Journal of Machine Learning Research 1 (2001) 211–244 Submitted 5/00; Published 6/01

Sparse Bayesian Learning and the Relevance Vector Machine

Michael E. Tipping MTIPPING@MICROSOFT.COM
Microsoft Research
St George House, 1 Guildhall Street
Cambridge CB2 3NH, U.K.

Editor: Alex Smola

Abstract

This paper introduces a general Bayesian framework for obtaining *sparse* solutions to regression and classification tasks utilising models linear in the parameters. Although this framework is fully general, we illustrate our approach with a particular specialisation that we denote the 'relevance vector machine' (RVM), a model of identical functional form to the popular and state-of-the-art 'support vector machine' (SVM). We demonstrate that by exploiting a probabilistic Bayesian learning framework, we can derive accurate prediction models which typically utilise dramatically fewer basis functions than a comparable SVM while offering a number of additional advantages. These include the benefits of probabilistic predictions, automatic estimation of 'nuisance' parameters, and the facility to utilise arbitrary basis functions (*e.g.* non-'Mercer' kernels).

We detail the Bayesian framework and associated learning algorithm for the RVM, and give some illustrative examples of its application along with some comparative benchmarks. We offer some explanation for the exceptional degree of sparsity obtained, and discuss and demonstrate some of the advantageous features, and potential extensions, of Bayesian relevance learning.

1. Introduction

In *supervised learning* we are given a set of examples of input vectors $\{\mathbf{x}_n\}_{n=1}^N$ along with corresponding targets $\{t_n\}_{n=1}^N$, the latter of which might be real values (in *regression*) or class labels (*classification*). From this 'training' set we wish to learn a model of the dependency of the targets on the inputs with the objective of making accurate predictions of t for previously unseen values of $\mathbf{x}$. In real-world data, the presence of noise (in regression) and class overlap (in classification) implies that the principal modelling challenge is to avoid 'over-fitting' of the training set.

Typically, we base our predictions upon some function $y(\mathbf{x})$ defined over the input space, and 'learning' is the process of inferring (perhaps the parameters of) this function. A flexible and popular set of candidates for $y(\mathbf{x})$ is that of the form:

$$y(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}), \tag{1}$$

where the output is a linearly-weighted sum of M, generally nonlinear and fixed, basis functions $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))^{\mathrm{T}}$. Analysis of functions of the type (1) is facilitated since the adjustable parameters (or 'weights') $\mathbf{w} = (w_1, w_2, \ldots, w_M)^{\mathrm{T}}$ appear linearly, and the objective is to estimate 'good' values for those parameters.

In this paper, we detail a Bayesian probabilistic framework for learning in general models of the form (1). The key feature of this approach is that as well as offering good generalisation performance, the inferred predictors are exceedingly *sparse* in that they contain relatively few non-zero w_i parameters. The majority of parameters are automatically set to zero during the learning process, giving a procedure that is extremely effective at discerning those basis functions which are 'relevant' for making good predictions.

While the range of models of the type (1) that we can address is extremely broad, we concentrate here on a specialisation that we denote the 'relevance vector machine' (RVM), originally introduced by Tipping (2000). We consider functions of a type corresponding to those implemented by another sparse linearly-parameterised model, the *support vector machine* (SVM) (Boser et al., 1992; Vapnik, 1998; Schölkopf et al., 1999a). The SVM makes predictions based on the function:

$$y(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x}, \mathbf{x}_i) + w_0, \tag{2}$$

where $K(\mathbf{x}, \mathbf{x}_i)$ is a *kernel* function, effectively defining one basis function for each example in the training set.[1] The key feature of the SVM is that, in the classification case, its target function attempts to minimise a measure of error on the training set while simultaneously maximising the 'margin' between the two classes (in the feature space implicitly defined by the kernel). This is a highly effective mechanism for avoiding over-fitting, which leads to good generalisation, and which furthermore results in a sparse model dependent only on a subset of kernel functions: those associated with training examples $\mathbf{x}_n$ (the "support vectors") that lie either on the margin or on the 'wrong' side of it. State-of-the-art results have been reported on many tasks where the SVM has been applied.

However, despite its success, we can identify a number of significant and practical disadvantages of the support vector learning methodology:

- Although relatively sparse, SVMs make unnecessarily liberal use of basis functions since the number of support vectors required typically grows linearly with the size of the training set. Some form of post-processing is often required to reduce computational complexity (Burges, 1996; Burges and Schölkopf, 1997).

- Predictions are not *probabilistic*. In regression the SVM outputs a point estimate, and in classification, a 'hard' binary decision. Ideally, we desire to estimate the conditional distribution $p(t|\mathbf{x})$ in order to capture uncertainty in our prediction. In regression this may take the form of 'error-bars', but it is particularly crucial in classification where posterior probabilities of class membership are necessary to adapt to varying class priors and asymmetric misclassification costs. Posterior probability estimates have been coerced from SVMs via post-processing (Platt, 2000), although we argue that these estimates are unreliable (Appendix D.2).

1. Note that the SVM predictor is not defined explicitly in this form — rather (2) emerges implicitly as a consequence of the use of the kernel function to define a dot-product in some notional feature space.

- It is necessary to estimate the error/margin trade-off parameter 'C' (and in regression, the insensitivity parameter 'ϵ' too). This generally entails a cross-validation procedure, which is wasteful both of data and computation.
- The kernel function $K(\mathbf{x}, \mathbf{x}_i)$ must satisfy Mercer's condition. That is, it must be the continuous symmetric kernel of a positive integral operator.[2]

The 'relevance vector machine' (RVM) is a Bayesian treatment[3] of (2) which does not suffer from any of the above limitations. Specifically, we adopt a fully probabilistic framework and introduce a prior over the model weights governed by a set of hyperparameters, one associated with each weight, whose most probable values are iteratively estimated from the data. Sparsity is achieved because in practice we find that the posterior distributions of many of the weights are sharply (indeed infinitely) peaked around zero. We term those training vectors associated with the remaining non-zero weights 'relevance' vectors, in deference to the principle of *automatic relevance determination* which motivates the presented approach (MacKay, 1994; Neal, 1996). The most compelling feature of the RVM is that, while capable of generalisation performance comparable to an equivalent SVM, it typically utilises dramatically fewer kernel functions.

