensional data sets, at the expense of an open parameter that needs to be set using cross-validation or through the optimization of a regularization "path" of solutions (Efron, Hastie, Johnstone, & Tibshirani, 2004).[2]

[2]That is, solutions that minimize the L_1 loss function. When the value of the open or tuning parameter changes, regularization paths of solutions are generated.

2.2 Data Structures for Fast Statistics. Significant computational gains can be achieved by using smarter data structures to organize the information required for statistical analysis. Examples of these include KD-trees and ball-trees (Friedman, Bentley, & Finkel, 1977; Gray & Moore, 2001; Omohundro, 1990), which allow caching of sufficient statistics over recursively smaller regions of the data space, and AD-trees (Moore & Lee, 1998; Komarek and Moore, 2000), which speed up computations involving conjunctive queries and "counting" statistics.

KD-trees (Friedman et al., 1977) are data structures that partition the input space into hyperrectangular regions. The root node contains the bounding box of the entire data set, and each nonleaf node has two children that partition the parents' space by splitting the bounding box along its longest dimension (see Figure 2). Splitting stops when the bounding boxes reach a certain minimum size or when the number of points in a box reaches a minimum value. The key computational saving results from annotating each node of the tree with specific statistics about the data in the partition of space rooted at that node. For example, caching the bounding box of the data in each node allows eliminating a significant number of explicit comparisons when answering nearest-neighbor queries. In this way, for each query, only a fraction of the leaves in the tree are visited, resulting in sublinear computational complexity for most operations that typically require at least linear time.

A similar computational saving is achievable for kernel density estimation if we are willing to sacrifice a small amount of accuracy. Given the bounding boxes of the nodes in the KD-tree, we can bound the minimum and maximum value of the kernel function (assuming a monotonically decreasing function) within a hyperrectangle. If the difference between the minimum and maximum is less than a tolerance value ϵ, we can skip the evaluation of each query point within the node and approximate it by an average value. This achieves significant savings when the query points are the data points themselves, as is frequently the case in settings where we evaluate the data on kernels that are centered at the data points—so-called *N*-body problems (Gray & Moore, 2001).

KD-trees suffer in higher-dimensional spaces since as the dimensionality increases, one observes that most of the volume is concentrated in a thin shell at the outer edges of the space. Metric trees and ball trees (Omohundro, 1990) are alternatives that are robust to high-dimensional problems. They do not necessarily require a Euclidean space, but merely one in which the triangle inequality holds (Moore, 2000). Because of this, we can derive simple yet computationally efficient bounds on the distances between a query point $\mathbf{q}$ and any point $\mathbf{x}$ belonging to a ball of radius r (as Figure 3 illustrates):

$$\begin{aligned} \|\mathbf{q} - \mathbf{x}\| &\leq |\, \mathbf{q} - \mathbf{c}\| + r \\ \|\mathbf{q} - \mathbf{x}\| &\geq |\, \mathbf{q} - \mathbf{c}\| - r \end{aligned}.$$

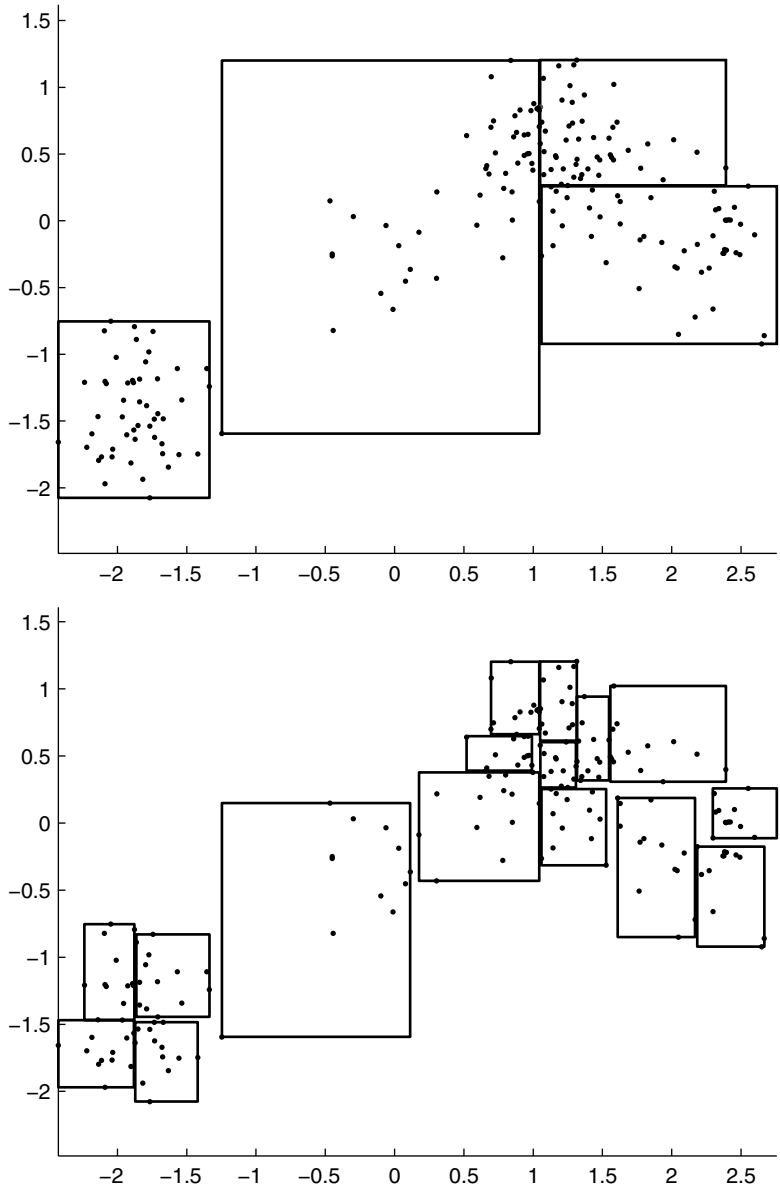
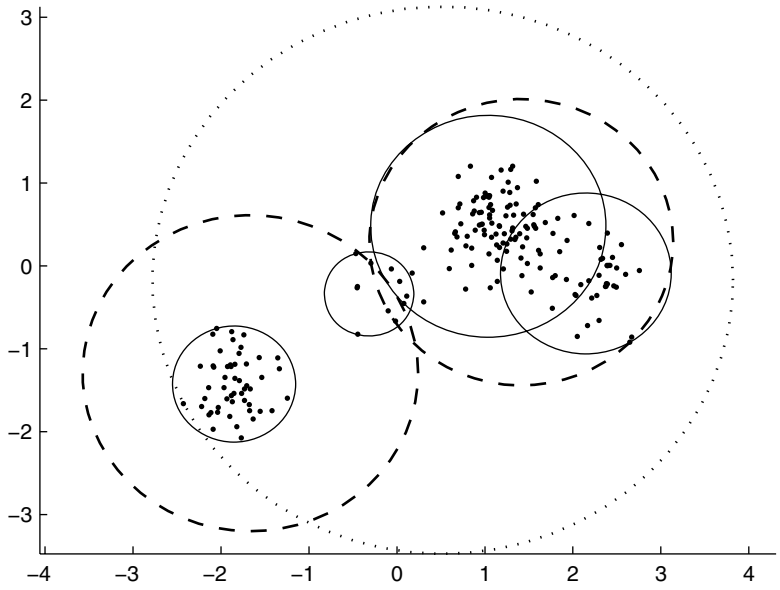
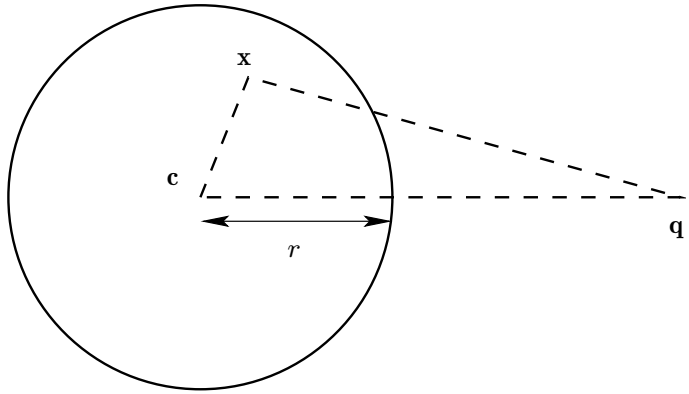


Figure 2: This figure shows the bounding boxes of the data stored at level 2 (top) and level 4 (bottom) nodes of a KD-tree. The tree is created by recursively splitting the hyperrectangles along the median of the longest dimension of the enclosed data. Bounding box information (as well as other statistics) is cached at each node and helps speed up querying the structure.



(a) Ball-tree



(b) Triangle inequality

Figure 3: (Top) The nodes at the root, first, and second levels of a ball tree (dotted, dashed, and solid balls, respectively). (Bottom) The triangle inequality used to derive computationally efficient bounds on the distance between an arbitrary query point and the points within a ball.

These distance bounds are then used in a manner similar to the bounding boxes of KD-trees to reduce the number of comparisons required to be performed with the actual data points.

AD-trees (Moore & Lee, 1998) are an efficient representation for statistical methods that rely on "counting" occurrences of records satisfying sets of conjunctive queries over the record attributes. Traditional representation schemes for such data include precomputing answers to each query, which are stored in so-called contingency tables. Contingency tables are useful in creating probability tables for Bayes nets and in conjunctive rule learning algorithms such as decision tree learning. Potential uses for statistical machine translation are obvious when we use the popular TF-IDF (term-frequency, inverse-document-frequency) representation of documents.

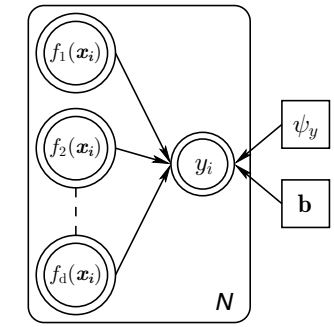
AD-trees allow the precomputed answers to queries that are available in contingency tables to be stored in a fraction of the memory requirements. For data sets in which records arrive incrementally or in which the initial cost of constructing the AD-tree is too high, an incremental version is also possible (Komarek & Moore, 2000).

2.3 Summary. For the purpose of comparison, we included the following algorithms for evaluation in our experiments: stepwise regression, PLS regression, and LASSO regression. We omitted the other methods reviewed in this section due to computational drawbacks or unsuitable model assumptions. For example, methods like joint-space factor analysis for regression, principal component regression, joint-space principal component regression, and kernel dimensionality reduction for regression require that the latent dimensionality K be known. Estimation of this quantity for high-dimensional data sets could be potentially very expensive due to the cross-validation procedures needed. Additionally, principal component regression may also discard important low-variance inputs that contribute to the output in favor of high-variance but irrelevant input dimensions since it seeks components to maximize the variance in the input data. Backfitting may be computationally more robust for high-dimensional inputs, but it is unable to detect irrelevant and redundant input dimensions should these exist in the input data. (See Schaal et al., 1998, for a more comprehensive review of these methods.)

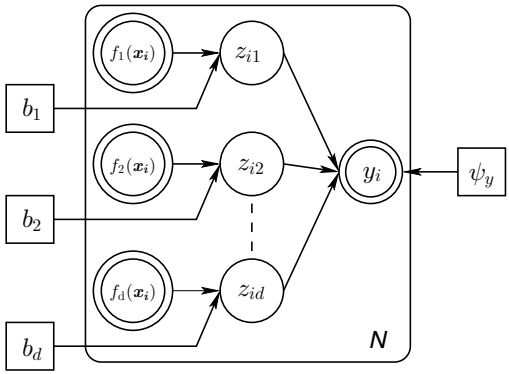
3 Probabilistic Backfitting

The model shown in Figure 4a generalizes our discussion in section 2 such that the input "dimensions" of Figure 1 are replaced by arbitrary basis functions $f_m(\mathbf{x})$ of the input—a model commonly known as generalized linear regression (GLR) (Hastie & Tibshirani, 1990). Our goal remains the same: given a data set $\mathbf{x}_D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, we wish to determine the most likely values of b_m, which linearly combine the basis functions f_m to generate the output y.

We also noted in section 2.1.7 that the backfitting family of algorithms is an efficient set of methods that, under the right circumstances, is extremely robust since it requires no expensive matrix inversion and thus avoids



(a) Generalized Linear Regression



(b) Probabilistic Backfitting

Figure 4: We modify the original graphical model for generalized linear regression by inserting hidden variables z_{im} in each branch of the fan-in. This modified model can be solved using the EM framework to derive a probabilistic version of backfitting.

the numerical pitfalls therein. A drawback of the backfitting procedure is that it does not stem from a generative probabilistic model, which limits its application in current Bayesian machine learning frameworks. In this section, we describe how a probabilistic version of backfitting can be derived by making a simple structural modification to the graphical model for standard generalized linear regression. The statistical model corresponding to Figure 4a can be written as

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

that is, multiple predictors $f_m(\mathbf{x}; \boldsymbol{\theta}_m)$ (where $1 \leq m \leq d$) that are generated by an adjustable nonlinear transformation with parameters $\boldsymbol{\theta}_m$ and fed linearly to an output y by an inner product with a regression vector

$\mathbf{b} = [b_1 b_2 \cdots b_d]^T$ plus additive noise ϵ. As we mentioned in section 2, evaluation of $\mathbf{b}$ using the OLS solution in equation 2.2 becomes increasingly computationally expensive and numerically brittle. Note that ridge regression can "fix" such problems numerically by stabilizing the matrix inversion with a small, additive diagonal term. However, a ridge factor typically introduces uncontrolled bias.

