Fast Marginal Likelihood Maximisation for Sparse Bayesian Models

Michael E. Tipping and Anita C. Faul
Microsoft Research, Cambridge, U.K.

Published as:	"Fast marginal likelihood maximisation for sparse Bayesian models." In C. M. Bishop and B. J. Frey (Eds.), Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, Key West, FL, Jan 3–6.
Year of publication:	2003
This version typeset:	June 26, 2006
Available from:	`http://www.miketipping.com/papers.htm`
Correspondence:	`mail@miketipping.com`

Abstract The 'sparse Bayesian' modelling approach, as exemplified by the 'relevance vector machine', enables sparse classification and regression functions to be obtained by linearly-weighting a small number of fixed basis functions from a large dictionary of potential candidates. Such a model conveys a number of advantages over the related and very popular 'support vector machine', but the necessary 'training' procedure — optimisation of the marginal likelihood function — is typically much slower. We describe a new and highly accelerated algorithm which exploits recently-elucidated properties of the marginal likelihood function to enable maximisation via a principled and efficient sequential addition and deletion of candidate basis functions.

1 Introduction

It is an understatement to say that there has been considerable focus on 'sparse' models in machine learning in recent years. The 'support vector machine' (SVM) [2, 12], and other related kernel approaches, have generated great interest.

Typically, a sparse model would initially take the form:

$$y(\mathbf{x}) = \sum_{m=1}^{M} w_m \phi_m(\mathbf{x}), \tag{1}$$

where $y(\mathbf{x})$ may be an intended approximation to a real-valued function of interest (*i.e.* regression) or a discriminant function (*i.e.* classification). Although the model is linear in the parameters, it may still be highly flexible as the size of the basis set, M, may be very large. Given a sample of N corresponding 'training' pairs $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$, the objective is to find values for the weights $\mathbf{w} = (w_1, \ldots w_M)^{\mathrm{T}}$ such that $y(\mathbf{x})$ generalises well to new data yet only a few elements of $\mathbf{w}$ are non-zero.

The notion of setting weights to zero (as distinct from constraining them, say, to small values), is a compelling one in terms of controlling model complexity and avoiding over-fitting. A secondary feature is that inducing sparsity can be a highly effective method to control the computational characteristics of a resulting model. This can hopefully be exploited to an extent in the training procedure, but is typically much more valuable when the model is to be deployed in a practical

scenario where CPU and memory resources may be significantly limited. Relevant examples would be an implementation of a handwriting recogniser on a hand-held computer or of an "AI engine" in a video game.

In the SVM, the model is implicitly defined such that $M = N$ and $\phi_m(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}_m)$, with $K(\cdot, \cdot)$ a positive definite kernel function and $\mathbf{x}_m$ an example from the training set. (Typically, a 'bias' would also be used.) As well as this definitional constraint, there are a number of other well-documented limitations of the SVM paradigm, most of which, such as the absence of posterior class probability estimates in classification, stem from there being no associated probabilistic model.

To overcome this deficiency, some researchers have proposed modifications to the SVM framework itself [9, 5]. An alternative perspective is to consider the linear model (1) directly within a 'sparse Bayesian' framework, which leads to a model such as the 'relevance vector machine' (RVM) [10, 11]. This approach is summarised in Section 2, and is able to utilise more flexible candidate models, which are typically much sparser, offers probabilistic predictions and avoids the need to set additional regularisation parameters. However, one drawback of this approach is that the run time for the existing training algorithm scales approximately in the cube of the number of basis functions[1] preventing practical application of the algorithm to large data sets. Conversely, there now exist some very effective algorithms for efficiently training SVMs on large-scale problems [8, 4].

In this paper we present an accelerated training algorithm for sparse Bayesian models. Specifically, we exploit a recent result concerning the properties of the marginal likelihood function [3], discussed in Section 3, to derive a 'constructive' method for maximisation thereof. The original RVM algorithm [10] commenced with all the M basis functions included in the model, and updated hyperparameters iteratively. As a consequence of these updates, some basis functions would be 'pruned' and the algorithm would accelerate, but nevertheless the first few iterations would still require $O(M^3)$ computations. In the algorithm described in Section 4 we initialise with an 'empty' model, and sequentially 'add' basis functions to increase the marginal likelihood, and also modify their weightings. Within the same principled framework, we can also increase the objective function by 'deleting' functions which subsequently become redundant. This is an important feature, as it offsets the inherent 'greediness' that is exhibited by other sequential algorithms of this type.

We compared the performance of the method with both the original RVM algorithm, as well as with that of an SVM implementation, on some examples in Section 5, and summarise in Section 6. Some further implementation details for the algorithm are included in an appendix.

2 Sparse Bayesian Modelling

We describe the sparse Bayesian regression model first, and consider the adjustments required for classification later.