In the next section, we introduce the Bayesian model, initially for regression, and define the procedure for obtaining hyperparameter values, and from them, the weights. The framework is then extended straightforwardly to the classification case in Section 3. In Section 4, we give some visualisable examples of application of the RVM in both scenarios, along with an illustration of some potentially powerful extensions to the basic model, before offering some benchmark comparisons with the SVM. We offer some theoretical insight into the reasons behind the observed sparsity of the technique in Section 5 before summarising in Section 6. To streamline the presentation within the main text, considerable theoretical and implementational details are reserved for the appendices.

2. Sparse Bayesian Learning for Regression

We now detail the sparse Bayesian regression model and associated inference procedures. The classification counterpart is considered in Section 3.

2.1 Model Specification

Given a data set of input-target pairs $\{\mathbf{x}_n, t_n\}_{n=1}^N$, considering scalar-valued target functions only, we follow the standard probabilistic formulation and assume that the targets are samples from the model with additive noise:

$$t_n = y(\mathbf{x}_n; \mathbf{w}) + \epsilon_n,$$

where ϵ_n are independent samples from some noise process which is further assumed to be mean-zero Gaussian with variance σ^2. Thus $p(t_n|\mathbf{x}) = \mathcal{N}(t_n|y(\mathbf{x}_n), \sigma^2)$, where the notation

2. This restriction can be relaxed slightly to include *conditionally* positive kernels (Smola et al., 1998; Schölkopf, 2001).

3. Note that our approach is not a Bayesian treatment of the SVM methodology *per se*, an area which has seen much recent interest (Sollich, 2000; Seeger, 2000; Kwok, 2000) — here we treat the kernel function as simply defining a set of basis functions, rather than as a definition of a dot-product in some space.

specifies a Gaussian distribution over t_n with mean $y(\mathbf{x}_n)$ and variance σ^2. The function $y(\mathbf{x})$ is as defined in (2) for the SVM where we identify our general basis functions with the kernel as parameterised by the training vectors: $\phi_i(\mathbf{x}) \equiv K(\mathbf{x}, \mathbf{x}_i)$. Due to the assumption of independence of the t_n, the likelihood of the complete data set can be written as

$$p(\mathbf{t}|\mathbf{w}, \sigma^2) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{t} - \mathbf{\Phi}\mathbf{w}\|^2\right\}, \tag{4}$$

where $\mathbf{t} = (t_1 \ldots t_N)^{\mathrm{T}}$, $\mathbf{w} = (w_0 \ldots w_N)^{\mathrm{T}}$ and $\mathbf{\Phi}$ is the $N \times (N+1)$ 'design' matrix with $\mathbf{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)]^{\mathrm{T}}$, wherein $\boldsymbol{\phi}(\mathbf{x}_n) = [1, K(\mathbf{x}_n, \mathbf{x}_1), K(\mathbf{x}_n, \mathbf{x}_2), \ldots, K(\mathbf{x}_n, \mathbf{x}_N)]^{\mathrm{T}}$. For clarity, we omit to notate the implicit conditioning upon the set of input vectors $\{\mathbf{x}_n\}$ in (4) and subsequent expressions.

With as many parameters in the model as training examples, we would expect maximum-likelihood estimation of $\mathbf{w}$ and σ^2 from (4) to lead to severe over-fitting. To avoid this, a common approach is to impose some additional constraint on the parameters, for example, through the addition of a 'complexity' penalty term to the likelihood or error function. This is implicitly effected by the inclusion of the 'margin' term in the SVM. Here, though, we adopt a Bayesian perspective, and 'constrain' the parameters by defining an explicit *prior* probability distribution over them.

We encode a preference for smoother (less complex) functions by making the popular choice of a zero-mean Gaussian prior distribution over $\mathbf{w}$:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1}), \tag{5}$$

with $\boldsymbol{\alpha}$ a vector of $N+1$ *hyperparameters*. Importantly, there is an individual hyperparameter associated independently with every weight, moderating the strength of the prior thereon.[4]

To complete the specification of this *hierarchical* prior, we must define hyperpriors over $\boldsymbol{\alpha}$, as well as over the final remaining parameter in the model, the noise variance σ^2. These quantities are examples of *scale* parameters, and suitable priors thereover are Gamma distributions (see, *e.g.*, Berger, 1985):

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathrm{Gamma}(\alpha_i|a, b),$$
$$p(\beta) = \mathrm{Gamma}(\beta|c, d),$$

with $\beta \equiv \sigma^{-2}$ and where

$$\mathrm{Gamma}(\alpha|a, b) = \Gamma(a)^{-1} b^a \alpha^{a-1} e^{-b\alpha}, \tag{6}$$

with $\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt$, the 'gamma function'. To make these priors non-informative (*i.e.* flat), we might fix their parameters to small values: *e.g.* $a = b = c = d = 10^{-4}$. However, by

4. Note that although it is not a characteristic of this parameter prior in general, for the case of the RVM that we consider here, the overall implied prior over functions is *data dependent* due to the appearance of $\mathbf{x}_n$ in the basis functions $K(\mathbf{x}, \mathbf{x}_n)$. This presents no practical difficulty, although we must take care in interpreting the "error-bars" implied by the model. In Appendix D.1 we consider this in further detail.

setting these parameters to zero, we obtain uniform hyperpriors (over a *logarithmic* scale). Since all scales are equally likely, a pleasing consequence of the use of such 'improper' hyperpriors here is that of scale-invariance: predictions are independent of linear scaling of both $\mathbf{t}$ and the basis function outputs so, for example, results do not depend on the unit of measurement of the targets. For completeness, the more detailed derivations offered in Appendix A will consider the case of general Gamma priors for $\boldsymbol{\alpha}$ and β, but in the main body of the paper, all further analysis and presented results will assume uniform scale priors with $a = b = c = d = 0$.