A simple modification of the graphical model of Figure 4a, however, enables us to create the desired algorithmic decoupling of the predictor functions and gives backfitting a probabilistic interpretation. Consider the introduction of random variables z_{im} as shown in Figure 4b. These variables are analogous to the output of the g_m function of algorithm 2 and can also be interpreted as an unknown fake target for each branch of the regression fan-in. For the derivation of our algorithm, we assume the following conditional distributions for each variable in the model:

$$\begin{aligned} y_i \mid \mathbf{z}_i &\sim \text{Normal}(y_i; \mathbf{1}^T \mathbf{z}_i, \psi_y) \\ z_{im} \mid \mathbf{x}_i &\sim \text{Normal}(z_{im}; b_m f_m(\mathbf{x}_i), \psi_{zm}), \end{aligned} \tag{3.1}$$

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. With the introduction of the random variables z_{im}, we are essentially in a situation where we wish to optimize the parameters $\boldsymbol{\phi} = \{\{b_m, \psi_{zm}\}_{m=1}^d, \psi_y\}$, given that we have observed variables $\{\mathbf{x}_i, y_i\}_{i=1}^N$ and have unobserved variables $\{\mathbf{z}_i\}_{i=1}^N$ in our graphical model. This situation fits naturally into the framework of maximum likelihood estimation via the EM algorithm.

3.1 An EM Algorithm for Probabilistic Backfitting. Given our modified statistical model represented by the graphical model of Figure 4b, we wish to estimate the parameters b_m and (possibly) optimize the individual functions $f_m(\mathbf{x}; \boldsymbol{\theta}_m)$ with respect to the parameters $\boldsymbol{\theta}_m$. This is easily formulated as an EM algorithm, which maximizes the incomplete log likelihood $\log p(\mathbf{y} \mid \mathbf{X})$, which, from Figure 4a, can be expressed as

$$\log p(\mathbf{y} \mid \mathbf{X}) = -\frac{N}{2} \log \psi_y - \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \mathbf{b}^T \mathbf{f}(\mathbf{x}_i)\right)^2 + const. \tag{3.2}$$

The EM algorithm however, operates by maximizing the expected complete log likelihood $\langle \log p(\mathbf{y}, \mathbf{Z} \mid \mathbf{X}; \boldsymbol{\phi}) \rangle$, where

$$\begin{aligned} \log p(\mathbf{y}, \mathbf{Z} \mid \mathbf{X}; \boldsymbol{\phi}) = &-\frac{N}{2} \log \psi_y - \frac{1}{2\psi_y} \sum_{i=1}^{N} \left(y_i - \mathbf{1}^T \mathbf{z}_i\right)^2 \\ &- \sum_{m=1}^{d} \left[\frac{N}{2} \log \psi_{zm} + \frac{1}{2\psi_{zm}} \sum_{i=1}^{N} \left(z_{im} - b_m f_m(\mathbf{x}_i; \boldsymbol{\theta}_m)\right)^2 \right] + const. \end{aligned} \tag{3.3}$$

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

E-Step

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

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

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

M-Step

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

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

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

where we define $s = \psi_y + \sum_{m=1}^{d} \psi_{zm}$ and $\boldsymbol{\Sigma}_{\mathbf{z}} = \mathrm{Cov}(\mathbf{z} \mid \mathbf{y}, \mathbf{X})$. In addition, the parameters θ_m of each function f_m can be updated by setting:

$$\sum_{i=1}^{N}\left(\langle z_{im} \rangle - b_m f_m(\mathbf{x}_i; \theta_m)\right)\frac{\partial f_m(\mathbf{x}_i; \theta_m)}{\partial \theta_m} = 0 \tag{3.4}$$

and solving for θ_m. As this step depends on the particular choice of f_m (e.g., splines, kernel smoothers, parametric models), we will not pursue it any further and note that any statistical approximation mechanism could be used.

There are two observations to be made regarding the above EM algorithm. 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 for the $(n+1)$th EM cycle:

$$b_m^{(n+1)} = b_m^{(n)} + \frac{\psi_{zm}}{s}\frac{\sum_{i=1}^{N}\left(y_i - \sum_{k=1}^{d} b_k^{(n)} f_k(\mathbf{x}_i)\right) f_m(\mathbf{x}_i)}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2}. \tag{3.5}$$

Thus, in an EM cycle, the mth regression coefficient is updated by an amount proportional to the correlation between the mth predictor and the residual error. Each b_m is updated independently (that is, independent of the other regression coefficients b_p, where $p \neq m$ for $p = 1, \ldots, d$). In each EM cycle, all d regression coefficients are updated.

In this way, the residual can be interpreted as forming a "fake target" for the mth branch of the fan-in. As the next section shows, this enables us to place the algorithm in the context of backfitting.

The matrix inversion in the OLS solution requires $O(d^2)$ if more efficient and robust matrix inversion methods are used. In comparison, the computational complexity of the EM-based probabilistic backfitting algorithm is $O(d)$ per EM iteration. Should the number of EM iterations be significant, it is true that the run time of the EM algorithm could be as long as non iterative approaches. However, the true benefit of our iterative approach arises when dealing with real-time applications, where decisions need to be made quickly in a short amount of time such that an approximate solution is acceptable. Additionally, EM-based probabilistic backfitting can be embedded into other iterative methods in order to realize more computationally efficient update equations.

3.2 Relating Traditional and Probabilistic Backfitting. To better understand how equation 3.5 can be interpreted as probabilistic backfitting, notice that backfitting can be viewed as a formal Gauss-Seidel algorithm: an equivalence 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

$$b_m = \frac{\sum_{i=1}^{N}\left(y_i - \sum_{k\neq m}^{d} b_k f_k(\mathbf{x}_i)\right) f_m(\mathbf{x}_i)}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2}. \tag{3.6}$$

Note that if used naively, equation 3.6 does not guarantee convergence at all. The Gauss-Seidel algorithm extends the above equation by adding a fraction $(1 - \omega)$ of b_m to the update, giving us the well-known relaxation algorithms,

$$b_m^{(n+1)} = (1-\omega)b_m^{(n)} + \omega\frac{\sum_{i=1}^{N}\left(y_i - \sum_{k\neq m}^{d} b_k f_k(\mathbf{x}_i)\right) f_m(\mathbf{x}_i)}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2}, \tag{3.7}$$

which has improved convergence rates for overrelaxation ($1 < \omega < 2$) or improved stability for underrelaxation ($0 < \omega < 1$). For $\omega = 1$, the standard backfitting of equation 3.6 is recovered. The appropriate value of ω, which allows the iterations to converge while still maintaining a reasonable convergence rate, can be determined only by treating equation 3.6 as a discrete

dynamical system and analyzing the eigenvalues of its system matrix—a task taking $O(d^3)$ (or $O(d^2)$, for more efficient methods). If, however, we set $\omega = \omega_m = \psi_{zm}/s$ in equation 3.7, it can be shown (after some algebraic rearrangement) that we obtain our EM update in equation 3.5 exactly. Indeed, this is a probabilistic version of backfitting.

A similar EM algorithm and model structure has been proposed in the context of signal processing (Feder & Weinstein, 1988), but we believe this is the first time that the connection of this probabilistic derivation to the backfitting algorithm has been demonstrated. As we show in section 4, this allows us to place this class of methods within a much wider framework of Bayesian model complexity estimation.

3.3 Convergence of Probabilistic Backfitting. In general, for any maximum likelihood problem, the EM algorithm guarantees monotonic increase in the incomplete likelihood, but does not guarantee that the final solution is the global maximum. This section addresses the following questions:

1. What is the point of convergence of the probabilistic backfitting EM algorithm?
2. Are there local maxima (globally suboptimal solutions) in its likelihood space?

The answers to both questions depend on the fact that the incomplete likelihood (or marginalized complete likelihood) function for linear regression in equation 3.2 has a (possibly nonunique but convex) global maximum corresponding to the OLS solution of equation 2.2 but no local maxima. Could the introduction of the hidden variables and additional parameters in equation 3.3 introduce local maxima in the likelihood landscape? Note that for examining convergence properties, we focus on only the estimation of the parameters $\boldsymbol{\phi} = [\mathbf{b}, \psi_{z1}, \ldots, \psi_{zd}, \psi_y]^T$, as the functions f_m cannot be treated in general without knowing their structure. We start with the assumption that we have reached a stationary point $\boldsymbol{\phi}^*$ in the EM algorithm, which implies

$$\left.\frac{\partial \langle \ln p(\mathbf{y}, \mathbf{Z} \mid \mathbf{X}; \boldsymbol{\phi}) \rangle}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = \mathbf{0}. \tag{3.8}$$

Using Jensen's inequality, it is easy to show that for an arbitrary distribution $Q(\mathbf{Z})$ over the hidden variables,

$$\ln p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\phi}) \geq \langle \ln p(\mathbf{y}, \mathbf{Z} \mid \mathbf{X}; \boldsymbol{\phi}) \rangle_{Q(\mathbf{Z})} + \mathcal{H}[Q(\mathbf{Z})] = \mathcal{F}(Q, \boldsymbol{\phi}), \tag{3.9}$$

where $\mathcal{H}[\cdot]$ denotes entropy. EM performs a coordinate ascent, alternately maximizing $\mathcal{F}$ with respect to Q (in the E-step) and $\boldsymbol{\phi}$ (in the M-step). Differentiating $\mathcal{F}(Q, \boldsymbol{\phi})$ with regard to $\boldsymbol{\phi}$ at the stationary point $\boldsymbol{\phi}^*$, and

noting that the entropy term $\mathcal{H}[Q(\mathbf{Z})]$ is independent of $\boldsymbol{\phi}$, gives

$$\left.\frac{\partial \mathcal{F}(Q, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = \left.\frac{\partial \langle \ln p(\mathbf{y}, \mathbf{Z} \mid \mathbf{X}; \boldsymbol{\phi}) \rangle}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = \mathbf{0}. \tag{3.10}$$

Note, however, that the preceding E-step sets $Q(\mathbf{Z})$ to the true posterior distribution $p(\mathbf{Z} \mid \mathbf{y}, \mathbf{X}; \boldsymbol{\phi}^*)$, which raises the lower bound in equation 3.9 to an equality—$\ln p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\phi}) = \mathcal{F}(Q, \boldsymbol{\phi})$—from which it follows that

$$\left.\frac{\partial \ln p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = \left.\frac{\partial \mathcal{F}(Q, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = 0, \tag{3.11}$$

that is, we have reached a maximum in the incomplete likelihood as well. Given that the incomplete log likelihood $\ln p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\phi})$ in equation 3.2 has only a global maximum (i.e., the OLS solution), reaching the stationary point of equation 3.8 in our EM algorithm for probabilistic backfitting must correspond to finding the OLS solution. Therefore, probabilistic backfitting is indeed performing true linear regression with a global optimum.

4 Variational Bayesian Least Squares

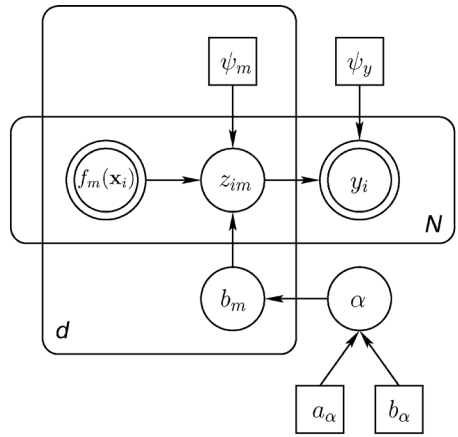
The probabilistic version of backfitting suffers from overfitting when the input data contain many redundant or irrelevant features. To regularize its OLS solution, we adopt a Bayesian framework and place a prior distribution over the regression coefficients $\mathbf{b}$ to get VBLS. As the following two sections demonstrate, our choice of prior structure results in two different, yet important, forms of regularization. We also discuss how we can easily obtain confidence intervals and demonstrate how VBLS can be additionally applied to classification problems.

4.1 Regularizing the Regression Vector Length. The graphical model for our first form of Bayesian prior is shown in Figure 5a. We place a gaussian prior over the regression coefficient vector $\mathbf{b}$ so that the variance of the prior is controlled by a single precision parameter α. As a result, our uncertainty in the value of this prior precision is represented by a broad gamma distribution over α:

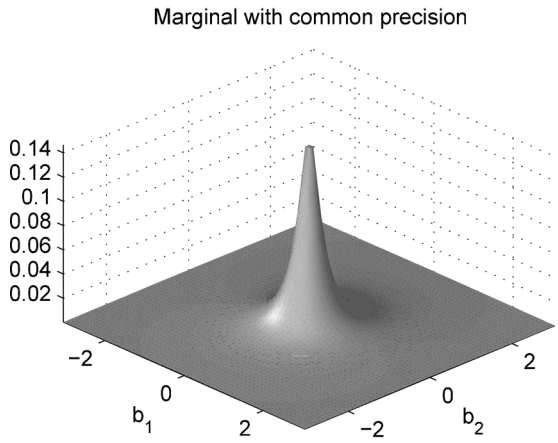
$$\begin{aligned} \mathbf{b} \mid \alpha &\sim \text{Normal}\,(\mathbf{b}; 0, \mathbf{I}/\alpha)\,, \\ \alpha &\sim \text{Gamma}\,(\alpha; a_{\alpha,0}, b_{\alpha,0})\,, \end{aligned} \tag{4.1}$$

where $a_{\alpha,0}$ and $b_{\alpha,0}$ are the initial hyperparameter values for the gamma distribution over α.