2.1 Regression

Given a data set $\{\mathbf{x}_n, t_n\}_{n=1}^N$ we write the targets as a vector $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$, and express it as the sum of an approximation vector $\mathbf{y} = (y(\mathbf{x}_1), \ldots, y(\mathbf{x}_N))^{\mathrm{T}}$ and an 'error' vector $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_N)^{\mathrm{T}}$:

$$\begin{aligned} \mathbf{t} &= \mathbf{y} + \boldsymbol{\epsilon}, \\ &= \boldsymbol{\Phi}\mathbf{w} + \boldsymbol{\epsilon}, \end{aligned} \tag{2}$$

[1] For the RVM, this is equivalent to the number of data points, as in the SVM. Unlike the SVM though, the sparse Bayesian framework permits use of arbitrarily sized sets of arbitrary basis functions, so computation can be constrained by explicitly reducing the basis size if so desired.

where $\mathbf{w}$ is the parameter vector and where $\mathbf{\Phi} = [\boldsymbol{\phi}_1 \dots \boldsymbol{\phi}_M]$ is the $N \times M$ 'design' matrix whose columns comprise the complete set of M 'basis vectors'.

The sparse Bayesian framework makes the conventional assumption that the errors are modelled probabilistically as independent zero-mean Gaussian, with variance σ^2: so $p(\boldsymbol{\epsilon}) = \prod_{n=1}^{N} N(\epsilon_n|0, \sigma^2)$. This is of course equivalent to a mean-squared-error criterion. The parameter σ^2 can be set in advance if known but more usually would be estimated from the data. This error model thus implies a multivariate Gaussian likelihood for the target vector $\mathbf{t}$:

$$p(\mathbf{t}|\mathbf{w}, \sigma^2) = (2\pi)^{-N/2}\sigma^{-N} \exp\left\{-\frac{\|\mathbf{t} - \mathbf{y}\|^2}{2\sigma^2}\right\}. \tag{3}$$

This likelihood function is complemented by a prior over the parameters which takes the form:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = (2\pi)^{-M/2} \prod_{m=1}^{M} \alpha_m^{1/2} \exp\left(-\frac{\alpha_m w_m^2}{2}\right). \tag{4}$$

Of note here is the introduction of M independent hyperparameters, $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_M)^{\mathrm{T}}$, each one individually controlling the strength of the prior over its associated weight. It is this form of prior that is ultimately responsible for the sparsity properties of the model (for more insight, see [11]). Given $\boldsymbol{\alpha}$, the *posterior* parameter distribution conditioned on the data is given by combining the likelihood and prior within Bayes' rule:

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})/p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2), \tag{5}$$

and is Gaussian $\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$ with

$$\mathbf{\Sigma} = (\mathbf{A} + \sigma^{-2}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1} \qquad \boldsymbol{\mu} = \sigma^{-2}\mathbf{\Sigma}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}, \tag{6}$$

and $\mathbf{A}$ defined as $\mathrm{diag}\,(\alpha_1, \dots, \alpha_M)$. Rather than extending the model to include Bayesian inference over those hyperparameters (which is analytically intractable), a most-probable point estimate $\boldsymbol{\alpha}_{\mathrm{MP}}$ may be found via a *type-II maximum likelihood* procedure. That is, sparse Bayesian 'learning' is formulated as the (local) maximisation with respect to $\boldsymbol{\alpha}$ of the *marginal likelihood*[2], or equivalently, its logarithm $\mathcal{L}(\boldsymbol{\alpha})$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) = \log p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) &= \log \int_{-\infty}^{\infty} p(\mathbf{t}|\mathbf{w}, \sigma^2)\; p(\mathbf{w}|\boldsymbol{\alpha})\; d\mathbf{w}, \\ &= -\frac{1}{2}\left[N \log 2\pi + \log|\mathbf{C}| + \mathbf{t}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{t}\right], \end{aligned} \tag{7}$$

with

$$\mathbf{C} = \sigma^2\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\mathrm{T}}. \tag{8}$$

A point estimate $\boldsymbol{\mu}_{\mathrm{MP}}$ for the parameters is then obtained by evaluating (6) with $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{\mathrm{MP}}$, giving a final (posterior mean) approximator $\mathbf{y} = \mathbf{\Phi}\boldsymbol{\mu}_{\mathrm{MP}}$. The crucial observation is that typically the optimal values of many hyperparameters are infinite [11]. From (6) this leads to a parameter posterior infinitely peaked at zero for many weights w_m with the consequence that $\boldsymbol{\mu}_{\mathrm{MP}}$ correspondingly comprises very few non-zero elements.

2.2 Classification

Sparse Bayesian classification follows an essentially identical framework as detailed for regression above, but using a Bernouilli likelihood and a sigmoidal link function to account for the change in the target quantities. As a consequence, there is an additional approximation step in the algorithm.

[2] This implicitly assumes a flat hyperprior over $\log(\boldsymbol{\alpha})$, which as well as being a practically convenient choice, is also pleasingly invariant to the scale of the target quantities. More general hyperpriors are considered in [11, 1].