This formulation of prior distributions is a type of *automatic relevance determination* (ARD) prior (MacKay, 1994; Neal, 1996). Using such priors in a neural network, individual hyperparameters would typically control *groups* of weights — those associated with each input dimension x (this idea has also been applied to the input variables in 'Gaussian process' models). Should the evidence from the data support such a hypothesis, using a broad prior over the hyperparameters allows the posterior probability mass to concentrate at very large values of some of these α variables, with the consequence that the posterior probability of the associated weights will be concentrated at zero, thus effectively 'switching off' the corresponding inputs, and so deeming them to be 'irrelevant'.

Here, the assignment of an individual hyperparameter to each weight, or basis function, is the key feature of the relevance vector machine, and is responsible ultimately for its sparsity properties. To introduce an additional $N + 1$ parameters to the model may seem counter-intuitive, since we have already conceded that we have too many parameters, but from a Bayesian perspective, if we correctly 'integrate out' all such 'nuisance' parameters (or can approximate such a procedure sufficiently accurately), then this presents no problem from a methodological perspective (see Neal, 1996, pp. 16–17). Any subsequently observed 'failure' in learning is attributable to the form, not the parameterisation, of the prior over functions.

2.2 Inference

Having defined the prior, Bayesian inference proceeds by computing, from Bayes' rule, the posterior over all unknowns given the data:

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)}{p(\mathbf{t})}. \tag{7}$$

Then, given a new test point, $\mathbf{x}_*$, predictions are made for the corresponding target t_*, in terms of the predictive distribution:

$$p(t_*|\mathbf{t}) = \int p(t_*|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)\ p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t})\ d\mathbf{w}\ d\boldsymbol{\alpha}\ d\sigma^2. \tag{8}$$

To those familiar, or even not-so-familiar, with Bayesian methods, it may come as no surprise to learn that we cannot perform these computations in full analytically, and must seek an effective approximation.

We cannot compute the posterior $p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t})$ in (7) directly since we cannot perform the normalising integral on the right-hand-side, $p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)\ d\mathbf{w}\ d\boldsymbol{\alpha}\ d\sigma^2$. Instead, we decompose the posterior as:

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t}) = p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t}), \tag{9}$$

and note that we can compute analytically the posterior distribution over the weights since its normalising integral, $p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2) = \int p(\mathbf{t}|\mathbf{w},\sigma^2)\, p(\mathbf{w}|\boldsymbol{\alpha})\, d\mathbf{w}$, is a convolution of Gaussians. The posterior distribution over the weights is thus given by:[5]

$$p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\sigma^2) = \frac{p(\mathbf{t}|\mathbf{w},\sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2)}, \tag{10}$$

$$= (2\pi)^{-(N+1)/2}|\boldsymbol{\Sigma}|^{-1/2} \;\exp\left\{-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right\}, \tag{11}$$

where the posterior covariance and mean are respectively:

$$\boldsymbol{\Sigma} = (\sigma^{-2}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \mathbf{A})^{-1}, \tag{12}$$

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}, \tag{13}$$

with $\mathbf{A} = \mathrm{diag}(\alpha_0, \alpha_1, \ldots, \alpha_N)$.

We are now forced to adopt some approximation, and do so by representing the second term on the right-hand-side in (9), the hyperparameter posterior $p(\boldsymbol{\alpha},\sigma^2|\mathbf{t})$, by a delta-function at its mode[6], *i.e.* at its most-probable values $\boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}}$. We do so on the basis that this point-estimate is representative of the posterior in the sense that functions generated utilising the posterior mode values are near-identical to those obtained by sampling from the full posterior distribution. It is important to realise that this does not necessitate that the entire mass of the posterior be accurately approximated by the delta-function. For predictive purposes, rather than requiring $p(\boldsymbol{\alpha},\sigma^2|\mathbf{t}) \approx \delta(\boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}})$, we only desire

$$\int p(t_*|\boldsymbol{\alpha},\sigma^2)\delta(\boldsymbol{\alpha}_{\mathrm{MP}},\sigma^2_{\mathrm{MP}})\; d\boldsymbol{\alpha}\; d\sigma^2 \approx \int p(t_*|\boldsymbol{\alpha},\sigma^2)p(\boldsymbol{\alpha},\sigma^2|\mathbf{t})\; d\boldsymbol{\alpha}\; d\sigma^2, \tag{14}$$

to be a good approximation. This notion may be visualised by a thought experiment where we consider that we are utilising two identical basis functions $\phi_i(\mathbf{x})$ and $\phi_j(\mathbf{x})$. It follows from (15) shortly that the mode of $p(\boldsymbol{\alpha},\sigma^2|\mathbf{t})$ will not be unique, but will comprise an infinite 'ridge' where $\alpha_i^{-1} + \alpha_j^{-1}$ is some constant value. No delta-function can be considered to reasonably approximate the probability mass associated with this ridge, yet any point along it implies an identical predictive distribution and so (14) holds. All the evidence from the experiments presented in this paper suggests that this predictive approximation is very effective in general.