Two motivations lie behind our choice of gamma prior. First, as a scale parameter, an uninformative distribution over α must be uniform over a



(a) VBLS with shrinkage prior



(b) Marginal prior over b

Figure 5: By associating a single gamma distributed precision with the regression vector, we create a marginal prior over **b** that favors minimum-norm solutions, similar to shrinkage methods such as ridge regression.

log scale—corresponding to a Jeffreys prior (Jeffreys, 1946; Gelman, Carlin, Stern, & Rubin, 2000). We fulfill the requirement of an uninformative distribution by choosing the gamma distribution parameters, a_α and b_α, appropriately (i.e., $a_\alpha, b_\alpha \to 0$). Second, the gamma distribution is analytically convenient, since it is a conjugate distribution for the gaussian precision. As the model in Figure 5a shows, our set of unobserved random variables in the model is now $\{\mathbf{b}, \alpha, \{\mathbf{z}_i\}_{i=1}^N\}$, and we are especially interested in obtaining posterior distributions over the random variables **b** and α. The

parameters we wish to optimize are now $\boldsymbol{\phi} = \{\mathbf{b}, \{\psi_{zm}\}_{m=1}^{d}, \psi_y, a_\alpha, b_\alpha\}$. The joint probability over this model extends equation 3.3 to

$$\begin{aligned}
\log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \alpha \mid \mathbf{X}; \boldsymbol{\phi}) = & \log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \alpha \mid \mathbf{X}; \boldsymbol{\Psi}_z, \psi_y, a_\alpha, b_\alpha) \\
= & -\frac{N}{2}\log\psi_y - \frac{1}{2\psi_y}\sum_{i=1}^{N}\left(y_i - \mathbf{1}^T\mathbf{z}_i\right)^2 \\
& -\sum_{m=1}^{d}\left[\frac{N}{2}\log\psi_{zm} + \frac{1}{2\psi_{zm}}\sum_{i=1}^{N}(z_{im} - b_m f_m(\mathbf{x}_i;\boldsymbol{\theta}_m))^2\right] \\
& +\frac{d}{2}\log\alpha - \frac{\alpha}{2}\sum_{m=1}^{d} b_m^2 + (a_{\alpha,0}-1)\log\alpha - b_{\alpha,0}\alpha \; + const.
\end{aligned} \tag{4.2}$$

While the log joint posterior $\log Q(\mathbf{Z}, \mathbf{b}, \alpha)$ is readily available from equation 4.2 (up to a constant additive term), the extraction of marginal probabilities of interest such as $Q(\mathbf{b})$ and $Q(\alpha)$ is analytically intractable. Therefore, we use a factorial variational approximation (Ghahramani & Beal, 2000b; Parisi, 1988; Rustagi, 1976) to the true posterior, in which we assume that the posterior distribution factorizes[3] over the variables of interest, that is, we restrict ourselves to a family of distributions of the form $Q(\mathbf{Z}, \mathbf{b}, \alpha) = Q(\mathbf{Z})Q(\mathbf{b})Q(\alpha)$. This procedure allows us to analytically derive an EM-like set of update equations for the individual posterior distributions:

$$\begin{aligned}
Q(\alpha) &= \text{Gamma}(\alpha; \hat{a}_\alpha, \hat{b}_\alpha) \\
Q(\mathbf{b}) &= \prod_{m=1}^{d} \text{Normal}\left(b_m; \mu_{b_m}, \sigma_{b_m}^2\right) \\
\hat{a}_\alpha &= a_{\alpha,0} + \frac{d}{2} \\
\hat{b}_\alpha &= b_{\alpha,0} + \frac{\langle \mathbf{b}^T\mathbf{b}\rangle}{2} \\
\sigma_{b_m}^2 &= \left(\frac{1}{\psi_{zm}}\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \langle\alpha\rangle\right) \\
\mu_{b_m} &= \sigma_{b_m}^2\left(\frac{1}{\psi_{zm}}\sum_{i=1}^{N} f_m(\mathbf{x}_i)\langle z_{im}\rangle\right),
\end{aligned} \tag{4.3}$$

[3]This particular factorization causes the marginal posterior of $\mathbf{b}$ to be a gaussian. An alternative (also analytically tractable) formulation $Q(\mathbf{Z}, \mathbf{b}, \alpha) = Q(\mathbf{Z})Q(\mathbf{b}, \alpha)$ is also possible, in which the resulting marginal for $\mathbf{b}$ is a Student-t distribution.

where

$$\left\langle \mathbf{b}^T \mathbf{b} \right\rangle = \langle \mathbf{b} \rangle^T \langle \mathbf{b} \rangle + \mathbf{1}^T \boldsymbol{\Sigma}_{\mathbf{b}} \mathbf{1},$$

$\langle \mathbf{b} \rangle = [\mu_{b_1}\ \mu_{b_2}\ \cdots\ \mu_{b_d}]^T$, and $\boldsymbol{\Sigma}_{\mathbf{b}}$ is the posterior covariance of $\mathbf{b}$ (i.e., a diagonal matrix with μ_{b_m} entries on its diagonal).

The form of the $Q(\mathbf{Z})$ distribution updates remains identical to that derived in section 3.1, with the exception that the parameters b_m are replaced with the expectations $\langle b_m \rangle$, so we shall not repeat them here. However, substituting the expressions for $\langle z_{im} \rangle$ in the update equations for the distribution of $Q(\mathbf{b})$ gives the following update for the regression coefficients,

$$\begin{aligned} \langle b_m \rangle^{(n+1)} = & \left(\frac{\sum_{i=1}^N f_m(\mathbf{x}_i)^2}{\sum_{i=1}^N f_m(\mathbf{x}_i)^2 + \psi_{zm} \langle \alpha \rangle} \right) \langle b_m \rangle^{(n)} \\ & + \frac{\psi_{zm}}{s} \frac{\sum_{i=1}^N \left(y_i - \sum_{k=1}^d \langle b_k \rangle^{(n)} f_k(\mathbf{x}_i) \right) f_m(\mathbf{x}_i)}{\left(\sum_{i=1}^N f_m(\mathbf{x}_i)^2 + \psi_{zm} \langle \alpha \rangle \right)}, \end{aligned} \tag{4.4}$$

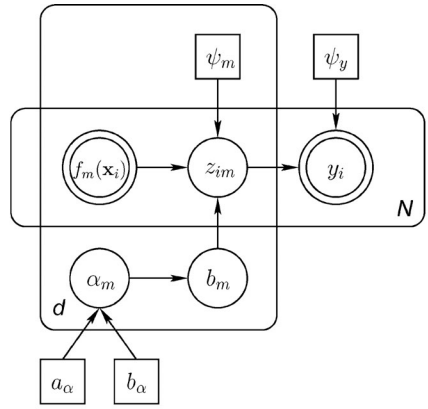
where, as in section 3.1, $s = \psi_y + \sum_{m=1}^d \psi_{zm}$.

Comparing the solution in equation 4.4 with the result derived for probabilistic backfitting in equation 3.5, we see that in the absence of correlation between the residual error and the kth predictor $f_k(\mathbf{x})$—that is, if the second term of equation 3.5 is zero, the first term of equation 4.4 is a decaying term. As a result, the corresponding regression coefficient $\langle b_m \rangle$ will go to zero after some number of EM iterations. This effect is similar to that of shrinkage methods such as ridge regression.

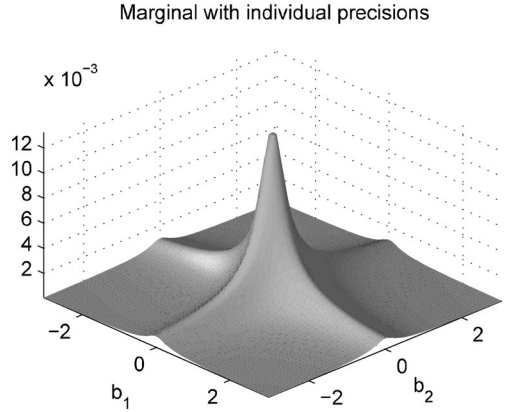
Note that the structure of the marginal prior over the regression coefficients $\mathbf{b}$ in Figure 5b—that is, the marginal prior of $\mathbf{b}$ from equation 4.1—suggests that solutions closer to the origin are favored. In fact, sharing the common precision variable α across all the regression coefficients results in a regularized solution that minimizes the norm $\|\mathbf{b}\|^2$ of the entire regression vector. This is, in fact, identical to a ridge regression solution with a single ridge parameter. However, in our formulation, the estimation of the "correct" value of the ridge parameter is implicitly inferred without the need for traditionally expensive cross-validation techniques.

Regularizing the regression vector length is particularly useful when there are groups of inputs supplying redundant information (e.g., for robustness across sensors), since the regression solution tends to distribute the responsibility for the output inference over all relevant input dimensions.

4.2 Regularizing the Number of Relevant Inputs. Modifying Figure 5a slightly, we now place individual precision variables α_m over each of the regression parameters b_m, resulting in Figure 6a. This model structure can



(a) VBLS with ARD prior



(b) Marginal prior over b

Figure 6: By associating an individual gamma distributed precision with each regression coefficient, we create a marginal prior over **b** that favors sparse solutions that lie along the (hyper)-spines of the distribution.

be captured by the following set of prior distributions:

$$\begin{aligned} \mathbf{b} \mid \boldsymbol{\alpha} &\sim \prod_{m=1}^{d} \text{Normal}\,(b_m; 0, 1/\alpha_m) \\ \boldsymbol{\alpha} &\sim \prod_{m=1}^{d} \text{Gamma}\big(\alpha_m; a_{\alpha_m,0}, b_{\alpha_m,0}\big) \end{aligned} \quad , \tag{4.5}$$

where $a_{\alpha_m,0}$ and $b_{\alpha_m,0}$ are the initial hyperparameter values for α_m.

As the graphical model in Figure 6a shows, our set of unobserved variables in the model is now $\{\mathbf{b}, \boldsymbol{\alpha}, \{\mathbf{z}_i\}_{i=1}^N\}$. The modified likelihood function can be rewritten as follows:

$$\begin{aligned}
\log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \alpha \mid \mathbf{X}; \boldsymbol{\phi}) \;&=\; \log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \alpha \mid \mathbf{X}; \boldsymbol{\Psi}_{\mathbf{z}}, \psi_y, a_{\boldsymbol{\alpha}}, b_{\boldsymbol{\alpha}}) \\
&= -\frac{N}{2} \log \psi_y - \frac{1}{2\psi_y} \sum_{i=1}^{N} \left(y_i - \mathbf{1}^T \mathbf{z}_i\right)^2 \\
&\quad - \sum_{m=1}^{d} \left[\frac{N}{2} \log \psi_{zm} + \frac{1}{2\psi_{zm}} \sum_{i=1}^{N} (z_{im} - b_m f_m(\mathbf{x}_i; \boldsymbol{\theta}_m))^2 \right] \\
&\quad + \sum_{m=1}^{d} \left[\frac{d}{2} \log \alpha_m - \frac{\alpha_m}{2} b_m^2 \right] \\
&\quad + \sum_{m=1}^{d} \left\{ (a_{\alpha_m,0} - 1) \log \alpha_m - b_{\alpha_m,0}\, \alpha_m \right\} + const. \qquad (4.6)
\end{aligned}$$

Proceeding as in section 4.1, we can derive the following iterative updates to the distributions of $Q(\mathbf{b})$ and $Q(\boldsymbol{\alpha})$:

$$\begin{aligned}
Q(\boldsymbol{\alpha}) &= \prod_{m=1}^{d} \text{Gamma}\left(\alpha_m; \hat{a}_{\alpha_m}, \hat{b}_{\alpha_m}\right) \\
Q(\mathbf{b}) &= \prod_{m=1}^{d} \text{Normal}\left(b_m; \mu_{b_M}, \sigma_{b_m}^2\right) \\
\hat{a}_{\alpha_m} &= a_{\alpha_m,0} + \frac{1}{2} \\
\hat{b}_{\alpha_m} &= b_{\alpha_m,0} + \frac{\langle b_m^2 \rangle}{2} \\
\sigma_{b_m}^2 &= \left(\frac{1}{\psi_{zm}} \sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \langle \alpha_m \rangle \right)^{-1} \\
\mu_{b_m} &= \sigma_{b_m}^2 \left(\frac{1}{\psi_{zm}} \sum_{i=1}^{N} f_m(\mathbf{x}_i) \langle z_{im} \rangle \right) \qquad (4.7)
\end{aligned}$$

where

$$\langle b_m^2 \rangle = \langle b_m \rangle^2 + \sigma_{b_m}^2 = \mu_{b_m}^2 + \sigma_{b_m}^2 .$$

Deriving the update equations for the mean of the regression coefficients as we did in equation 4.4, we get

$$\langle b_m \rangle^{(n+1)} = \left(\frac{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \langle \alpha_m \rangle} \right) \langle b_m \rangle^{(n)} + \frac{\psi_{zm}}{s} \frac{\sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{d} \langle b_k \rangle^{(n)} f_k(\mathbf{x}_i) \right) f_m(\mathbf{x}_i)}{\left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \langle \alpha_m \rangle \right)}, \tag{4.8}$$

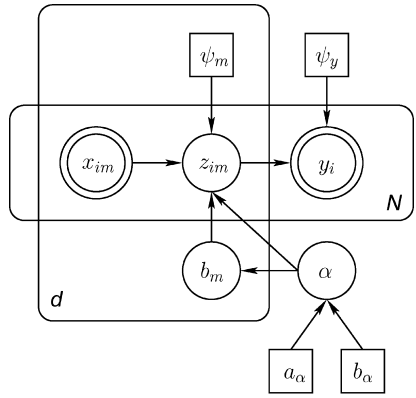
where, as in section 3.1, $s = \psi_y + \sum_{m=1}^{d} \psi_{zm}$.

The solution in equation 4.8 is almost identical to that of equation 4.4, except now the regularization of the regression solution occurs over the magnitude of each regression coefficient rather than the overall norm. This is a direct effect of having individual precision variables rather than a common precision variable. The result is a regression solution that minimizes the number of relevant inputs required to accurately predict the output, much like the automatic relevance detection (ARD) framework in neural networks (Neal, 1994). This is also intuitively apparent from the marginal prior over **b** shown in Figure 6b—that is, the marginal prior of **b** from equation 4.5, which favors sparse solutions that lie along the (hyper-)spines of the distribution.