Applying the logistic sigmoid link function $\sigma(y) = 1/(1+e^{-y})$ to $y(\mathbf{x})$ and, adopting the Bernoulli distribution for $P(t|\mathbf{x})$, we write the likelihood as:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma\{y(\mathbf{x}_n;\mathbf{w})\}^{t_n} \left[1-\sigma\{y(\mathbf{x}_n;\mathbf{w})\}\right]^{1-t_n}, \tag{9}$$

where, following from the probabilistic specification, the targets $t_n \in \{0,1\}$.

Unlike the regression case, the weights cannot be integrated out analytically, precluding closed-form expressions for either the weight posterior $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$ or the marginal likelihood $P(\mathbf{t}|\boldsymbol{\alpha})$. We thus utilise the Laplace approximation procedure, as used in [6]:

1. For the current values of $\boldsymbol{\alpha}$, the the mode of the posterior distribution is found iteratively to give the 'most probable' weights $\boldsymbol{\mu}_{\mathrm{MP}}$.

   Since $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha}) \propto P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})$, this is equivalent to finding the maximum, over $\mathbf{w}$, of

   $$\log\{P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})\} = \sum_{n=1}^{N}[t_n \log y_n + (1-t_n)\log(1-y_n)] - \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{A}\mathbf{w}, \tag{10}$$

   with $y_n = \sigma\{y(\mathbf{x}_n;\mathbf{w})\}$. This is a standard procedure, since (10) is a penalised logistic log-likelihood function, and necessitates iterative maximisation. We used a second-order Newton method by adapting the efficient 'iteratively-reweighted least-squares' algorithm (*e.g.* [7]) to find $\boldsymbol{\mu}_{\mathrm{MP}}$.

2. Laplace's method is simply a quadratic approximation to the log-posterior around its mode. The quantity (10) is differentiated twice to give:

   $$\nabla_{\mathbf{w}}\nabla_{\mathbf{w}} \log p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})\big|_{\boldsymbol{\mu}_{\mathrm{MP}}} = -(\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A}), \tag{11}$$

   where $\mathbf{B} = \mathrm{diag}\,(\beta_1,\beta_2,\ldots,\beta_N)$ is a diagonal matrix with $\beta_n = \sigma\{y(\mathbf{x}_n)\}\left[1-\sigma\{y(\mathbf{x}_n)\}\right]$. This is then negated and inverted to give the covariance $\boldsymbol{\Sigma}$ for a Gaussian approximation to the posterior over weights centred at $\boldsymbol{\mu}_{\mathrm{MP}}$.

At the mode of $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$, using (11) and the fact that $\nabla_{\mathbf{w}} \log p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})\big|_{\boldsymbol{\mu}_{\mathrm{MP}}} = 0$, we can see we have effectively locally 'linearised' the classification problem around $\boldsymbol{\mu}_{\mathrm{MP}}$ with

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A})^{-1}, \tag{12}$$

$$\boldsymbol{\mu}_{\mathrm{MP}} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\widehat{\mathbf{t}}, \tag{13}$$

and

$$\widehat{\mathbf{t}} = \boldsymbol{\Phi}\boldsymbol{\mu}_{\mathrm{MP}} + \mathbf{B}^{-1}(\mathbf{t}-\mathbf{y}). \tag{14}$$

These equations are equivalent to the solution to a 'generalised least squares' problem. Compared with (6), it can be seen that the Laplace approximation effectively maps the classification problem to a regression one with 'targets' $\widehat{\mathbf{t}}$ and data-dependent (heteroscedastic) noise, with the inverse noise variance for ϵ_n given by $\beta_n = \sigma\{y(\mathbf{x}_n)\}\left[1-\sigma\{y(\mathbf{x}_n)\}\right]$. It will be convenient to utilise this linearised form in the classification variant of the newly-proposed algorithm.

3 Marginal Likelihood Maximisation

The algorithm proposed in this paper is a particular strategy for maximisation of the marginal likelihood (7), the effectiveness of which is dependent on the properties thereof. We summarise some key properties here, and greater detail is given in [3].

Considering the dependence of $\mathcal{L}(\boldsymbol{\alpha})$ on a single hyperparameter α_i, $i \in \{1 \dots M\}$, we can decompose $\mathbf{C}$ in (8) as

$$\begin{aligned}\mathbf{C} &= \sigma^2\mathbf{I} + \sum_{m\neq i} \alpha_m^{-1}\boldsymbol{\phi}_m\boldsymbol{\phi}_m^{\mathrm{T}} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}},\\ &= \mathbf{C}_{-i} + \alpha_i^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}},\end{aligned} \tag{15}$$

where $\mathbf{C}_{-i}$ is $\mathbf{C}$ with the contribution of basis vector i removed. Established matrix determinant and inverse identities may be used to write the terms of interest in $\mathcal{L}(\boldsymbol{\alpha})$ as:

$$|\mathbf{C}| = |\mathbf{C}_{-i}|\,|1 + \alpha_i^{-1}\boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i|, \tag{16}$$

$$\mathbf{C}^{-1} = \mathbf{C}_{-i}^{-1} - \frac{\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}}{\alpha_i + \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i}. \tag{17}$$