Relevance vector 'learning' thus becomes the search for the hyperparameter posterior mode, *i.e.* the maximisation of $p(\boldsymbol{\alpha},\sigma^2|\mathbf{t}) \propto p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2)p(\boldsymbol{\alpha})p(\sigma^2)$ with respect to $\boldsymbol{\alpha}$ and β.

5. Rather than evaluating (10) explicitly, there is a quicker way to obtain both the weight posterior (11) and the marginal likelihood (15) together. From Bayes rule simply write $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\sigma^2)p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2) = p(\mathbf{t}|\mathbf{w},\sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})$. Then, expanding the known right-hand-side quantities, gather together all terms in $\mathbf{w}$ that appear within the exponential, and complete the square, introducing some new terms in $\mathbf{t}$, to give by inspection the posterior Gaussian distribution $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\sigma^2)$. Combining all the remaining terms in $\mathbf{t}$ then gives $p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2)$ (15).
6. An alternative approach is to iteratively maximise a *variational* lower bound on $p(\mathbf{t})$, via a factorised approximation to $p(\mathbf{w},\boldsymbol{\alpha},\sigma^2|\mathbf{t})$, the joint posterior distribution over all the model parameters (Bishop and Tipping, 2000). This is a computationally intensive technique and in practice gives *expected* values for the hyperparameters which are identical to the point-estimates obtained by the method described here.

For the case of uniform hyperpriors (we consider the general case in Appendix A), we need only maximise the term $p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)$, which is computable and given by:

$$\begin{aligned} p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) &= \int p(\mathbf{t}|\mathbf{w}, \sigma^2) p(\mathbf{w}|\boldsymbol{\alpha}) \, d\mathbf{w}, \\ &= (2\pi)^{-N/2} |\sigma^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}|^{-1/2} \exp\left\{ -\frac{1}{2}\mathbf{t}^{\mathrm{T}}(\sigma^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\mathrm{T}})^{-1}\mathbf{t} \right\}. \end{aligned} \tag{15}$$

In related Bayesian models, this quantity is known as the *marginal likelihood*, and its maximisation known as the *type-II maximum likelihood* method (Berger, 1985). The marginal likelihood is also referred to as the "evidence for the hyperparameters" by MacKay (1992a), and its maximisation as the "evidence procedure".

2.3 Optimising the Hyperparameters

Values of $\boldsymbol{\alpha}$ and σ^2 which maximise (15) cannot be obtained in closed form, and here we summarise formulae for their iterative re-estimation. Further details concerning hyperparameter estimation, including alternative expectation-maximisation-based re-estimates, are given in Appendix A.

For $\boldsymbol{\alpha}$, differentiation of (15), equating to zero and rearranging, following the approach of MacKay (1992a), gives:

$$\alpha_i^{\mathrm{new}} = \frac{\gamma_i}{\mu_i^2}, \tag{16}$$

where μ_i is the i-th posterior mean weight from (13) and we have defined the quantities γ_i by:

$$\gamma_i \equiv 1 - \alpha_i \Sigma_{ii},$$

with Σ_{ii} the i-th diagonal element of the posterior weight covariance from (12) computed with the current $\boldsymbol{\alpha}$ and σ^2 values. Each $\gamma_i \in [0, 1]$ can be interpreted as a measure of how 'well-determined' its corresponding parameter w_i is by the data (MacKay, 1992a). For α_i large, where w_i is highly constrained by the prior, $\Sigma_{ii} \approx \alpha_i^{-1}$ and it follows that $\gamma_i \approx 0$. Conversely, when α_i is small and w_i fits the data, $\gamma_i \approx 1$.

For the noise variance σ^2, differentiation leads to the re-estimate:

$$(\sigma^2)^{\mathrm{new}} = \frac{\|\mathbf{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2}{N - \sum_i \gamma_i}. \tag{18}$$

Note that the 'N' in the denominator refers to the number of data examples and not the number of basis functions.

The learning algorithm thus proceeds by repeated application of (16) and (18), concurrent with updating of the posterior statistics $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ from (12) and (13), until some suitable convergence criteria have been satisfied (see Appendix A for some further implementation details). In practice, during re-estimation, we generally find that many of the α_i tend to infinity (or, in fact, become numerically indistinguishable from infinity given the machine accuracy)[7]. From (11), this implies that $p(w_i|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2)$ becomes highly (in princi-

7. This is true only for the case of the uniform hyperparameter priors adopted here. The use of more general Gamma priors, detailed in the appendix, would typically lead to some α's taking on large, but finite, values, and so implying some small, but non-zero, weights. Sparsity would then be realised through thresholding of the weights.

ple, infinitely) peaked at zero — *i.e.* we are *a posteriori* 'certain' that those w_i are zero. The corresponding basis functions can thus be 'pruned', and sparsity is realised.

2.4 Making Predictions

At convergence of the hyperparameter estimation procedure, we make predictions based on the posterior distribution over the weights, conditioned on the maximising values $\boldsymbol{\alpha}_{\mathrm{MP}}$ and σ^2_{MP}. We can then compute the predictive distribution, from (8), for a new datum $\mathbf{x}_*$ using (11):

$$p(t_*|\mathbf{t}, \boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}}) = \int p(t_*|\mathbf{w}, \sigma^2_{\mathrm{MP}}) p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}}) \; d\mathbf{w}. \tag{19}$$

Since both terms in the integrand are Gaussian, this is readily computed, giving:

$$p(t_*|\mathbf{t}, \boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}}) = \mathcal{N}(t_*|y_*, \sigma^2_*),$$

with

$$y_* = \boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_*), \tag{21}$$

$$\sigma^2_* = \sigma^2_{\mathrm{MP}} + \boldsymbol{\phi}(\mathbf{x}_*)^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{\phi}(\mathbf{x}_*). \tag{22}$$

So the predictive mean is intuitively $y(\mathbf{x}_*; \boldsymbol{\mu})$, or the basis functions weighted by the posterior mean weights, many of which will typically be zero. The predictive variance (or 'error-bars') comprises the sum of two variance components: the estimated noise on the data and that due to the uncertainty in the prediction of the weights. In practice, then, we may thus choose to set our parameters $\mathbf{w}$ to fixed values $\boldsymbol{\mu}$ for the purposes of point prediction, and retain $\boldsymbol{\Sigma}$ if required for computation of error bars (see Appendix D.1).