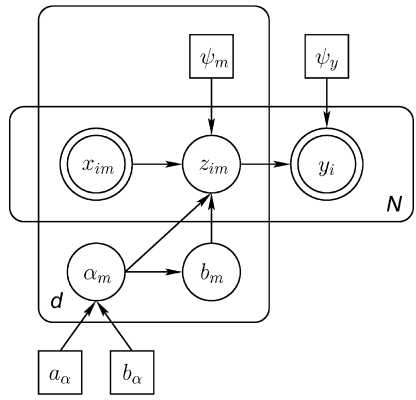
While the regularization discussed in section 4.1 is useful in situations where redundant information is to be regularized but not eliminated in the regression, this current form of regularization (i.e., regularizing the number of inputs) is desirable when the input contains information that is irrelevant to predicting the output.

Note that the graphical models of Figures 5a and 6a are two extremes in a spectrum of regularization options. One can certainly conceive of models in which groups of regression coefficients are placed under the control of individual precision parameters. This situation may make sense, for example, when we have groups of redundant sensors providing input. It allows an irrelevant signal (set of sensors) to be eliminated if it does not contribute to the output. At the same time, it allows a relevant set to exploit the redundancy of information within its group to provide a more robust input signal.

4.3 Alternative Posterior Factorization. In sections 4.1 and 4.2, we made the assumption that the posterior distribution factorized over the regression coefficients b_m and their precisions α (or α_m, in section 4.2). We can relax this assumption if we make a small modification to the graphical model to retain analytical tractability. Figure 7a shows an alternative to Figure 5a, which can be described by the following conditional



(a) Common prior



(b) Individual priors

Figure 7: We can relax the assumption of factorization $Q(\mathbf{b})Q(\boldsymbol{\alpha})$ between the regression coefficients and their precision variables by modifying the graphical models as shown in this figure. The marginal posterior distribution over the regression coefficients $\mathbf{b}$ can now be analytically derived as a Student t-distribution.

distributions:

$$\begin{aligned} y_i \mid \mathbf{z}_i &\sim \text{Normal}(y_i; \mathbf{1}^T \mathbf{z}_i, \psi_y) \\ z_{im} \mid b_m, \alpha, x_{im} &\sim \text{Normal}\,(z_{im}; b_m x_{im}, \psi_{zm}/\alpha) \\ \mathbf{b} \mid \alpha &\sim \text{Normal}\,(\mathbf{b}; 0, \mathbf{I}/\alpha) \\ \alpha &\sim \text{Gamma}\,(\alpha; a_{\alpha,0}, b_{\alpha,0})\,. \end{aligned} \tag{4.9}$$

The dependency of z_{im} on the precision α may seem unnecessary, but Gelman et al. (2000) provide a justification: it is reasonable to assume that the variance in z_{im} scales with the variance in b_m since increasing our uncertainty in the prior of b_m should imply a corresponding increase in the uncertainty of z_{im} as well. In this case, we will obtain a joint posterior distribution $Q(\mathbf{b}, \alpha)$, which is then marginalized to get the individual distributions $Q(\mathbf{b})$ and $Q(\alpha)$. The derivation proceeds in a manner similar to that described in the previous sections. The crucial difference is that the marginal distribution over $\mathbf{b}$ is now a product of Student-t distributions instead of the gaussian distributions of sections 4.1 and 4.2. The following equations summarize the marginal posteriors for the graphical model of Figure 7a:

$$
\begin{aligned}
Q(\alpha) &= \text{Gamma}(\alpha; \hat{a}_\alpha, \hat{b}_\alpha) \\
Q(\mathbf{b}) &= \prod_{m=1}^{d} t_v\left(b_m; \mu_{b_m}, \sigma^2\right) \\
\hat{a}_\alpha &= a_{\alpha,0} + \frac{Nd}{2} \\
\hat{b}_\alpha &= b_{\alpha,0} + \sum_{m=1}^{d} \frac{1}{2\psi_{zm}} \\
&\qquad \left[\sum_{i=1}^{N} \langle z_{im}^2 \rangle - \left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm}\right)^{-1} \left(\sum_{i=1}^{N} \langle z_{im} \rangle \, f_m(\mathbf{x}_i)\right)^2\right] \\
\nu &= 2\hat{a}_\alpha \\
\mu_{b_m} &= \left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm}\right)^{-1} \left(\sum_{i=1}^{N} \langle z_{im} \rangle \, f_m(\mathbf{x}_i)\right) \\
\sigma_{b_m}^2 &= \frac{\hat{b}_\alpha \psi_{zm}}{\hat{a}_\alpha} \left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm}\right)^{-1},
\end{aligned}
\tag{4.10}
$$

where $\langle z_{im}^2 \rangle = \langle z_{im} \rangle^2 + \sigma_{z_m}^2$.

For the case of individual precision variables α_m shown in Figure 7b, we can derive an alternative model to Figure 6a having the following posterior distributions:

$$
\begin{aligned}
Q(\boldsymbol{\alpha}) &= \prod_{m=1}^{d} \text{Gamma}\left(\alpha_m; \hat{a}_{\alpha_m}, \hat{b}_{\alpha_m}\right) \\
Q(\mathbf{b}) &= \prod_{m=1}^{d} t_v\left(b_m; \mu_{b_m}, \sigma^2\right)
\end{aligned}
$$

$$
\begin{aligned}
\hat{a}_{\alpha_m} &= a_{\alpha_m,0} + \frac{N}{2} \\
\hat{b}_{\alpha_m} &= b_{\alpha_m,0} + \frac{1}{2\psi_{zm}} \\
&\quad \times \left[\sum_{i=1}^{N} \langle z_{im}^2 \rangle - \left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \right)^{-1} \left(\sum_{i=1}^{N} \langle z_{im} \rangle \, f_m(\mathbf{x}_i) \right)^2 \right] \\
\nu &= 2\hat{a}_\alpha \\
\mu_{b_m} &= \left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \right)^{-1} \left(\sum_{i=1}^{N} \langle z_{im} \rangle \, f_m(\mathbf{x}_i) \right) \\
\sigma_{b_m}^2 &= \frac{\hat{b}_\alpha^{(m)} \psi_{zm}}{\hat{a}_\alpha} \left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \right)^{-1},
\end{aligned}
\tag{4.11}
$$

where $\langle z_{im}^2 \rangle = \langle z_{im} \rangle^2 + \sigma_{z_m}^2$. This approximation can be used in conjunction with a distribution over the noise parameter ψ_y to derive a form of robust regression that is less sensitive to outliers than in our original formulation (where the predictive distribution over the output is a gaussian).

4.4 Initialization of Parameters. A few comments should be made regarding the initialization of priors used for the models in sections 4.1, 4.2, and 4.3. Specifically, the initial hyperparameter values $\{a_{\alpha,0}, b_{\alpha,0}\}$—or, for the ARD model, $\{a_{\alpha_m,0}, b_{\alpha_m,0}\}_{m=1}^{d}$—need to be set before running the EM algorithm. We set $a_{\alpha_m,0}$ and $b_{\alpha_m,0}$ so that the prior distribution over α_m is uninformative or "flat," using values of $a_{\alpha_m,0} = b_{\alpha_m,0} = 10^{-8}$, for all $m = 1, \ldots, d$. This means that the initial mean of α_m is 1, with high uncertainty—α_m has a rather flat prior distribution. These initial hyperparameter values can be used for all data sets and need never be modified. We use these values for all our experiments and data sets.

4.5 Obtaining Confidence Intervals. Given any of the two versions of VBLS, it is easy to obtain predictive distributions over regression outputs at query points. Marginalizing over the hidden z_{im} $\boldsymbol{\alpha}_m$ variables in the model gives us the following distribution over $y \mid \mathbf{x}, \mathcal{D}$, where $\mathcal{D}$ is the training data set,

$$
p(y \mid \mathbf{x}, \mathcal{D}) = \text{Normal}(y; \langle \mathbf{b} \rangle^T \mathbf{x}, \psi_y + \mathbf{1}^T \boldsymbol{\Psi}_\mathbf{z} \mathbf{1} + \mathbf{x}^T \boldsymbol{\Sigma}_\mathbf{b} \mathbf{x}), \tag{4.12}
$$

where $\boldsymbol{\Psi}_z$ is the noise variance of $\mathbf{z}$ and $\boldsymbol{\Sigma}_\mathbf{b}$ is the posterior covariance of $\mathbf{b}$.

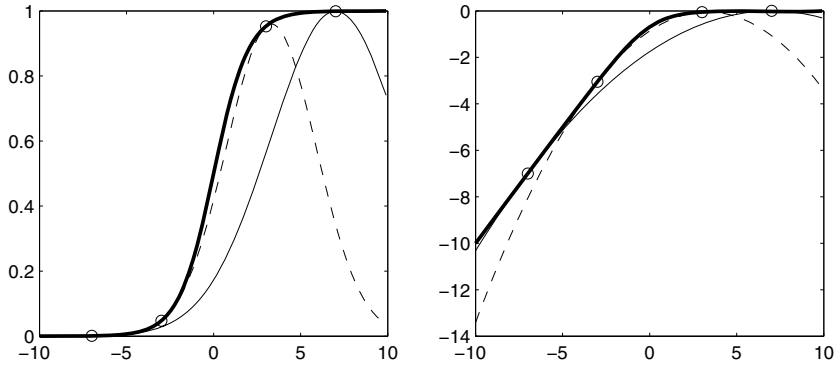


Figure 8: (Left) The logistic function (solid thick line) and two approximations with the variational parameters set to $\xi = 3$ (dashed line) and $\xi = 7$ (solid thin line). The points of tangency between the true function and the approximation are circled. (Right) The same plots on a log scale.

4.6 Extension to Classification. VBLS can be adapted to handle categorical outputs $y_i \in \{-1, +1\}$ by changing the target conditional distribution $p(y_i \mid \mathbf{z}_i)$ to a Bernoulli distribution via the sigmoid *link* function $g(x) = \left(1 + \exp(-x)\right)^{-1}$. In this case, the conditional distribution can be expressed as

$$p(y_i \mid \mathbf{z}_i) = \frac{1}{1 + \exp\left(-y_i \mathbf{1}^T \mathbf{z}_i\right)} = g(y_i \mathbf{1}^T \mathbf{z}_i).$$

Since this renders the posterior intractable due to nonconjugacy with $p(\mathbf{z}_i \mid \mathbf{x}_i)$, we follow Jaakkola and Jordan (2000) and introduce an additional lower bound using the inequality:

$$g(x) \geq g(\xi) \exp\left\{\frac{x - \xi}{2} - \varphi(\xi)\left(x^2 - \xi^2\right)\right\},$$

where $\varphi(\xi) = \tan(\xi/2)/4\xi$ and ξ is the variational parameter for the family of lower bounds to $g(x)$ (see Figure 8). Hence, we can lower-bound the likelihood $p(y_i \mid \mathbf{z}_i)$ by the parameterized version $p(y_i \mid \mathbf{z}_i, \xi_i)$ as follows:

$$\begin{aligned} p(y_i \mid \mathbf{z}_i) &= g(y_i \mathbf{1}^T \mathbf{z}_i) \\ &\geq p(y_i \mid \mathbf{z}_i; \xi_i) \\ &= g(\xi_i) \exp\left\{\frac{y_i \mathbf{1}^T \mathbf{z}_i - \xi_i}{2} - \varphi(\xi_i)\left(\mathbf{z}_i^T \mathbf{1}\mathbf{1}^T \mathbf{z}_i - \xi_i^2\right)\right\}. \end{aligned} \tag{4.13}$$

Note that this form is still an exponent of a quadratic in $\mathbf{z}_i$ which retains conjugacy with $p(\mathbf{z}_i \mid \mathbf{b}; \mathbf{x}_i)$ and allows us to proceed with our EM derivation as before—with the additional step that we must optimize the ξ_i parameters. We again start by writing out the log complete likelihood, which is the joint distribution over the known and unknown variables in the model:

$$\begin{aligned}
&\log p(y_i \mid \mathbf{z}_i, \mathbf{x}_i; \mathbf{b}, \psi_z) \\
&= \sum_{i=1}^{N} \log p(y \mid \mathbf{z}_i) + \sum_{i=1}^{N} \sum_{m=1}^{d} \log p(z_{im} \mid \mathbf{x}_i; b_m, \psi_{zm}) \\
&\geq \sum_{i=1}^{N} p(y \mid \mathbf{z}_i; \xi_i) + \sum_{i=1}^{N} \sum_{m=1}^{d} \log p(z_{im} \mid \mathbf{x}_i; b_m, \psi_{zm}) \\
&= \sum_{i=1}^{N} \left[\log g(\xi_i) + \frac{y_i \mathbf{1}^T \langle \mathbf{z}_i \rangle - \xi_i}{2} - \varphi(\xi_i) \left(\mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1} - \xi_i^2 \right) \right] \\
&\quad - \sum_{m=1}^{d} \left[\frac{N}{2} \ln \psi_{zm} + \frac{1}{2\psi_{zm}} \sum_{i=1}^{N} \left(z_{im} - b_m f_m(\mathbf{x}_i; \boldsymbol{\theta}_m) \right)^2 \right] + const,
\end{aligned} \tag{4.14}$$

where

$$\mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1} = \mathbf{1}^T \langle \mathbf{z}_i \rangle \langle \mathbf{z}_i \rangle^T \mathbf{1} + \mathbf{1}^T \boldsymbol{\Sigma}_{\mathbf{z}_i} \mathbf{1},$$

and $\boldsymbol{\Sigma}_{\mathbf{z}_i}$ is the posterior covariance of $\mathbf{z}_i$.