From this, we can write $\mathcal{L}(\boldsymbol{\alpha})$ as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) &= -\frac{1}{2}\Big[N\log(2\pi) + \log|\mathbf{C}_{-i}| + \mathbf{t}^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\mathbf{t}\\ &\qquad - \log\alpha_i + \log(\alpha_i + \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i) - \frac{(\boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\mathbf{t})^2}{\alpha_i + \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i}\Big],\\ &= \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \frac{1}{2}\left[\log\alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right]\\ &= \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \ell(\alpha_i),\end{aligned} \tag{18}$$

where for simplification of forthcoming expressions, we have defined:

$$s_i \triangleq \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\boldsymbol{\phi}_i, \quad \text{and} \quad q_i \triangleq \boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{C}_{-i}^{-1}\mathbf{t}. \tag{19}$$

The 'sparsity factor' s_i can be seen to be a measure of the extent that basis vector $\boldsymbol{\phi}_i$ 'overlaps' those already present in the model. The 'quality factor' can be written as $q_i = \sigma^{-2}\boldsymbol{\phi}_i^{\mathrm{T}}(\mathbf{t} - \mathbf{y}_{-i})$, and is thus a measure of the alignment of $\boldsymbol{\phi}_i$ with the error of the model with that vector excluded.

The objective function has now been decomposed into $\mathcal{L}(\boldsymbol{\alpha}_{-i})$, the marginal likelihood with $\boldsymbol{\phi}_i$ excluded, and $\ell(\alpha_i)$, where terms in α_i are now conveniently isolated.

Analysis of $\ell(\alpha_i)$ [3] shows that $\mathcal{L}(\boldsymbol{\alpha})$ has a unique maximum with respect to α_i:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}, \qquad \text{if } q_i^2 > s_i, \tag{20}$$

$$\alpha_i = \infty, \qquad \text{if } q_i^2 \leq s_i. \tag{21}$$

An example illustrating these two cases is given in Figure 1.

It is relatively straightforward to compute q_i and s_i for all the basis functions $\boldsymbol{\phi}_i$ in the dictionary, including those not currently utilised in the model (*i.e.* for which $\alpha_i = \infty$). In fact, it is easier to maintain and update values of

$$S_m = \boldsymbol{\phi}_m^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{\phi}_m, \qquad Q_m = \boldsymbol{\phi}_m^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{t}, \tag{22}$$

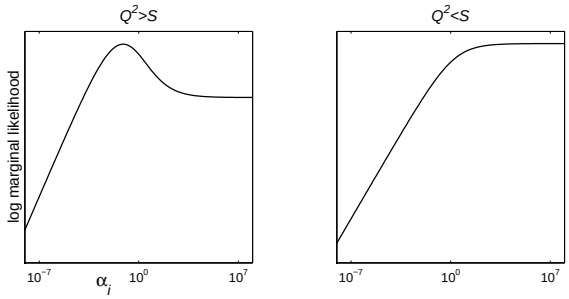

Figure 1: Example plots of $\ell(\alpha_i)$ against α_i (on a log scale) for $q^2 > s$ (left), showing the single maximum at finite α_i, and $q^2 < s$ (right), showing the maximum as $\alpha_i \to \infty$.

and from these it follows simply:

$$s_m = \frac{\alpha_m S_m}{\alpha_m - S_m}, \qquad q_m = \frac{\alpha_m Q_m}{\alpha_m - S_m}. \tag{23}$$

Note that when $\alpha_m = \infty$, $s_m = S_m$ and $q_m = Q_m$. In practice, then, it is convenient to utilise the Woodbury identity to obtain the quantities of interest:

$$S_m = \boldsymbol{\phi}_m^{\mathrm{T}} \mathbf{B} \boldsymbol{\phi}_m - \boldsymbol{\phi}_m^{\mathrm{T}} \mathbf{B} \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{B} \boldsymbol{\phi}_m, \tag{24}$$

$$Q_m = \boldsymbol{\phi}_m^{\mathrm{T}} \mathbf{B} \widehat{\mathbf{t}} - \boldsymbol{\phi}_m^{\mathrm{T}} \mathbf{B} \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{B} \widehat{\mathbf{t}}, \tag{25}$$

where $\mathbf{B} \equiv \sigma^{-2}\mathbf{I}$ and $\widehat{\mathbf{t}} \equiv \mathbf{t}$ in the regression case, and for the classification case as defined by (11) and (14) in Section 2.2. Here quantities $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$ contain only those basis functions that are currently included in the model, and computation thus scales in the cube of that measure which is typically only a very small fraction of the full M. Furthermore, if σ^2 is fixed in regression, these quantities can all be calculated via 'update' formulae, given in the Appendix, with reduced computation.

The results (20) and (21) imply that:

- If $\boldsymbol{\phi}_i$ is 'in the model' (*i.e.* $\alpha_i < \infty$) yet $q_i^2 \leq s_i$, then $\boldsymbol{\phi}_i$ may be deleted (*i.e.* α_i set to ∞),
- If $\boldsymbol{\phi}_i$ is excluded from the model ($\alpha_i = \infty$) and $q_i^2 > s_i$, $\boldsymbol{\phi}_i$ may be added (*i.e.* α_i is set to some optimal finite value).