3. Sparse Bayesian Classification

Relevance vector classification follows an essentially identical framework as detailed for regression in the previous section. We simply adapt the target conditional distribution (likelihood function) and the link function to account for the change in the target quantities. As a consequence, we must introduce an additional approximation step in the algorithm.

For two-class classification, it is desired to predict the posterior probability of membership of one of the classes given the input $\mathbf{x}$. We follow statistical convention and generalise the linear model by 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:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma\{y(\mathbf{x}_n; \mathbf{w})\}^{t_n} \left[1 - \sigma\{y(\mathbf{x}_n; \mathbf{w})\}\right]^{1-t_n}, \tag{23}$$

where, following from the probabilistic specification, the targets $t_n \in \{0, 1\}$. Note that there is no 'noise' variance here.

However, unlike the regression case, we cannot integrate out the weights analytically, and so are denied 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 choose to utilise the following approximation procedure, as used by MacKay (1992b), which is based on Laplace's method:

1. For the current, fixed, values of $\boldsymbol{\alpha}$, the 'most probable' weights $\mathbf{w}_{\mathrm{MP}}$ are found, giving the location of the mode of the posterior distribution.

   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

$$\log\left\{P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})\right\} = \sum_{n=1}^{N}\left[t_n \log y_n + (1-t_n)\log(1-y_n)\right] - \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{A}\mathbf{w}, \qquad (24)$$

   with $y_n = \sigma\{y(\mathbf{x}_n;\mathbf{w})\}$. This is a standard procedure, since (24) is a penalised (regularised) logistic log-likelihood function, and necessitates iterative maximisation. Second-order Newton methods may be effectively applied, since the Hessian of (24), required next in step 2, is explicitly computed. We adapted the efficient 'iteratively-reweighted least-squares' algorithm (*e.g.* Nabney, 1999) to find $\mathbf{w}_{\mathrm{MP}}$.

2. Laplace's method is simply a quadratic approximation to the log-posterior around its mode. The quantity (24) is differentiated twice to give:

$$\nabla_{\mathbf{w}}\nabla_{\mathbf{w}} \log p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})\Big|_{\mathbf{w}_{\mathrm{MP}}} = -(\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A}), \qquad (25)$$

   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 $\mathbf{w}_{\mathrm{MP}}$.

3. Using the statistics $\boldsymbol{\Sigma}$ and $\mathbf{w}_{\mathrm{MP}}$ (in place of $\boldsymbol{\mu}$) of the Gaussian approximation, the hyperparameters $\boldsymbol{\alpha}$ are updated using (16) in identical fashion to the regression case.

At the mode of $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$, using (25) and the fact that $\nabla_{\mathbf{w}} \log p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})\Big|_{\mathbf{w}_{\mathrm{MP}}} = 0$, we can write:

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A})^{-1}, \qquad (26)$$

$$\mathbf{w}_{\mathrm{MP}} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\mathbf{t}. \qquad (27)$$

These equations are equivalent to the solution to a 'generalised least squares' problem (*e.g.* Mardia et al., 1979, p.172). Compared with (12) and (13), it can be seen that the Laplace approximation effectively maps the classification problem to a regression one with data-dependent (heteroscedastic) noise, with the inverse noise variance for ϵ_n given by $\beta_n = \sigma\{y(\mathbf{x}_n)\}\left[1-\sigma\{y(\mathbf{x}_n)\}\right]$.

How accurate is the Laplace approximation? In the Bayesian treatment of multilayer neural networks, the Gaussian approximation is considered a weakness of the method as the single mode of $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$ at $\mathbf{w}_{\mathrm{MP}}$ can often be unrepresentative of the overall posterior mass, particularly when there are multiple such modes (as is often the case). Here in this linearly-parameterised model, we know that $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$ is log-concave (as the Hessian is negative-definite everywhere). Not only is the posterior thus unimodal, log-concavity also implies that its tails are no heavier than $\exp(-|w|)$, and so we expect much better accuracy.[8]

8. An alternative Gaussian approximation is realisable using the variational bound of Jaakkola and Jordan (1997), exploited in the variational RVM (Bishop and Tipping, 2000).

For polychotomous classification, where the number of classes K is greater than two, the likelihood (23) is generalised to the standard multinomial form:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \sigma\{y_k(\mathbf{x}_n; \mathbf{w}_k)\}^{t_{nk}}, \tag{28}$$

where a conventional "one-of-K" target coding for t is used and the classifier has multiple outputs $y_k(\mathbf{x}; \mathbf{w}_k)$, each with its own parameter vector $\mathbf{w}_k$ and associated hyperparameters $\boldsymbol{\alpha}_k$ (although the hyperparameters could be shared amongst outputs if desired). The modified Hessian is computed from this (see Nabney, 1999) and inference proceeds as shown above. There is no need to heuristically combine multiple classifiers as is the case with, for example, the SVM. However, the size of $\boldsymbol{\Sigma}$ scales with K, which is a highly disadvantageous consequence from a computational perspective.