As it turns out, this additional approximation affects only the E-step equations, which are summarized as follows:

E-Step

$$\begin{aligned}
\mathbf{1}^T \boldsymbol{\Sigma}_{\mathbf{z}_i} \mathbf{1} &= \left(\sum_{m=1}^{d} \psi_{zm} \right) \left[1 - \frac{2\varphi(\xi_i)}{s_i} \left(\sum_{m=1}^{d} \psi_{zm} \right) \right] \\
\sigma_{zim}^2 &= \psi_{zm} \left(1 - \frac{2\varphi(\xi_i)}{s_i} \psi_{zm} \right) \\
\langle z_{im} \rangle &= b_m f_m(\mathbf{x}_i) + \frac{\psi_{zm}}{s_i} \left(\frac{y_i}{2} - 2\varphi(\xi_i) \mathbf{b}^T \mathbf{f}(\mathbf{x}_i) \right),
\end{aligned}$$

where $s_i = 1 + 2\varphi(\xi_i) \mathbf{1}^T \boldsymbol{\Psi}_{\mathbf{z}} \mathbf{1}$. The estimation of each ξ_i can be done by differentiating the expected log likelihood with respect

to each ξ_i:

$$\begin{aligned}
&\frac{\partial}{\partial \xi_i} \left\langle \ln p(\mathbf{y}, \boldsymbol{\theta} \mid \mathbf{X}) \right\rangle \\
&= \frac{\partial}{\partial \xi_i} \left[\ln g(\xi_i) + \frac{y_i \mathbf{1}^T \langle \mathbf{z}_i \rangle - \xi_i}{2} - \varphi(\xi_i) \left(\mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1} - \xi_i^2 \right) + const_{\xi_i} \right] \\
&= \underbrace{1 - g(\xi_i) - \frac{1}{2} + 2\xi_i \varphi(\xi_i)}_{=0} - \frac{\partial \varphi(\xi_i)}{\partial \xi_i} \left(\mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1} - \xi_i^2 \right).
\end{aligned}$$

Hence the likelihood is maximized by solving

$$\frac{\partial \varphi(\xi_i)}{\partial \xi_i} \left(\mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1} - \xi_i^2 \right) = 0,$$

which has solutions at $\partial \varphi(\xi_i)/\partial \xi_i = 0$ and at $\xi_i^2 = \mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1}$. One can show that the solution $\partial \varphi(\xi_i)/\partial \xi_i = 0$ occurs for the value $\xi_i = 0$ and actually corresponds to a minimum rather than a maximum of the expected log likelihood. Hence we have the admissible solutions for ξ_i being

$$\xi_i = \pm \sqrt{\mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1}}.$$

The sign of ξ_i can be chosen arbitrarily, since the likelihood is an even function of ξ_i, that is, both solutions result in the likelihood taking the same maximal value (cf. Figure 8). Importantly, the $O(d)$ complexity of all update equations is preserved even in the extension to categorical output data, making backfitting for classification an equally robust and efficient tool as its regression counterpart.

4.6.1 Bayesian Extension. Given that the functional approximation of equation 4.13 allows us to retain the conjugacy necessary for an analytical treatment, the Bayesian extensions of section 4 are straightforward to apply to our classification model. For the case in which we have a common shared precision parameter α across all regression parameters (cf. section 4.1), the b_m variables still have a posterior gaussian with the mean update as follows:

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

For the case in which we have an individual precision parameter α_m over each regression parameter (cf. section 4.2), the b_m variables again have a posterior gaussian with the mean update as follows:

$$\langle b_m \rangle^{(n+1)} = \left(\frac{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2}{\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \langle \alpha_m \rangle} \right) \langle b_m \rangle^{(n)} + \frac{\psi_{zm} \sum_{i=1}^{N} \frac{1}{s_i} \left(\frac{y_i}{2} - 2\varphi(\xi_i) \langle \mathbf{b} \rangle^{(n)^T} \mathbf{f}(\mathbf{x}_i) \right) f_m(\mathbf{x}_i)}{\left(\sum_{i=1}^{N} f_m(\mathbf{x}_i)^2 + \psi_{zm} \langle \alpha_m \rangle \right)}. \tag{4.15}$$

Importantly, the extension of VBLS to categorical output data preserves the $O(d)$ complexity of all update equations.

5 Extensions to Nonlinear Regression

While we derived VBLS in section 4 in the context of a linear model, it can also be used for supervised learning in nonlinear settings. In this section, we first discuss the relevance vector machine (RVM), a sparse Bayesian learning algorithm that operates in a framework similar to generalized linear regression.

The support vector machine (SVM; Cortes & Vapnik, 1995) is a common and popular method for classification problems, but it can be extended to regression, resulting in support vector regression (SVR). We show that the RVM serves as an alternative to SVR and can be derived as a highly efficient sparse algorithm with VBLS at its core. In section 6, we illustrate the computational advantages of this VBLS-RVM algorithm on experimental synthetic and real data sets.

5.1 Relevance Vector Machine (RVM). Introduced by Bishop and Tipping (2000) and Tipping (2001), the RVM uses the following generative, model,

$$y(\mathbf{x}; \mathbf{b}) = \sum_{i=1}^{N} b_i k(\mathbf{x}, \mathbf{x}_i) + \epsilon, \tag{5.1}$$

where $k(\mathbf{x}, \mathbf{x}_i)$ is a bivariate kernel function. The RVM creates N basis functions by centering a kernel function on each training data point $\mathbf{x}_i$, and these are linearly combined by a regression vector $\mathbf{b}$ to generate the prediction.

As in SVR, the goal of the RVM is accurate prediction of the target function while retaining as few basis functions as possible in the linear combination. That is, one hopes that the regression vector $\mathbf{b}$ remains as sparse as possible (as in the framework of sparse Bayesian learning). This can be achieved by introducing prior distributions over each element of $\mathbf{b}$, as discussed in section 4.2. The RVM's success at sparsifying the regression

solution hinges on the fact that this form of prior favors solutions that lie along the hyperspines of the distribution. The introduction of hyperparameters makes it impossible to obtain exact analytical posteriors (i.e., they are intractable). Nevertheless, we can obtain successful approximate solutions (albeit iteratively) by using the Laplace method (Tipping, 2001) or factorial variational approximations (Bishop & Tipping, 2000). Both of these approximations require hyperparameter updates for $\boldsymbol{\alpha}$ that need reestimation of the posterior covariance and mean of $\mathbf{b}$ as

$$\boldsymbol{\Sigma}_{\mathbf{b}} = \left(\operatorname{diag}\left(\langle \boldsymbol{\alpha} \rangle\right) + \left\langle \frac{1}{\psi_y} \right\rangle \sum_{i=1}^{N} \mathbf{k}_i \mathbf{k}_i^T \right)^{-1} \tag{5.2}$$

$$\mu_{\mathbf{b}} = \left(\frac{1}{\psi_y} \right) \boldsymbol{\Sigma}_{\mathbf{b}} \sum_{i=1}^{N} \mathbf{k}_i y_i, \tag{5.3}$$

where $\mathbf{k}_i \equiv [k(\mathbf{x}_1, \mathbf{x}_i), \ldots, k(\mathbf{x}_N, \mathbf{x}_i)]^T$ and ψ_y is the noise variance in the targets y. This requires an $O(N^3)$ Cholesky decomposition after each hyperparameter update. As the number of data samples increases, the RVM faces an explosion in the computational requirements, similar to that observed in gaussian processes and support vector machines. Indeed, as a generalized linear problem, each new data point adds an extra "dimension" to the input. Tipping (2001) mentions several enhancements to the algorithm, including pruning of unneeded basis functions, which may help speed up the RVM estimation. However, the crux of the problem remains that the expensive linear regression step must be performed after each hyperparameter update.

5.2 Variational Bayesian Least Squares RVM. We have so far not commented on the nature of the basis functions $f_m(\mathbf{x})$ in our model. Let us now switch to the RVM framework described above, where N basis functions are created by centering a bivariate kernel function $k(\mathbf{x}, \mathbf{x}')$ on each individual data point. This implies

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

where $1 \leq m \leq d$ and $d = N$. Notice that this transformation makes our VBLS model of Figure 6a equivalent to the RVM model discussed in section 5.1, with the notable difference that VBLS offers a significant advantage over the standard RVM in computational complexity. Note that while the computational complexity of a VBLS update is linear in the dimensionality of the problem, it is also linear in the number of data points, that is, $O(Nd)$. When cast into the RVM framework, using $d = N$ decreases the complexity from the $O(N^3)$ of RVM to $O(N^2)$. In particular, we emphasize the following:

- At each update of the α_m hyperparameters, the RVM requires an $O(N^3)$ Cholesky decomposition to reestimate the regression parameters,

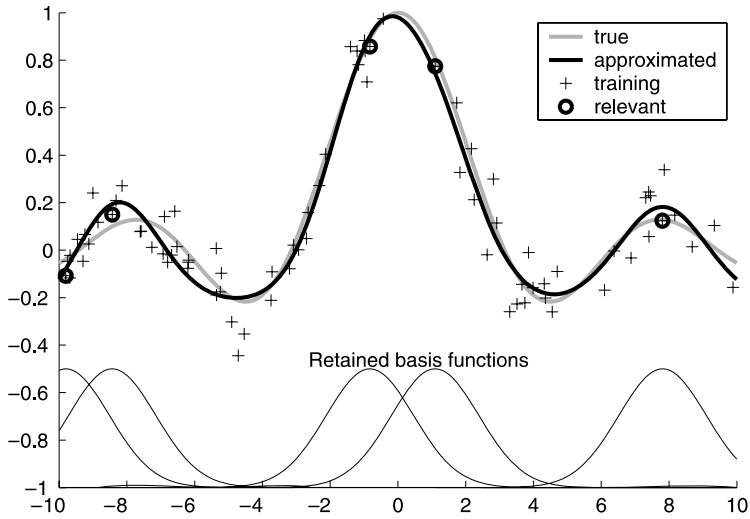


Figure 9: The VBLS-RVM solution to fitting data from the "sinc" function. Note that out of 50 data points, only 5 are considered "relevant." The retained basis functions (corresponding to the relevant points, indicated by black circles) are shown superimposed.

while discarding the estimate at the previous iteration. In the VBLS-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 (approximately 10 iterations were sufficient in our simulations). The savings in computation are especially evident when the number of data points (and hence the effective dimensionality) is large and when 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 VBLS updates to work with partially converged estimates, such that the brunt of computation is expended only to accurately estimate a variable when one is more confident about the variables in its Markov blanket.

As an illustrative example, Figure 9 shows results from using VBLS-RVM to fit a toy data set. This synthetic data set was generated using the one-dimensional sinc function $sinc(x) = sin(x)/x$, using the gaussian kernel:

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

where $\lambda > 0$.

Although VBLS-RVM is an order of magnitude faster than the standard RVM, it suffers no penalty in generalization error or in its ability to sparsify the set of basis vectors. More details on this are presented in section 6.3, where we compare the generalization performance of VBLS-RVM on a sinc function approximation problem to other competitive nonlinear regression techniques such as the RVM, SVR, gaussian process (GP) regression (Williams & Rasmussen, 1996), and locally weighted projection regression (LWPR) (Vijayakumar & Schaal, 2000).

Note that Tipping (2001) proposes an optimization of the distance metric λ, based on gradient ascent in the log likelihood. We can also compute such a gradient for VBLS-RVM as

$$\frac{\partial \langle \log p(\mathbf{y}, \mathbf{Z} \mid \mathbf{X}) \rangle}{\partial \lambda} = \sum_{j=1}^{N} \frac{b_j}{\psi_{zj}} \sum_{i=1}^{N} (\langle z_{ij} \rangle - b_j k_{ij})(x_i - x_j)^2 k_{ij}, \tag{5.4}$$

where $k_{ij} = k\left(x_i, x_j\right)$. Based on our experience, however, we caution against unconstrained maximization of the likelihood, especially over distance metrics. Instead, we recommend the route taken in the gaussian process community: treat these variables as hyperparameters and place prior distributions over them. Since exact solutions are typically intractable, we can either optimize them by using maximum a posteriori estimates (MacKay, 1999) or by Monte Carlo techniques (Williams & Rasmussen, 1996).

Note that there are several optimizations suggested by Tipping (2001) and Tipping and Faul (2003). These include pruning the basis functions when their precision variables indicate that they are unneeded and adopting a greedy (but potentially suboptimal) strategy in which the algorithm starts with a single basis function and adds candidates as necessary. We emphasize that our implementation of VBLS-RVM performs neither of these optimizations, although they are easy to introduce into our framework.

In the next section, we demonstrate experimentally on synthetic and real data sets that VBLS-RVM possesses a significant computational advantage over the RVM, while retaining the accuracy and sparseness of the standard RVM. It should be noted that in the RVM, each update of the hyperparameters requires a subsequent full matrix inversion to compute the posterior distribution over the regression parameters. In contrast, VBLS-RVM requires no matrix inversion. Moreover, each update of the hyperparameters causes the distribution over the regression coefficients to shift by only a small amount. VBLS-RVM performs especially well in these circumstances since it can use the current (partially good) solution and can update the regression coefficient distributions within a very small number of iterations rather than requiring a complete recalculation of the matrix inverse.

6 Experimental Results

We evaluate VBLS on synthetic and real data sets on both regression and classification problems, comparing it to other standard methods in order to show its competitive performance and computational advantage. First, we run VBLS on a synthetic data set where ground truth is known, in order to better evaluate its performance in a controlled setting. Then we apply VBLS, along with other standard methods, on neurophysiological data sets. Specifically, we would like to predict the electromyographic (EMG) activity of muscles from the neural data recorded in the primary motor (M1) cortices of monkeys, under the assumption that the relationship between neural and muscle activity is approximately linear.

We then move on to benchmark regression and classification data sets. We evaluate the generalization ability of VBLS-RVM along with state-of-the-art regression tools on popular nonlinear regression benchmark data sets. The algorithms discussed in section 2 are meant for linear regression problems and, hence, are unsuitable for these nonlinear regression benchmark data sets. Instead, we will compare VBLS-RVM to common nonlinear regression methods such as the RVM, SVR, gaussian process (GP) regression (Williams & Rasmussen, 1996), and locally weighted projection regression (LWPR) (Vijayakumar & Schaal, 2000).