In both these cases we are able to make discrete changes to the model structure while at the same time we are guaranteed to increase the marginal likelihood objective function. Additionally, we can re-estimate α_i if (20) applies for basis vectors already in the model. If we perform these operations sequentially for varying i, we can realise an efficient learning algorithm which we detail in the next section.

4 A Sequential Sparse Bayesian Learning Algorithm

The proposed marginal likelihood maximisation algorithm is as follows:

1. If *regression* initialise σ^2 to some sensible value (*e.g.* $\text{var}[t] \times 0.1$).
2. Initialise with a single basis vector $\boldsymbol{\phi}_i$, setting, from (20):
$$\alpha_i = \frac{\|\boldsymbol{\phi}_i\|^2}{\|\boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{t}\|^2/\|\boldsymbol{\phi}_i\|^2 - \sigma^2}. \tag{26}$$
All other α_m are notionally set to infinity.
3. Explicitly compute $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ (which are scalars initially), along with initial values of s_m and q_m for all M bases $\boldsymbol{\phi}_m$.
4. Select a candidate basis vector $\boldsymbol{\phi}_i$ from the set of all M.
5. Compute $\theta_i \triangleq q_i^2 - s_i$.
6. If $\theta_i > 0$ and $\alpha_i < \infty$ (*i.e.* $\boldsymbol{\phi}_i$ is in the model), **re-estimate** α_i.
7. If $\theta_i > 0$ and $\alpha_i = \infty$, **add** $\boldsymbol{\phi}_i$ to the model with updated α_i.
8. If $\theta_i \leq 0$ and $\alpha_i < \infty$, then **delete** $\boldsymbol{\phi}_i$ from the model and set $\alpha_i = \infty$.
9. If *regression* and estimating the noise level, update $\sigma^2 = \|\mathbf{t} - \mathbf{y}\|^2/(N - M + \sum_m \alpha_m \boldsymbol{\Sigma}_{mm})$ [11].
10. Recompute/update $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$ (using the Laplace approximation procedure in *classification*) and all s_m and q_m using equations (23)–(25).
11. If converged terminate, otherwise goto 4.

4.1 Initialisation

A potential basis vector for initialisation in step 2 could be the bias, *i.e.* $\boldsymbol{\phi}_i = (1, 1, \ldots, 1)^{\mathrm{T}}$, if included. A sensible alternative would be that vector with the largest normalised projection onto the target vector, $\|\boldsymbol{\phi}_i^{\mathrm{T}}\mathbf{t}\|^2/\|\boldsymbol{\phi}_i\|^2$, which can also be seen to give the largest initial likelihood.

4.2 Selection

In step 4, we must select a candidate basis function for updating. This should be from the set of all functions, both included in and excluded from the current model. Candidates could be selected (even generated) purely at random, or from an ordered list in sequence. Alternatively, for proportionately little additional computational expense, values of θ_i and updated α_i can be calculated for *all* bases, and the change in marginal likelihood computed for each potential update (see Appendix). That giving the greatest increase can then be implemented.

4.3 Updating statistics

In step 10, we desire to recompute $\mathbf{\Sigma}$, $\boldsymbol{\mu}$, and all quantities s_m and q_m. If the noise variance σ^2 is fixed, this can be done efficiently using 'update' formulae which are detailed in the Appendix. If σ^2 has been updated in a regression iteration, the necessary quantities are recomputed in full from (24) and (25). In classification, the Laplace approximation must be re-fitted, and the linearised quantities used in the full recomputations.

4.4 Convergence

In step 11, we must judge if we have attained a local maximum of the marginal likelihood. We terminate when the changes in $\log \alpha$ in Step 6 for all basis functions in the model are smaller than 10^{-6} and all other $\theta_i \leq 0$.

We would re-iterate here that the above algorithm is guaranteed to increase the marginal likelihood at each step, until a local maximum is attained. Indeed, although superficially we appear to be 'adding' and 'deleting' basis functions, we can consider that notionally we are maintaining posterior statistics for *all* basis vectors concurrently (all elements of $\mathbf{\Sigma}$ and $\boldsymbol{\mu}$ corresponding to 'out of model' basis vectors are trivially zero).

5 Simulations

In this section we offer some running-time comparisons between the previous sparse Bayesian learning algorithm and that proposed in this paper. We utilise RVM (*i.e.* $M = N$ plus bias) models so that, for reasons of illustration rather than critical comparison, we can compare with a state-of-the-art SVM implementation, SVM^{light}[4]. In addition, we also compare the sparsity and generalisation errors of the old and new sparse Bayesian algorithms[3].