4. Relevance Vector Examples

In this section we first present some visualisations of the relevance vector machine applied to simple example synthetic data sets in both regression (Section 4.1) and classification (Section 4.2), followed by another synthetic regression example to demonstrate some potential extensions of the approach (Section 4.3). We then offer some illustrative 'benchmark' results in Section 4.4.

4.1 Relevance Vector Regression: the 'sinc' function

The function $\mathrm{sinc}(x) = \sin(x)/x$ has been a popular choice to illustrate support vector regression (Vapnik et al., 1997; Vapnik, 1998), where in place of the classification margin, the ϵ-*insensitive region* is introduced, a 'tube' of $\pm\epsilon$ around the function within which errors are not penalised. In this case, the support vectors lie on the edge of, or outside, this region. For example, using a univariate 'linear spline' kernel:

$$K(x_m, x_n) = 1 + x_m x_n + x_m x_n \min(x_m, x_n) - \frac{x_m + x_n}{2} \min(x_m, x_n)^2 + \frac{\min(x_m, x_n)^3}{3}, \tag{29}$$

and with $\epsilon = 0.01$, the approximation of $\mathrm{sinc}(x)$ based on 100 uniformly-spaced noise-free samples in $[-10, 10]$ utilises 36 support vectors as shown in Figure 1 (left).

In the RVM, we model the same data with the same kernel (29), which is utilised to define a set of basis functions $\phi_n(x) = K(x, x_n)$, $n = 1 \ldots N$. Typically, we will be tackling problems where the target function has some additive noise component, the variance of which we attempt to estimate with σ^2. However, for the purposes of comparison with this 'function approximation' SVM example, we model the 'sinc' function with a relevance vector machine but *fix* the noise variance in this case at 0.01^2 and then re-estimate $\boldsymbol{\alpha}$ alone. This setting of the noise standard deviation to 0.01 is intended to be analogous, in an approximate sense, to the setting the ϵ-insensitivity to the same value in the SVM. Using this fixed σ, the RVM approximator is plotted in Figure 1 (right), and requires only 9 relevance vectors. The largest error is 0.0070, compared to 0.0100 in the support vector case, and we have obtained the dual benefit of both increased accuracy and sparsity.

More representative of 'real' problems, Figure 2 illustrates the case where uniform noise (*i.e.* not corresponding to the RVM noise model) in $[-0.2, 0.2]$ is added to the targets. Again, a linear spline kernel was used. The trained RVM uses 6 relevance vectors, compared to 29 for the SVM. The root-mean-square (RMS) deviation from the true function for the RVM is 0.0245, while for the SVM it is 0.0291. Note that for the latter model, it was necessary to tune the parameters C and ϵ, in this case using 5-fold cross-validation. For the RVM, the analogues of these parameters (the α's and σ^2) are automatically estimated by the learning procedure.

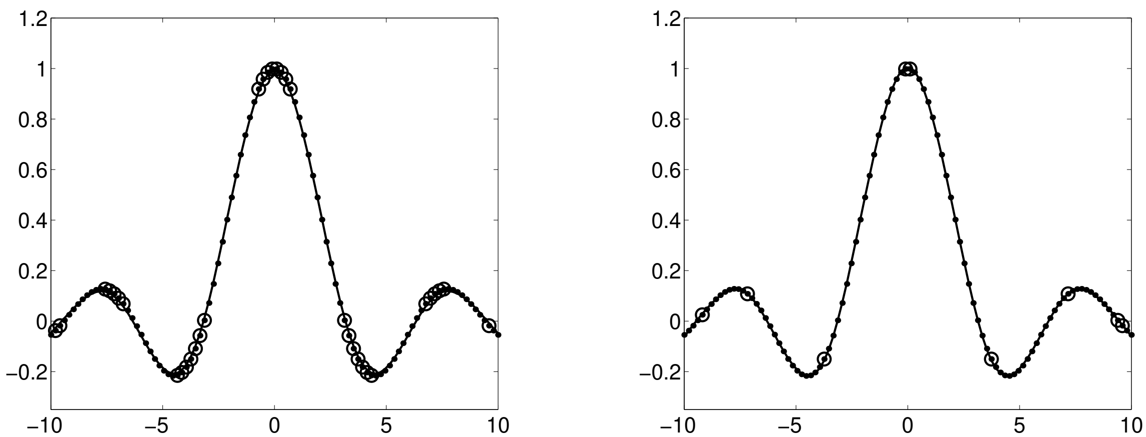

Figure 1: Support (left) and relevance (right) vector approximations to $\mathrm{sinc}(x)$ from 100 noise-free examples using 'linear spline' basis functions. The estimated functions are drawn as solid lines with support/relevance vectors shown circled.

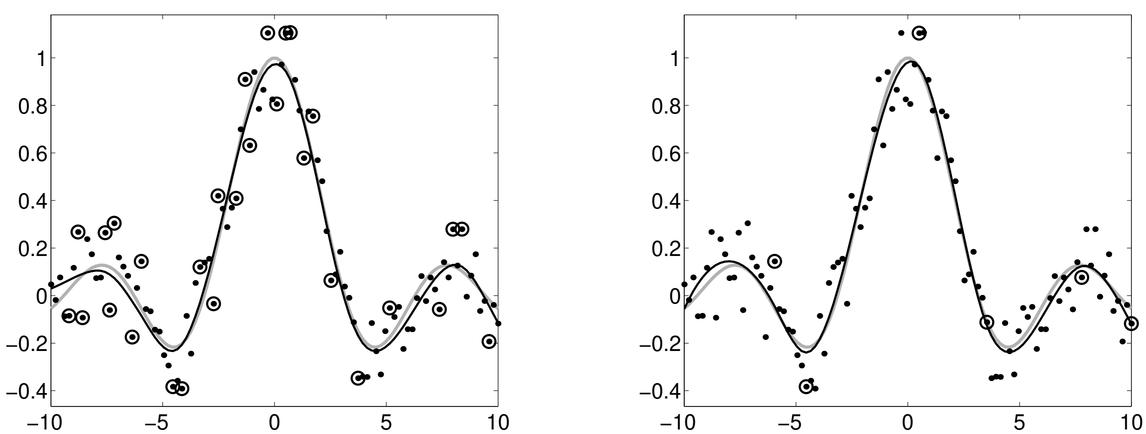

Figure 2: Support (left) and relevance (right) vector approximations to $\mathrm{sinc}(x)$, based on 100 noisy samples. The estimated functions are drawn as solid lines, the true function in grey, and support/relevance vectors are again shown circled.