Finally, we evaluate VBLS-RVM on benchmark classification problems. We compare the performance of VBLS-RVM to that of standard classification methods such as the RVM classifier and the SVM. Since the benchmark data sets that we consider involve two-class problems, we also include logistic regression for comparison since it is suited for classification problems with only two classes. Even though the algorithms surveyed in section 2 are for regression problems and could be augmented, making them suitable for classification problems by passing the outputs through a sigmoid function, we omit comparisons to them, choosing to draw comparisons between classifiers instead.

6.1 Synthetic Data

6.1.1 Data Sets. We generated random input training data consisting of 100 dimensions, 10 of which were relevant dimensions. The other 90 were either irrelevant or redundant dimensions, as we explain below. Each of the first 10 relevant input dimensions was drawn from a Normal$(0, 1)$ distribution. We then applied a random 10-dimensional rotation matrix to create input data with dimensions that are linear combinations of the original (unrotated) 10-dimensional data. The output data were then generated from the relevant input data using the vector $\mathbf{b} \in \Re^{10\times 1}$, where each coefficient of $\mathbf{b}$, b_m, was drawn from a Normal$(0, 100)$ distribution. Noise of varying levels was added to the outputs.

Noise in the outputs was parameterized with the coefficient of determination, r^2, of standard linear regression, defined as

$$r^2 = \frac{\left(\sigma_y^2 - \sigma_{res}^2\right)}{\sigma_y^2},$$

where σ_y^2 is the variance of the outputs and σ_{res}^2 is the variance of the residual error. We added noise scaled to the variance of the noiseless outputs $\bar{y}$ such that $\sigma_{noise}^2 = c\sigma_{\bar{y}}^2$, where $c = \frac{1}{r^2} - 1$. Results are quantified as normalized mean squared errors (nMSE), that is, the mean squared error on the test set normalized by the variance of the outputs of the test set. Note that the best normalized mean squared training error that can be achieved by the learning system under this noise level is $1 - r^2$, unless the system overfits the data. We used a value of $r^2 = 0.8$ for high-output noise and a value of $r^2 = 0.9$ for lower-output noise.

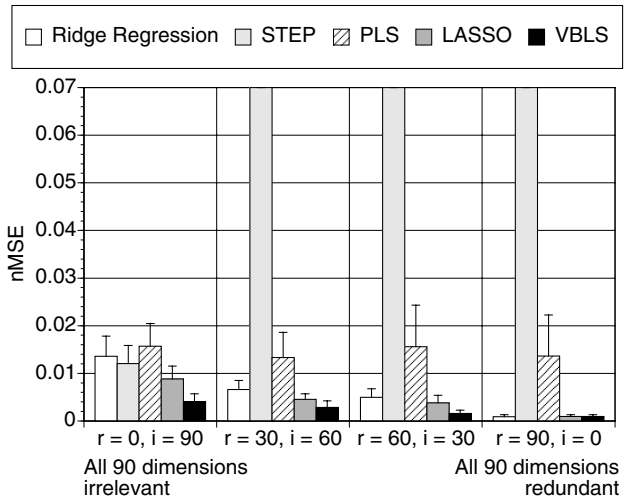
A varying number of redundant data vectors was added to the input data, generated from random convex combinations of the 10 relevant vectors. Finally, we added irrelevant data columns, drawn from a Normal(0,1) distribution, until 100 input dimensions were reached, generating training input data that contained irrelevant and redundant dimensions.

We created the test data set in a similar manner except that the input data and output data were left noise free. For our experiments, we considered a synthetic training data set with $N = 1000$ data samples and a synthetic test data set with 20 data samples. We examined the following four combinations of redundant, r, and irrelevant, i, input dimensions in order to better analyze the performance of the algorithms on different data sets:

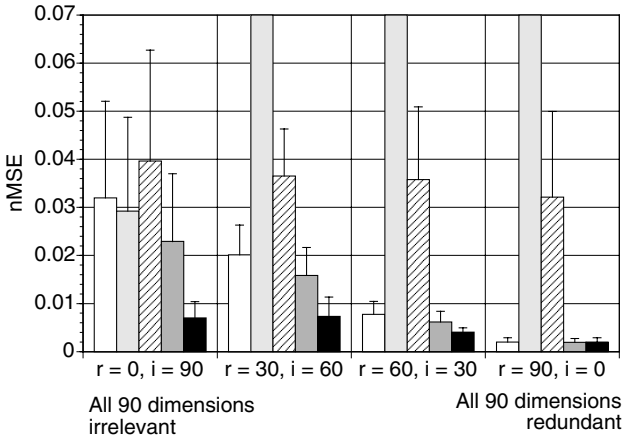
1. $r = 0, i = 90$ (all the 90 input dimensions are irrelevant).
2. $r = 30, i = 60$.
3. $r = 60, i = 30$.
4. $r = 90, i = 0$ (all the 90 input dimensions are redundant).

6.1.2 Methods. We compared VBLS to four other methods that were described in section 2: ridge regression, stepwise regression, PLS regression, and LASSO regression. For ridge regression, we introduced a small ridge parameter value of 10^{-10} to avoid ill-conditioned matrix inversions. We used Matlab's "stepwisefit" function to run stepwise regression. The number of PLS projections for each data set fit was found by leave-one-out cross-validation. Finally, we chose the optimal tuning parameter in LASSO regression using k-fold cross-validation.

6.1.3 Results. For evaluation, we calculated the prediction error on noiseless test data, using the learned regression coefficients from each technique. Results are quantified as normalized mean squared errors (nMSE). Figure 10



(a) Training data with low output noise ($r^2 = 0.9$)



(b) Training data with low output noise ($r^2 = 0.8$)

Figure 10: Average normalized mean squared prediction error for synthetic 100-input-dimensional data, averaged over 10 trials. The number of redundant dimension is denoted by r, and the number of irrelevant dimensions is i.

shows the average prediction error for noiseless test data, given training data where the output noise is either low ($r^2 = 0.9$) or high ($r^2 = 0.8$).

All the algorithms were executed on 10 randomly generated sets of data. The predictive nMSE results reported in Figure 10 were averaged over the 10 trials. Note that the best training nMSE values possible under the two noise conditions are 0.1 for the low-noise case and 0.2 for the high-noise

case. The training nMSE values were omitted for both graphs, since all algorithms attained training errors that were around the lowest possible values.

From Figures 10a and 10b, we see that regardless of output noise level, VBLS achieves either the lowest predictive nMSE value or a predictive nMSE value comparable to that of the other four algorithms. In general, as the number of redundant input dimensions increases and the number of irrelevant input dimensions decreases, the prediction error improves (i.e., it decreases). This may be attributed to the fact that redundancy in the input data provides more "information," making the problem easier to solve.

The performance of stepwise regression degrades as the number of redundant dimensions increases, as shown in Figures 10a and 10b, due to its inability to cope with collinear data. LASSO regression appears to perform quite well, compared to PLS regression and ridge regression, confirming previously published results that it can produce robust sparse regression solutions.

In summary, we can confirm that VBLS performs very well—as well as or better than classical robust regression methods (such as LASSO) on synthetic tests. Interestingly, PLS regression and ridge regression are significantly inferior in problems that have a large number of irrelevant dimensions. Stepwise regression has deteriorated performance as soon as colinear inputs are introduced.

6.2 Predicting EMG Activity from Neural Firing

6.2.1 Data Sets. We analyzed data from two different neurophysiological experiments (Sergio & Kalaska, 1998; Kakei, Hoffman, & Strick, 1999) involving monkeys trained to perform different arm movements while having their M1 neural firing rates and EMG activity recorded. The first experiment (Sergio & Kalaska, 1998) consisted of a monkey that applied either a movement or isometric force to a manipulandum in a center-out task in eight directions, equally spaced in a horizontal planar circle. They recorded neural activity of 71 M1 neurons in all conditions, along with the EMG outputs of 11 muscles, resulting in 2320 data samples for each neuron-muscle pair. In the second experiment (Kakei et al., 1999), a monkey was trained to perform eight different combinations of wrist flexion-extension and radial-ulnar movements while in three different arm postures. The data set consisted of neural data of 92 M1 neurons that were recorded at all three wrist postures, along with the resulting EMG outputs of seven contributing muscles, resulting in 2616 data samples for each neuron-muscle pair. In all experiments, each data sample consisted of the average firing rates from a particular neuron (averaged over a window of 10 msec) and the corresponding EMG activation from a particular muscle.

The goal of the analysis was to determine how well VBLS compares to other techniques when reconstructing EMG data of each muscle. The

relationship between neural and muscle activity is assumed to be linear, such that the basis functions in VBLS are simply a copy of the respective input dimensions: $f_m(\mathbf{x}) = x_m$.

6.2.2 Methods. To analyze the data set, we applied VBLS, as described in section 4.2, along with a selection of methods discussed in section 2: ridge regression, stepwise regression, PLS regression, and LASSO regression. We omitted the other methods due to unsuitable model assumptions or computationally expensive procedures.

A baseline comparison of good EMG construction was obtained through a limited combinatorial search over possible regression models. This approach served as our baseline study (referred to ModelSearch in the figures) and served as our gold standard, with a particular model being characterized by the subset of neurons used to predict the EMG data. For a data set with n neurons, the number of possible models that exist for a particular muscle is

$$\sum_{m=1}^{n} \binom{n}{m} = 2^n - 1,$$

since the order of contributing neurons is not important (only combinations not permutations of neurons are considered). The number of possible models given in the expression above is too large for an exhaustive search. As a result, for ModelSearch, we considered all possible combinations of neurons of up to a subset of the total number of neurons in the data set. For example, we considered only possible combinations of up to 20 neurons for the Sergio and Kalaska (1998) data set to give $2^{20} - 1$ possible models instead of the full, exhaustive $2^{71} - 1$ possible models to search over. Even so, ModelSearch required several weeks of computation on a 30-node cluster computer. We determined the optimal predictive subset of neurons using cross-validation[4] and used ordinary least squares regression to calculate the training and generalization performance. For both the Sergio and Kalaska (1998) and Kakei et al. (1999) data sets, the cross-validation procedure used in the baseline study (ModelSearch) was used to determine the optimal subset of neurons and was done in the context of the behavorial experiments and not in a statistically randomized way. While we realize cross-validation has a danger of overfitting, the purpose of ModelSearch is to serve as a method for comparison to evaluate the performance of all algorithms and to give some indication of a baseline performance that is

[4]Eight-fold cross-validation was used for the Sergio and Kalaska (1998) data set (with a training set consisting of 50% of data and two test sets containing 25% of the data each). Six-fold cross-validation sets were used for the Kakei et al. (1999) data set (with the data split evenly between a training set and a test set).

Table 1: Percentage of M1 Neuron Matches Between the Baseline and All Other Algorithms, Averaged over All Muscles in Each Data Set.

Data Sets	STEP	PLS	LASSO	VBLS
Sergio and Kalaska (1998) M1 neural data set	7.2%	7.4%	6.4%	94.2%
Kakei et al. (1999) M1 neural data set	65.1	42.9	80.6	94.3

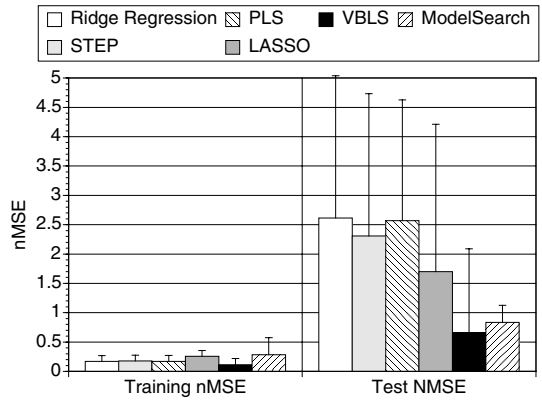
achievable using a crude combinatorial search of a subset of models. We compared ModelSearch with ridge regression, stepwise regression, PLS regression, LASSO regression, and VBLS. We used the same validation sets for these five algorithms as in ModelSearch in order to perform a consistent comparison.

For ridge regression, we introduced a small ridge parameter value of 10^{-10} to avoid illconditioned matrix inversions. We used Matlab's stepwisefit function for stepwise regression. The number of PLS projections for each data fit was found by leave-one-out crossvalidation. Finally, the optimal value of the open parameter in LASSO regression was chosen using cross-validation.

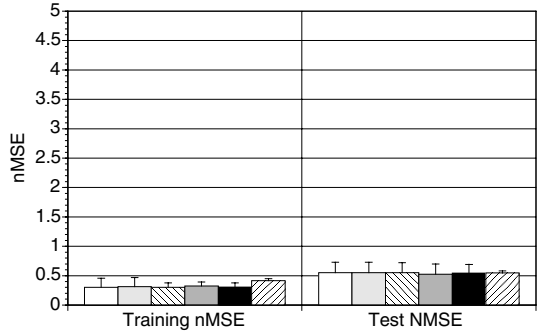
The baseline method ModelSearch identified a subset of neurons as relevant. Table 1 shows the percentage match of relevant neurons found by the algorithms, relative to the relevant neurons found by ModelSearch. The final set of relevant neurons used in Table 1 was reached by each algorithm (except VBLS) by taking the common neurons found to be relevant over the eight cross-validation sets for the Sergio and Kalaska (1998) data set (or six cross-validation sets for the Kakei et al. (1999) data set). The relevant neurons found by VBLS and reported in Table 1 were obtained by using the entire data set since VBLS does not require the data be divided into separate training or test sets.

Inference of relevant neurons in PLS was based on the subspace spanned by the PLS projections, while relevant neurons in VBLS were inferred from the magnitude of the $\boldsymbol{\alpha}$ vector (the corresponding α_m of an irrelevant input dimension m would have an extremely large value compared to the α values of the relevant dimensions such that we can use a threshold value to find irrelevant dimensions).[5] The number of relevant neurons from stepwise regression and LASSO regression, were determined from the inputs that were included in the final model. Note that since ridge regression retained all input dimensions, this algorithm was omitted in relevant neuron comparisons.