5.1 Regression

For this comparison, we synthetically generated data from the two-dimensional 'sinc' function $y(\mathbf{x}) = \sin(\|\mathbf{x}\|)/\|\mathbf{x}\|$, with added noise of standard deviation 0.1. A Gaussian kernel function of 'width' 2.5 was utilised. In the RVM case, we also re-estimated the noise variance one in every five iterations.

The experiment was run with varying values of N, from 100 to 4000, such that at the upper limit, the design matrix was still contained in memory. The 'old' RVM algorithm was only run up to $N = 1000$. Figure 2 shows the resulting run times[4], averaged over 10 random generations of the data set.

[3]Note that we choose not to compare errors between RVM and SVM here, as our focus is on the comparison between new and old sparse Bayesian techniques. Any error results for the SVM would be very much dependent on the strategy adopted for optimising its control parameters C and ϵ, which makes deterministic comparison difficult, and both models' performance depends on the choice of kernel parameters too. However, a general observation for both sets of data presented here is that the RVM does give noticably lower error, which evidence suggests is typical for regression but atypical for classification [11].

[4]Quoted values were obtained on an AMD Athlon XP 2000+ PC under Windows XP.

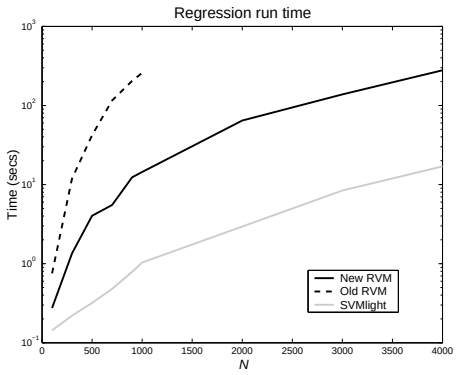

Figure 2: Regression run time for the old and new sparse Bayesian learning algorithms, along with that of the equivalent SVM^{light}.

Comparing at $N = 1000$ we have:

Old RVM	4 mins 17 secs
New RVM	14.42 secs
SVM^{light}	1.03 secs

There is a clear and dramatic improvement over the existing sparse Bayesian algorithm, though SVM^{light}remains significantly faster.

A comparison of generalisation error (from an independent test set) and sparsity for the two Bayesian regression algorithms is given in figure 3. In terms of error, the two approaches appear comparable, although the new algorithm appears systematically sparser to a small degree.

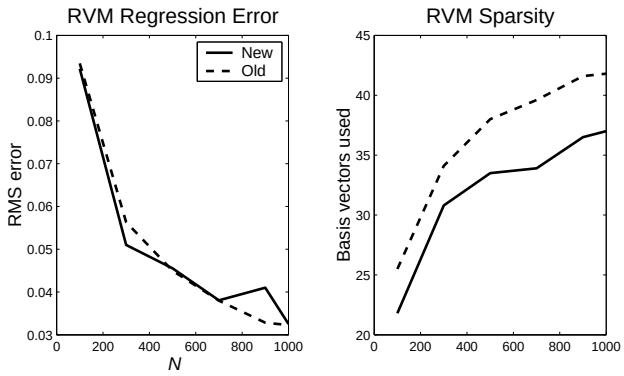

Figure 3: RMS errors and final model sparsity for the old and new sparse Bayesian learning algorithms.

5.2 Classification

For this comparison, we again synthetically generated data from a two-dimensional problem, this time from two classes with mixtures of Gaussian conditional distributions, as used and illustrated in [10]. A Gaussian kernel function of 'width' 1.0 was utilised.

The experiment was again run with varying values of N, from 100 to 4000. Again, the 'old' RVM algorithm was only run up to $N = 1000$. Figure 4 shows the resulting run times, averaged over 10 random generations of the data set.

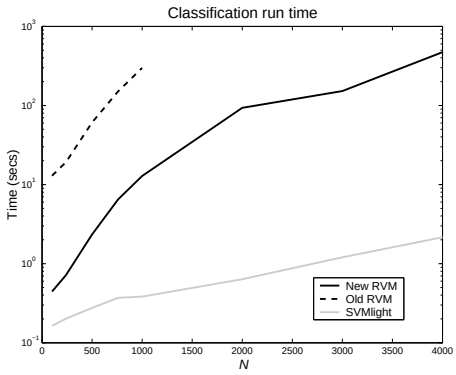

Figure 4: Classification run time for the old and new sparse Bayesian learning algorithms, along with that of the equivalent SVM^{light}.

Comparing at $N = 1000$ we have:

Old RVM	4 mins 58 secs
New RVM	12.84 secs
SVM^{light}	0.38 secs

A comparison of generalisation error and sparsity for the two sparse Bayesian algorithms is given in figure 5.