4.2 Relevance Vector Classification: Ripley's synthetic data

We utilise artificially-generated data in two dimensions in order to illustrate graphically the selection of relevance vectors for classification. Both class 1 (denoted by '$\times$') and class 2 ('$\bullet$') were generated from mixtures of two Gaussians by Ripley (1996), with the classes overlapping to the extent that the Bayes error is around 8%.

A relevance vector classifier is compared to its support vector counterpart, using a 'Gaussian' kernel which we define as

$$K(\mathbf{x}_m, \mathbf{x}_n) = \exp(-r^{-2}\|\mathbf{x}_m - \mathbf{x}_n\|^2), \tag{30}$$

with r the 'width' parameter, chosen here to be 0.5. A value of C for the SVM was selected using 5-fold cross-validation on the training set. The results for a 100-example training set (randomly chosen from Ripley's original 250) are given in Figure 3. The test error (from the associated 1000-example test set) for the RVM (9.3%) is slightly superior to the SVM (10.6%), but the remarkable feature of contrast is the complexity of the classifiers. The support vector machine utilises 38 kernel functions compared to just 4 for the relevance vector method. This considerable difference in sparsity between the two methods is typical, as the later results on benchmark data sets support.

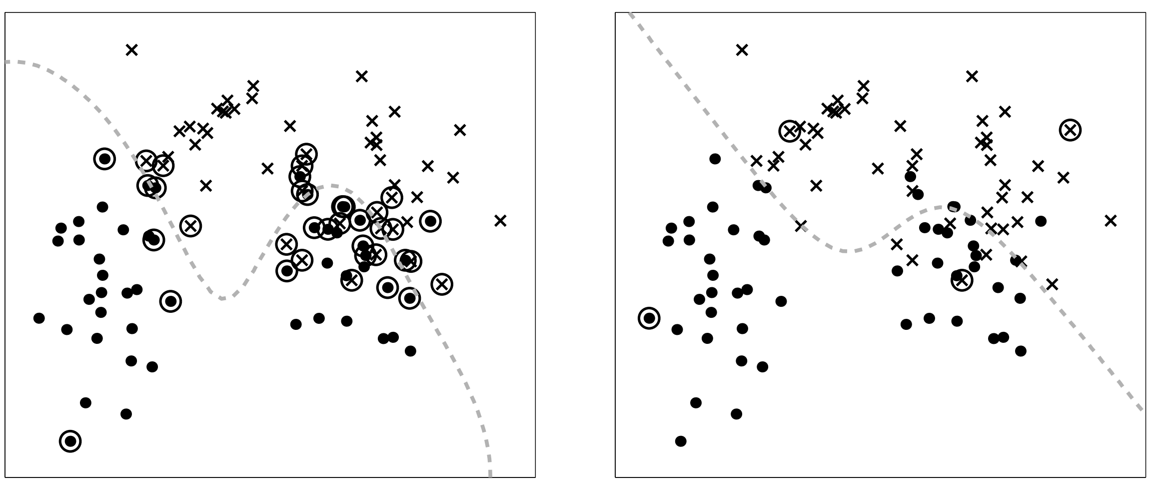

Figure 3: SVM (left) and RVM (right) classifiers on 100 examples from Ripley's Gaussian-mixture data set. The decision boundary is shown dashed, and relevance/support vectors are shown circled to emphasise the dramatic reduction in complexity of the RVM model.

Of interest also is the fact that, unlike for the SVM, the relevance vectors are some distance from the decision boundary (in $\mathbf{x}$-space), appearing more 'prototypical' or even 'anti-boundary' in character. Given further analysis, this observation can be seen to be consistent with the hyperparameter update equations given the form of the posterior induced by the Laplace approximation of Section 3. A more qualitative explanation is that the output of a basis function centred on or near the decision boundary is an unreliable indicator of class membership (*i.e.* its output is poorly-aligned with the data set in $\mathbf{t}$-space — see Section 5.2 for an illustration of this concept), and such basis functions are naturally penalised (deemed 'irrelevant') under the Bayesian framework. Of course, there is no implication that the utilisation of either boundary-located or prototypically-located functions is 'correct' in any sense.

4.3 Extensions

Before giving some example results on benchmark data sets, we use another synthetic example to demonstrate the potential of two advantageous features of the sparse Bayesian approach: the ability to utilise arbitrary basis functions, and the facility to directly 'optimise' parameters within the kernel specification, such as those which moderate the input scales.

This latter feature is of considerable importance: in both the SVM and RVM, it is necessary to choose the type of kernel function, and also to determine appropriate values for any associated parameters — *e.g.* the input scale (width) parameter[9] $\eta = r^{-2}$ of the Gaussian kernel (30). In the examples of Section 4.4 which follow, η is estimated by cross-validation for both the SVM and the RVM. This is a sensible and practical approach for a single such parameter, but is inapplicable if it is desired to associate an individual η_i with each input variable. Use of such multiple input scale parameters within kernels (or other basis functions) is inherently sensible, and as will be seen, can be an effective way of dealing with irrelevant input variables.