[5]If we choose to factor the marginal posterior so that $Q(\mathbf{Z}, \mathbf{b}, \alpha) = Q(\mathbf{Z})Q(\mathbf{b}, \alpha)$ such that the marginal distribution for $\mathbf{b}$ is a Student-t distribution, then we can perform t-tests on the regression coefficients to find relevant dimensions, using a significance of $p < 0.05$. Refer to Ting et al. (2005) for details.



(a) Sergio and Kalaska (1998) M1 neural data



(b) Kakei et al. (1999) M1 neural data

Figure 11: Average normalized mean squared error for M1 neurons, averaged over all cross-validation sets and over all muscles. Six-fold cross-validation was used for the Kakei et al. (1999) M1 neural data set, and eight-fold cross-validation was used for the Sergio and Kalaska (1998) M1 neural data set.

6.2.3 Results. Figure 11 shows that EMG traces appear to be generally well predictable from M1 neural firing. In particular, VBLS had a generalization error comparable to that of the baseline study. Figure 12 illustrates the EMG trace predicted by VBLS for a sample muscle (muscle 7) from the Sergio and Kalaska (1998) neural data set. Figure 11b shows that all algorithms achieve similarly low prediction errors on the Kakei et al. (1999) neural data set. On the Sergio and Kalaska (1998) data set, however, ridge regression, stepwise regression, LASSO regression, and PLS performed far worse, with ridge regression attaining the worst error. The difference between prediction errors on both neural data sets suggests that the Sergio and Kalaska (1998) neural data set is somehow much richer and, hence, more challenging to analyze.

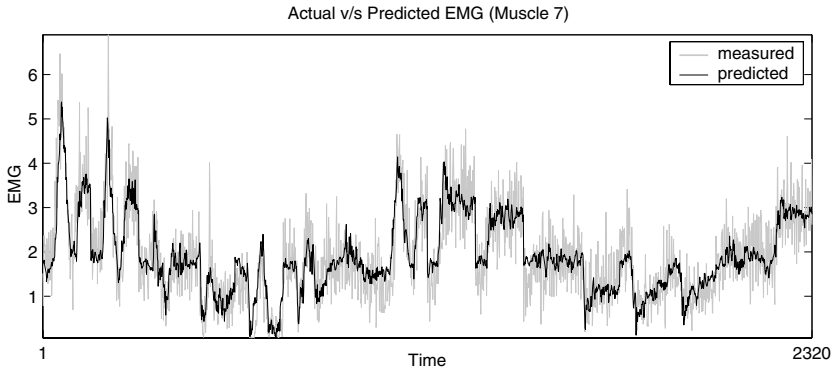


Figure 12: VBLS predicts muscle activity from neural firing rate for muscle 7 from the Sergio and Kalaska (1998) M1 neural data set.

Note that the average number of relevant M1 neurons found by VBLS, averaged over all 11 muscles in the Sergio and Kalaska (1998) data set and averaged over all 7 muscles in the Kakei et al. (1999) data set, was slightly higher than ModelSearch.[6] This is hardly surprisingly given that ModelSearch did not consider all possible combinations of neurons in both neural data sets. In contrast, VBLS considered all 71 neurons in the Sergio and Kalaska (1998) data set and all 92 neurons in the Kakei et al. (1999) data set.

Table 1 compares how the various methods fare in terms of finding relevant neurons, using the results of ModelSearch as a baseline comparison. As a result, a higher percentage match in the table does not necessarily mean that the method should result in a lower prediction error. Regardless, we see from Table 1 that the relevant neurons identified by VBLS coincided at a very high percentage with those of ModelSearch, while PLS regression and stepwise regression had inferior outcomes.

LASSO regression matched a high percentage of relevant M1 neurons in the Kakei et al. (1999) data set but failed to perform as well on the Sergio and Kalaska (1998) data set. As an aside, it is possible to use VBLS as a preprocessing step to reduce the search space of possible models for ModelSearch to consider.

The consistent and good generalization properties of VBLS on all neural data sets, as shown in Figures 11a and 11b, suggest that the Bayesian approach of VBLS sufficiently regularizes the participating neurons such that no overfitting occurs, despite finding a larger number of relevant neurons. The performance of VBLS on these particularly difficult data sets shows that it is a viable alternative to traditional generalized linear regression

[6]More details on the neural interpretation of this analysis can be found in Ting et al. (2005).

tools. Even with the additional Bayesian inference for ARD, it maintains its algorithmic efficiency since no matrix inversion is required. While VBLS is an iterative statistical method, which performs more slowly than classical "one-shot" linear least squares methods (i.e., on the order of several minutes for the data sets in our analysis on a standard PC),[7] it achieved comparable results with our baseline combinatorial model search, which took weeks on a cluster computer.

6.3 Benchmark Regression Problems

6.3.1 Data Sets. To evaluate the generalization ability of VBLS-RVM, we compared it to other state-of-the art nonlinear regression tools on the following benchmark data sets:

- Synthetic sinc data set (generated in the same way as Tipping, 2001)
- Boston Housing data set
- Abalone data set[8]
- Netflix Prize data set[9]

The sinc data set was constructed to have 100 uniformly spaced samples in $[-10, 10]$ and uniform noise in $[-0.2, 0.2]$ was added to the targets. The Boston Housing data set had 14 attributes and was split randomly in 10 different random splits into a training set of 404 samples and a test set of 102 samples (e.g., 20% of data used for test and the rest for training). The Abalone data set had 10 attributes and was downsampled to 10 disjoint sets, also with 20% of the data randomly selected for test and the remainder for training (3327 samples for training and 850 samples for test).

The Netflix Prize data set consisted of 17,770 movies and 480,189 customers, with each movie having reviews submitted by a small subset of customers. Each review consisted of a rating (from 0 to 5 stars) and the date that the review was made. Customers were identified with a unique customer identification number. Movies were also identified with a unique movie identification movie, with titles of movies additionally available. The data set was downsampled so that only 355 movies and 1412 customers were randomly selected.

We formulated the Netflix Prize problem as a linear regression problem in order to see how simple linear methods performed on a real-world, complex data set and also to compare VBLS to the other methods. The goal was to predict the rating that a customer c gives to a movie m, given we have access to all the ratings movie m has received and all the ratings customer c has made. The downsampled data consisted of 7249 samples,

[7] Pentium IV class machine, 1.7GHz.

[8] Both the Boston housing and Abalone data sets are available from the UCI Repository.

[9] The data set is available online from http://www.netflixprize.com.

Table 2: Average Normalized Mean Squared Error over 10 Trials of RVM, SVR, GP Regression, LWPR, and VBLS-RVM on Benchmark Regression Data Sets.

Data Sets	RVM	SVR	GP	LWPR	VBLS-RVM
Sinc	0.0134	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

with each sample consisting of a 1767-dimensional input vector (i.e., all the ratings that movie m received from each customer and all the ratings made by customer c—this gives $1412 + 355$ or 1767 elements in the input vector) and an output scalar (i.e., the number of stars that customer c gave to movie m). 10% of the downsample data was used for test, while the remaining was used for training (6524 training samples and 725 test samples).

6.3.2 Methods. For all the benchmark data sets (except for the Netflix Prize data set), we compared VBLS-RVM with other algorithms suited for nonlinear regression such as the standard RVM, SVR,[10] gaussian process (GP) regression, and locally weighted projection regression (LWPR). Both VBLS-RVM and the RVM used gaussian kernels with distance metrics optimized by five-fold cross-validation. The gaussian process regression algorithm used a radial basis function (RBF) covariance function with automatic hyperparameter optimization.

For the Netflix Prize problem, we evaluated the following linear methods that were previously evaluated in sections 6.1 and 6.2: ridge regression, stepwise regression, PLS regression, LASSO regression (using the Gauss-Seidel method of (Shevade & Keerthi, 2003), in order to accommodate the size of the data), and VBLS. Note that the number of optimal projection directions to use in PLS regression (i.e., the parameter K described in section 2.1.6) was set to 20 after much user tuning—numbers larger than this took far too long to run. Additionally, the number of steps for stepwise regression was capped to 100. Larger step values were explored, but the running time was excessively long, with poorer generalization performance.

6.3.3 Results. Table 2 shows the normalized mean squared errors on the test sets averaged over 10 experimental trials. We can see that VBLS-RVM provides an extremely competitive solution in terms of generalization ability when compared to other popular regression methods.

In order to examine the level of "sparsification" of the set of basis functions, we compared the average number of basis functions retained by the RVM, SVR, and VBLS-RVM since these three methods had the ability to

[10] RVM and SVR results adapted from Tipping (2001).

Table 3: Number of "Relevant" Vectors Retained by RVM, SVR, and VBLS-RVM for Benchmark Regression Data Sets.

Data Sets	RVM	SVR	VBLS-RVM
Sinc	6.7	45.2	4.8
Boston	39	142.8	57.4
Abalone	437	1320	368

Table 4: Average Computation Time in Seconds for RVM and VBLS-RVM on Benchmark Regression Data Sets.

Data Sets	RVM	VBLS-RVM	N_{training}	N_{test}	d
Sinc	18.71 sec	6.24 sec	100	100	1
Boston	372 sec	155 sec	404	102	13
Abalone	2767 sec	428 sec	3327	850	10

Note: N_{train}, N_{test}, and d are the number of training samples, test samples, and input dimensionality, respectively.

retain "relevant" samples. Table 3 shows the average number of relevant samples (known as "relevant vectors" in the RVM) retained in the final solution (averaged over all 10 experimental trials) on the sinc, Boston Housing, and Abalone data sets.

The experiments demonstrate that VBLS-RVM is a competitive regression solution when compared to other current state-of-the-art statistical methods in both its generalization ability and efficacy as a sparse Bayesian learning algorithm. However, the main advantage of VBLS-RVM is in its computation time relative to the RVM. We compare the execution time of the RVM to VBLS-RVM in order to examine how much speed is gained by incorporating VBLS into the RVM. Table 4 gives the average execution time in seconds required by the RVM and VBLS-RVM for convergence of their regression parameter estimates on the sinc, Boston Housing, and Abalone data sets. The table also shows the number of training samples N_{training} the number of test samples N_{test}, and the dimensionality of the inputs d. Note that the number of $O(N^2)$ updates to $\mathbf{b}$ per update cycle of the hyperparameters is very small (around 10 updates), 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 RVM can significantly gain from the iterative nature of the VBLS generalized linear regression procedure.

The baseline method, Cinematch, used for comparison in the Netflix Prize competition, reported a predictive, root mean squared error values (RMSE) of 0.9514 on a quiz (unknown) test subset.[11] Table 5 shows the

[11]The Netflix Prize competition offers a grand prize for a rmse achieved that is $\leq$ 0.8563.

Table 5: RMSE of Ridge Regression, Stepwise Regression, PLS Regression, LASSO Regression, and VBLS on a Downsampled Version of the Netflix Prize Data Set.

	Training RMSE	Test RMSE	Time Taken
Cinematch (baseline)	NA	0.9514[b]	NA
Ridge regression	0.7527	18.59	5.83 sec
Stepwise regression	13.39	6.375	1140 sec
PLS regression[a]	0.9399	1.3280	247.4 sec
LASSO regression	0.9999	1.0594	8.872 sec
VBLS regression	1.0219	1.0443	104.7 sec

Notes: The downsampled data set has 6524 training samples and 725 test samples, with an input dimensionality of 1767. Computation time for training is shown in seconds.
[a]In the interest of constraining the running time, the maximum number of projections was capped at 20.
[b]The results reported by the baseline, Cinematch, is on a separate test subset, as provided by the Netflix Prize competition.

predictive RMSE for all five linear methods evaluated. We can see that the best performance of a linear method is actually not so far off from the baseline performance of Cinematch—although the test sets used are different, so such a comparison cannot be fairly made. VBLS had a comparable performance to LASSO regression but took a longer time to train than the other linear methods. However, unlike the other methods, with the exception of LASSO regression, whose training times were reported after the optimal values of open parameters were set or determined, VBLS did not have any open parameters to be set or cross-validated and performed "out of the box."

6.4 Benchmark Classification Problems

6.4.1 Data Sets. We also evaluated the classification accuracy of the RVM, SVM, and VBLS-RVM on some benchmark data sets. To facilitate comparison, we trained and tested VBLS-RVM on exactly the same data used in Tipping (2001), along with an additional real-world large-scale data set. The following data sets were used for comparison:

- Ripley's synthetic data (Ripley, 1996)
- Banana data set (Rätsh et al., 2001)
- Pima Diabetes data set
- Binary classification problem of the MNIST database of handwritten digits[12]

All data sets presented a two-class classification problem.

[12]The MNIST data set is publicly available online from http://yann.lecun.com/exdb/mnist. It is a popular benchmark data set that has been analyzed in various forms by many (e.g., Lecun, Bottou, Bengio, & Haffner, 1998; Keerthi, Chapelle, & DeCoste, 2006).

Ripley's synthetic data were generated, as described in Tipping (2001), in two dimensions from mixtures of two gaussians with the classes overlapping such that the Bayes error is around 8%. A training set of 100 samples was randomly chosen from Ripley's original 250-sample training data set. The test error was computed using a test set of 1000 samples. The Banana data set was taken from the online repository and consisted of 100 training and test splits provided by Rätsch et al. (2001). Each training set had 400 training samples, and each test set had 4900 samples. Both had two-dimensional inputs. The Pima Diabetes set had, seven-dimensional inputs and was split into a training set with 200 samples and a test set with 332 test samples.

Finally, we formulated the MNIST handwritten digit data set into a binary classification problem for the purpose of distinguishing the digit 0 from all other digits (i.e., a binary classification problem). The complete MNIST database consists of binary images of handwritten digits (numbered 0 to 9). It has 60,000 training samples, 10,000 test samples, and an input dimensionality of 784. We conducted experiments with varying sizes of training data in order to evaluate the performance and run times of various methods. For example, we considered training set sizes of 1000, 3000, 5000, and 10,000 samples and test sets of 500 samples.