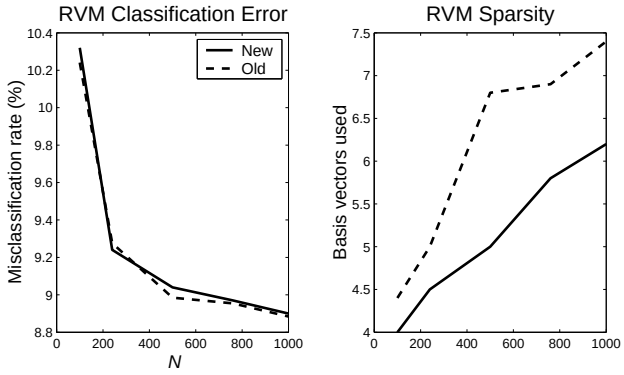

Figure 5: Classification errors and final model sparsity for the old and new sparse Bayesian learning algorithms.

Similarly to the regression case, in terms of speed the new algorithm is much faster than the old, while appearing comparable in terms of error and slightly sparser in utilisation of basis functions.

6 Summary

6.1 Efficacy of the sequential algorithm

Based on the evidence of the experiments in Section 5, the new marginal likelihood maximisation algorithm appears to operate highly effectively. It offers a very clear speed advantage over the originally proposed approach. Given that the sparse Bayesian framework arguably offers a num-

ber of advantageous features when compared with the popular SVM, it is salient that its main disadvantage, that of grossly extended training time for large data sets, can be largely overcome.

That said, sparse Bayesian models still require significantly greater training time than SVM^{light}, which operates at very impressive speed. The comparison with SVM^{light} is intended to be illustrative only, but given the differential between RVM and SVM training times, it is fair to make the following points:

- SVM^{light} is a refined C implementation of the SVM, while our experiments were run using 'prototype'-level *Matlab* code. Some room for improvement in the latter is to be expected.
- The SVM timings understate the 'real' time requirement of fitting the model as in practice multiple runs must be undertaken to validate a value for the error/margin trade-off parameter ('C'), and in regression, the error tolerance ('ϵ') too. Such validation is not necessary in the sparse Bayesian case.

The new algorithm also conveys an advantage over the old in that deletion of basis functions (*i.e.* setting α to infinity) now occurs analytically. In [10], basis functions were pruned when individual iterated α values grew numerically too large. The increased numerical precision of the deletion operation is the likely explanation for the improvement in sparsity demonstrated by the new algorithm.

In a sequential algorithm such as presented here, there is always a concern that the procedure will be 'greedy' and converge on a sub-optimal model. The experimental evidence of generalisation error (which is only weakly an indicator of the 'greediness' of the procedure, but the one of most interest) indicates that little if any cost in solution 'quality' is being paid, on the simple data studied here at least.

6.2 Directions for further investigation

- There is a potential modification to the algorithm to reduce its greediness. As well as addition, deletion and updating, it is possible to consider *exchanging* of basis vectors — *i.e.* simultaneous deletion and addition. This permits the deletion of a basis function which would initially lower the marginal likelihood, which is then more than compensated for by an additional vector. This requires only little further computation in practice. A first implementation shows this to work effectively, but we have encountered initial numerical difficulties with this feature which we are currently working to overcome.
- An alternative suggestion is to propose 'split' and 'merge' operations, where, for example, two basis functions could be combined and replaced with a single such function located at their respective midpoint. The effect of such modifications on the marginal likelihood could be assessed relatively efficiently given the framework presented here.
- One final notable issue we have not considered here is application to larger data sets where the design matrix will not fit in memory and where the original algorithm cannot be used. The presented sequential algorithm can be applied without modification in this case, although there is a large additional cost associated with the extra repeated re-evaluations of the basis functions. For practical efficiency, we have therefore adopted an approach where we apply the algorithm in full to *subsets* of the basis functions in sequence, and terminate when all have simultaneously converged (which still maximises the correct marginal likelihood measure). We are currently finalising the exact mechanism of this technique, but have already obtained successful results on as many as one million data points.

Acknowledgments

The authors would like to thank the reviewers for helpful comments and suggestions.

References

[1] C. M. Bishop and M. E. Tipping. Variational relevance vector machines. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 46–53. Morgan Kaufmann, 2000.

[2] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[3] A. C. Faul and M. E. Tipping. Analysis of sparse Bayesian learning. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 383–389. MIT Press, 2002.

[4] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. MIT Press, 1999.

[5] J. T.-K. Kwok. The evidence framework applied to support vector machines. *IEEE Transactions on Neural Networks*, 11(5):1162–1173, 2000.

[6] D. J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736, 1992.

[7] I. T. Nabney. Efficient training of RBF networks for classification. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN99)*, pages 210–215. IEE, 1999.

[8] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[9] P. Sollich. Probabilistic methods for support vector machines. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 349–355. MIT Press, 2000.

[10] M. E. Tipping. The Relevance Vector Machine. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658. MIT Press, 2000.

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

[12] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

A Efficient calculations for some quantities of interest

In step 4 of the algorithm in Section 4 we may wish to calculate the increase or decrease of the marginal likelihood $\mathcal{L}(\boldsymbol{\alpha})$ according to which basis function is added, deleted or re-estimated. Efficient expressions for the likelihood changes are given below.

Also, in step 10, it is necessary to recompute $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$, and all quantities s_m and q_m. If σ^2 is fixed (perhaps only for the iteration in question), then in all cases update formulae can be utilised for efficiency and accuracy; no quantities need be calculated *ab initio*.

A.1 Notation

The size of the basis at step t of the algorithm is denoted as M_t. For efficiency, $\boldsymbol{\Phi}$ as used below need only comprise columns of included basis functions, and so is $N \times M_t$. Similarly $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ likewise are computed only for the 'current' basis and so are of order M_t (all other entries in the 'full' version of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ would

be zero). The integer $i \in \{1 \dots M\}$ is used to index the single basis function for which α_i is to be updated, and the integer $j \in \{1 \dots M_t\}$ to denote the index within the current basis that corresponds to i. The index m ranges over all basis functions. For convenience, $\beta \triangleq \sigma^{-2}$. Updated quantities are denoted by a tilde (*e.g.* $\widetilde{\alpha}$).

A.2 Adding a new basis function

$$2\Delta\mathcal{L} = \frac{Q_i^2 - S_i}{S_i} + \log\frac{S_i}{Q_i^2}, \tag{27}$$

$$\widetilde{\boldsymbol{\Sigma}} = \begin{bmatrix} \boldsymbol{\Sigma} + \beta^2\boldsymbol{\Sigma}_{ii}\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Sigma} & -\beta^2\boldsymbol{\Sigma}_{ii}\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_i \\ -\beta^2\boldsymbol{\Sigma}_{ii}(\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_i)^{\mathrm{T}} & \boldsymbol{\Sigma}_{ii} \end{bmatrix}, \tag{28}$$

$$\widetilde{\boldsymbol{\mu}} = \begin{bmatrix} \boldsymbol{\mu} - \mu_i\beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_i \\ \mu_i \end{bmatrix}, \tag{29}$$

$$\widetilde{S}_m = S_m - \boldsymbol{\Sigma}_{ii}(\beta\boldsymbol{\phi}_m^{\mathrm{T}}\mathbf{e}_i)^2, \tag{30}$$

$$\widetilde{Q}_m = Q_m - \mu_i(\beta\boldsymbol{\phi}_m^{\mathrm{T}}\mathbf{e}_i), \tag{31}$$

where $\boldsymbol{\Sigma}_{ii} = (\alpha_i + S_i)^{-1}$, $\mu_i = \boldsymbol{\Sigma}_{ii}Q_i$ and we define $\mathbf{e}_i \triangleq \boldsymbol{\phi}_i - \beta\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_i$.

A.3 Re-estimating a basis function

Defining $\kappa_j \triangleq (\boldsymbol{\Sigma}_{jj} + (\widetilde{\alpha}_i - \alpha_i)^{-1})^{-1}$ and $\boldsymbol{\Sigma}_j$ as the j-th column of $\boldsymbol{\Sigma}$:

$$2\Delta\mathcal{L} = \frac{Q_i^2}{S_i + [\widetilde{\alpha}_i^{-1} - \alpha_i^{-1}]^{-1}} - \log\left\{1 + S_i[\widetilde{\alpha}_i^{-1} - \alpha_i^{-1}]\right\}, \tag{32}$$

$$\widetilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} - \kappa_j\boldsymbol{\Sigma}_j\boldsymbol{\Sigma}_j^{\mathrm{T}}, \tag{33}$$

$$\widetilde{\boldsymbol{\mu}} = \boldsymbol{\mu} - \kappa_j\mu_j\boldsymbol{\Sigma}_j, \tag{34}$$

$$\widetilde{S}_m = S_m + \kappa_j(\beta\boldsymbol{\Sigma}_j^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_m)^2, \tag{35}$$

$$\widetilde{Q}_m = Q_m + \kappa_j\mu_j(\beta\boldsymbol{\Sigma}_j^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_m). \tag{36}$$

A.4 Deleting a basis function

$$2\Delta\mathcal{L} = \frac{Q_i^2}{S_i - \alpha_i} - \log\left(1 - \frac{S_i}{\alpha_i}\right), \tag{37}$$

$$\widetilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} - \frac{1}{\boldsymbol{\Sigma}_{jj}}\boldsymbol{\Sigma}_j\boldsymbol{\Sigma}_j^{\mathrm{T}}, \tag{38}$$

$$\widetilde{\boldsymbol{\mu}} = \boldsymbol{\mu} - \frac{\mu_j}{\boldsymbol{\Sigma}_{jj}}\boldsymbol{\Sigma}_j, \tag{39}$$

$$\widetilde{S}_m = S_m + \frac{1}{\boldsymbol{\Sigma}_{jj}}(\beta\boldsymbol{\Sigma}_j^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_m)^2, \tag{40}$$

$$\widetilde{Q}_m = Q_m + \frac{\mu_j}{\boldsymbol{\Sigma}_{jj}}(\beta\boldsymbol{\Sigma}_j^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\phi}_m). \tag{41}$$

Following updates (38) and (39), the appropriate row and/or column j is removed from $\widetilde{\boldsymbol{\Sigma}}$ and $\widetilde{\boldsymbol{\mu}}$.