6.4.2 Methods. We compared VBLS-RVM to an RVM classifier, the SVM, and logistic regression. As mentioned previously, the lists of algorithms surveyed in section 2 are designed for regression problems. Though we could augment them and make them suitable for classification by passing the outputs through a sigmoid function, we omit them, choosing instead to make comparisons to native classifiers.

The RVM classifier used a gaussian kernel, $k(\mathbf{x}_m, \mathbf{x}_n) = \exp(-r^{-2} \parallel \mathbf{x}_m - \mathbf{x}_n \parallel^2)$, with the width parameter r set to 0.5. The error/margin trade-off parameters of the SVM were tuned using five-fold cross-validation. Both VBLS-RVM and the RVM used gaussian kernels with distance metrics optimized by five-fold cross-validation.

For the MNIST data set, we evaluated the following methods: (1) the primal SVM of Keerthi et al. (2006), designed for fast performance on large-scale data sets; (2) the fast RVM of Tipping and Faul (2003), (3) VBLS-RVM, and (4) the original RVM of Tipping (2001). Other competitive classifiers, aside from the primal SVM (Keerthi et al., 2006), include sparse multinomial logistic regression (Krishnapuram, Carin, Figueiredo, & Hartemink, 2005) and the doubly regularized SVM (Wang, Zhu, & Zou, 2006), to list a few. Note that the original RVM does not scale well to large-scale data sets due to its $O(N^3)$ computational complexity (per EM iteration).

6.4.3 Results. Table 6 shows the classification accuracies of all the algorithms on the benchmark data sets. Results for the Ripley data set were averaged over 10 experimental trials, while the results for the Banana data

Table 6: Average Classification Percentage Accuracies on Standard Classification Benchmark Data Sets for the RVM, SVM, VBLS-RVM, and Logistic Regression.

	RVM	SVM	VBLS-RVM	Logistic	N_{training}	N_{test}	d
Ripley	9.75%	10.8%	9.66%	11.37%	100	1000	2
Banana	10.8	10.9	11.2	11.0	400	4900	2
Pima Diabetes	19.8	20.1	19.6	22.2	200	332	7

Notes: Results were averaged over 10 experimental trials for the Ripley synthetic data set and over the first 10 training and test splits for the Banana data set. N_{train}, N_{test}, and d are the number of training samples, test samples, and input dimensionality, respectively.

Table 7: Classification Percentage Accuracies (Prediction Errors) for the MNIST Handwritten Digit Data Set for Training Sets of Various Sizes.

	N_{training}	N_{test}	Test Error	Time Taken
Primal SVM (Keerthi et al., 2006)	1000	500	13.4%	5.888 sec
Fast RVM (Tipping and Faul, 2003)	1000	500	11.8	10.033 sec
VBLS-RVM	1000	500	7.6	0.9761 sec (55.56 sec)
RVM (Tipping, 2001)	1000	500	7.6	164.9 sec
Primal SVM	3000	500	14.2	12.88 sec
Fast RVM	3000	500	9	10.56 sec
VBLS-RVM	3000	500	7.6	15.21 sec (401.9 sec)
RVM	3000	500	7.6	4295.3 sec
Primal SVM	5000	500	14.4	20.70 sec
Fast RVM	5000	500	7.8	23.23 sec
VBLS-RVM	5000	500	7.6	56.77 sec (1123.2 sec)
RVM	5000	500	7.6	1685.7 sec
Primal SVM	10,000	500	14.2	38.83 sec
Fast RVM	10,000	500	7.6	43.51 sec
VBLS-RVM	10,000	500	7.6	238.7 sec (4171.6 sec)
RVM	10,000	500	NA	NA

Notes: N_{training} and N_{test} are the number of training samples and test samples, respectively. The dimensionality of the input data is 784.

set were averaged over the first 10 training and test splits. The classification errors reported for the Pima Diabetes set were for the given training and test sets. We see from the table that the classification accuracy of VBLS-RVM is comparable to and as competitive as that of the RVM classifier, SVM, and logistic regression.

Table 7 reports the classification percentage accuracies on test sets, as well as the running times for each method. The computation time of VBLS was reported with and without the preprocessing step (i.e., the one-time step where the training and test design matrices of the RVM's basis vectors are constructed. The computational time of VBLS with preprocessing is shown

in brackets. The construction of the training and test matrices, which have dimensions $N_{\text{training}} \times N_{\text{training}}$ and $N_{\text{training}} \times N_{\text{test}}$, respectively,[13] takes the bulk of the computation time for VBLS. As such, it is not surprising that VBLS-RVM takes longer computational times than the fast RVM and SVM. The VBLS-RVM could be modified to accommodate large-scale data sets by greedily adding basis vectors to the design matrix (similar to that done in Tipping & Faul, 2003).

On average, the fast RVM of Tipping and Faul (2003) performs faster than the RVM and the VBLS-RVM, which is unsurprising given that the modified RVM adds basis vectors in a greedy fashion, potentially converging on a suboptimal model. Indeed, we see in Table 7 that the fast RVM is not necessarily the best performing. However, the speed advantage that VBLS-RVM offers over the standard RVM is easily observed in the table.

7 Discussion

For efficient learning and feature selection, we introduce a new Bayesian technique for linear regression analysis with automatic regularization called variational Bayesian least squares, focusing on scenarios with large samples of high-dimensional data that are commonly found in the application domains of robotics and brain-machine interfaces. Although derived in a linear regression model, VBLS can also be extended to nonlinear regression and classification settings, as done, for example, in VBLS-RVM. VBLS is competitive with classical linear regression and other sparse regression techniques and, furthermore, does not require any manual parameter tuning, giving it a black box statistical property. The iterative nature of VBLS makes it suitable for real-time, incremental learning (when decisions need to be made quickly) and allows it to be embedded in other iterative methods to offer a speed-up advantage.

One issue is the effect of the variational approximation used in the algorithm on the quality of function fit. One could assume, for example, that VBLS may tend to overfit, since factorial approximations to a joint distribution are known to create more peaked distributions. However, since the factorial approximation is made over the regression coefficients, a more peaked distribution ensures only that the regression coefficients are closer to zero, making VBLS slightly pessimistic and unlikely to overfit.

VBLS can also be applied to other problems such as parameter identification in noisy high-dimensional regression (Ting, D'Souza, & Schaal, 2006), growing mixtures of experts or locally weighted regression in high-dimensional spaces. In these scenarios, use of VBLS can potentially help overcome numerical matrix inversion operations that may otherwise make an algorithm too computationally expensive to be viable for

[13] Note that N_{training} and N_{test} are the number of training and test samples, respectively.

high-dimensional or real-time learning. We are currently pursuing such research directions as future work.

Acknowledgments

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

References

Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics: Identifying influential data and sources of collinearity*. New York: Wiley.

Bishop, C. M., & Tipping, M. E. (2000). Variational relevance vector machines. In C. Boutilier & M. Goldszmidt (Eds.), *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence* (pp. 46–53). San Francisco: Morgan Kaufmann.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297.

Derksen, S., & Keselman, H. (1992). Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables. *British Journal of Mathematical and Statistical Psychology, 45*, 265–282.

Efron, B., Hastie, T., Johnstone, L., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics, 32*, 407–499.

Everitt, B. S. (1984). *An introduction to latent variable models*. London: Chapman and Hall.

Fahlman, S. E., & Lebiere, C. (1989). The cascade-correlation learning architecture. In D. Touretzty (Ed.), *Advances in neural information processing systems, 2*. Cambridge, MA: MIT Press.

Feder, M., & Weinstein, E. (1988). Parameter estimation of superimposed signals using the EM algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing, 36*(4), 477–490.

Frank, I., & Friedman, J. (1993). A statistical view of some chemometric regression tools. *Technometrics, 35*, 109–135.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software, 3*, 209–226.

Friedman, J., Hastie, T., & Tibshirani, R. (2007). *Pathwise coordinate optimization* (Tech. Rep.). Stanford, CA: Department of Statistics, Stanford University.

Fukumizu, K., Bach, F. R., & Jordan, M. I. (2006). *Kernel dimensionality reduction in regression* (Tech. Rep.). Berkeley: Department of Statistics, University of California, Berkeley.

Gelman, A., Carlin, J., Stern, H., & Rubin, D. (2000). *Bayesian data analysis*. London: Chapman and Hall.

Ghahramani, Z., & Beal, M. (2000a). Graphical models and variational methods. In D. Saad & M. Opper (Eds.), *Advanced mean field methods: Theory and practice.* Cambridge, MA: MIT Press.

Ghahramani, Z., & Beal, M. J. (2000b). Variational inference for Bayesian mixtures of factor analyzers. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 12*. Cambridge, MA: MIT Press.

Ghahramani, Z., & Hinton, G. E. (1997). *The EM algorithm for mixtures of factor analyzers* (Tech. Rep. No CRG-TR-96-1). Toronto: University of Toronto.

Goodman, J. (2004). Exponential priors for maximum entropy models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. San Francisco: Morgan Kaufmann.

Gray, A. G., & Moore, A. (2001). N-body problems in statistical learning. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems, 13*. Cambridge, MA: MIT Press.

Hastie, T., & Tibshirani, R. (1990). *Generalized additive models*. London: Chapman and Hall.

Hastie, T., & Tibshirani, R. (2000). Bayesian backfitting. *Statistical Science, 15*(3), 196–223.

Jaakkola, T. S., & Jordan, M. I. (2000). Bayesian parameter estimation via variational methods. *Statistics and Computing, 10*(1), 25–37.

Jeffreys, H. (1946). An invariant form for the prior probability in estimation problems. *Journal of the Royal Statistical Society. Series A, 186*, 453–461.

Kakei, S., Hoffman, D., & Strick, P. (1999). Muscle and movement representations in the primary motor cortex. *Science, 285*, 2136–2139.

Keerthi, S. S., Chapelle, O., & DeCoste, D. (2006). Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research, 7*, 1493–1515.

Kim, S.-J., Koh, K., Lustig, M., Boyd, S., & Gorinevsky, D. (2007). A method for large-scale l1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing, 1*(4), 606–617.

Komarek, P., & Moore, A. (2000). A dynamic adaptation of AD-trees for efficient machine learning on large data sets. In *Proceedings of the 17th International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

Krishnapuram, B., Carin, L., Figueiredo, M. A., & Hartemink, A. J. (2005). Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(6), 957–968.

Lecun, Y., Bottou, A., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*. Piscataway, NJ: IEEE.

Lee, S., Lee, H., Abbeel, P., & Ng, A. (2006). Efficient L1-regularized logistic regresssion. In *Proceedings of the 21st National Conference on Artificial Intelligence*: Menlo Park, CA: AAAI Press.

Lokhorst, J. (1999). *The LASSO and generalized linear models* (Tech, Rep.). South Adelaide: Department of Statistics, University of South Adelaide, South Australia, Australia.

MacKay, D. J. C. (1999). Comparison of approximate methods for handling hyperparameters. *Neural Computation, 11*(5), 1035–1068.

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

Moore, A. (2000). The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In C. Boutilier & M. Goldszmidt (Eds.), *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence* (pp. 397–405). San Francisco: Morgan Kaufmann.

Moore, A., & Lee, M. S. (1998). Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research, 8*, 67–91.

Neal, R. (1994). *Bayesian learning for neural networks*. Unpublished doctoral dissertation, University of Toronto.

Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the 21st International Conference on Machine Learning*. New York: ACM Press.

Omohundro, S. M. (1990). Bumptrees for efficient function, constraint and classification learning. In D. Touretzky & R. Lippmann (Eds.), *Advances in neural information processing systems, 3*. Cambridge, MA: MIT Press.

Parisi, G. (1988). *Statistical field theory*. Reading, MA: Addison-Wesley.

Rätsh, G., Onoda, T., & Müller, K.-R. (2001). Soft margins for AdaBoost. *Machine Learning, 42*(3), 287–320.

Ripley, B. D. (1996). *Pattern recognition and neural networks*. Cambridge: Cambridge University Press.

Rustagi, J. (1976). *Variational methods in statistics*. Orlando, FL: Academic Press.

Schaal, S., Vijayakumar, S., & Atkeson, C. (1998). Local dimensionality reduction. In M. Jordan, M. Kearns, & S. Solla (Eds.), *Advances in neural information processing systems, 10.* Cambridge, MA: MIT Press.

Sergio, L., & Kalaska, J. (1998). Changes in the temporal pattern of primary motor cortex activity in a directional isometric force versus limb movement task. *Journal of Neurophysiology, 80*, 1577–1583.

Shevade, S., & Keerthi, S. (2003). A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics, 19*(17), 2246–2253.

Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of Royal Statistical Society, Series B, 58*(1), 267–288.

Ting, J., D'Souza, A., & Schaal, S. (2006). Bayesian regression with input noise for high dimensional data. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 937–944). New York: ACM.

Ting, J., D'Souza, A., Yamamoto, K., Yoshioka, T., Hoffman, D., Kakei, S., et al. (2005). Predicting EMG data from M1 neurons with variational Bayesian least squares. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in neural information processing systems, 18.* 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., & Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers. *Neural Computation, 11*(2), 443–482.

Tipping, M. E., & Faul, A. C. (2003). Fast marginal likelihood maximisation for sparse Bayesian methods. In C. M. Bishop & B. J. Frey (Eds.), *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics.* N.p.: Society for Artificial Intelligence and Statistics.

Vijayakumar, S., & Schaal, S. (2000). Locally weighted projection regression: Incremental real time learning in high dimensional space. In *Proceedings of the 17th International Conference on Machine Learning* (pp. 1079–1086). San Francisco: Morgan Kaufmann.

Wang, L., Zhu, J., & Zou, H. (2006). The doubly regularized support vector machine. *Statistica Sinica, 16*, 589–615.

Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems, 8*. Cambridge, MA: MIT Press.

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 honor of M. S. Bartlett.* London: Academic Press.

Received January 27, 2008; accepted August 3, 2009.