Consider now the problem of estimating the following, quite simple, two-dimensional function

$$y(x_1, x_2) = \mathrm{sinc}(x_1) + 0.1x_2,$$

based on 100 examples with additive Gaussian noise of standard deviation 0.1. There are two evident problems with direct application of a support or relevance vector model to this data:

- The function is linear in x_2, but this will be modelled rather unsatisfactorily by a superposition of nonlinear functions.
- The nonlinear element, $\mathrm{sinc}(x_1)$, is a function of x_1 alone, and so x_2 will simply add irrelevant 'noise' to the input, and thus output, of the basis functions and this will be reflected in the overall approximator.

These two features make the function difficult to learn accurately, and the function along with its SVM approximation (the RVM gives similar, only marginally superior, results here) is shown in Figure 4.

To improve upon the results shown in Figure 4 (right), we implement two modifications. We emphasised earlier that the RVM is really a specialisation of a Bayesian procedure defined for arbitrary basis sets and as such we are free to modify the type and number of

9. In the Gaussian kernel, this input scale parameter is explicit, but all non-trivial kernel functions, even those which do not typically incorporate such a parameter, are sensitive to the scaling of the input variables. As such, there may be considered to be an input scale parameter η_i associated with each input dimension, even if these are all implicitly assumed to be equal to unity.

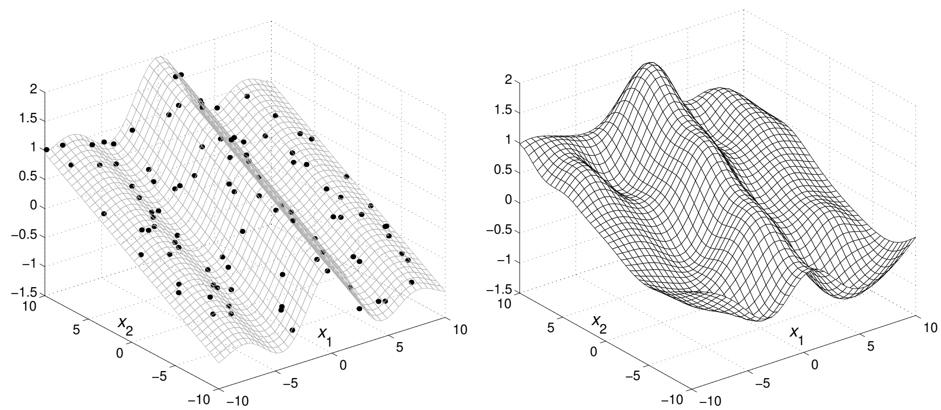

Figure 4: LEFT: the function $\mathrm{sinc}(x_1) + 0.1x_2$ along with the training data, and RIGHT: its approximation by a support vector model with a Gaussian kernel ($r = 3$).

basis functions that we feel might be useful in a given problem. Here, mirroring the general approach sometimes taken in Gaussian process regression (Rasmussen, 1996), we introduce the input variables as two extra 'functions'. This is achieved by simply appending two extra columns to the design matrix $\mathbf{\Phi}$ containing the x_1 and x_2 values, and introducing corresponding weights and hyperparameters which are updated identically to all others. We would hope that the weight associated with x_1 would be zero (and indeed, would be pruned), while that corresponding to x_2 should be approximately 0.1. In fact, to complicate the problem further here, we also introduced three additional quadratic terms x_1^2, x_2^2 and x_1x_2 which we hope to similarly prune.

A second modification is to directly optimise the marginal likelihood with respect to the kernel input scale parameters. For this problem we thus introduce the parameters η_1 and η_2 such that the kernel function becomes

$$K(\mathbf{x}_m, \mathbf{x}_n) = \exp\left\{-\eta_1(x_{m1} - x_{n1})^2 - \eta_2(x_{m2} - x_{n2})^2\right\}. \tag{31}$$

To estimate these parameters, at each iteration of the hyperparameter updates (16) a cycle of maximisation of the marginal likelihood (15) with respect to η_1 and η_2 was performed (using a gradient-based method). We give further implementation details in Appendix C.

Given these two modifications then, the final, and much more accurate, RVM approximating function is shown in Figure 5. The error and sparsity of this modified model is compared with the SVM in Table 1 and the estimates of the 'interesting' RVM parameters are shown in Table 2.

While Figure 4 and Table 1 indicate both qualitatively and quantitatively the improvement obtained with the modified RVM, Table 2 confirms that this is as a result of the model learning the 'correct' values of the newly introduced parameters. First, w_2 is a good approximation of the true function's value while all other candidates have been pruned. Second, $\eta_2 \ll \eta_1$, such that $K(\mathbf{x}_m, \mathbf{x}_n) \approx \exp\left\{-\eta_1(x_{m1} - x_{n1})^2\right\}$ and the basis functions depend approximately on input variable x_1 alone. One might fear that this procedure could

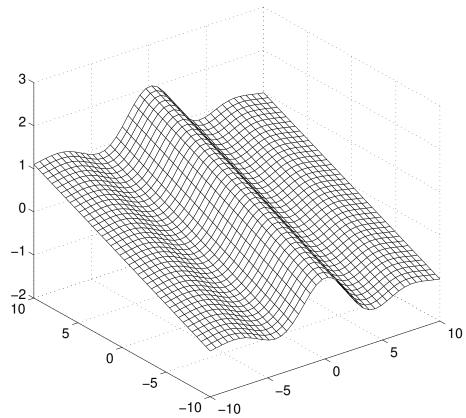

Figure 5: Approximation by a relevance vector model, with additional linear inputs and optimisation of the input scales. The noise estimate was $\sigma = 0.101$.

Model	**error**	**# functions**
SVM	0.0194	75
RVM	0.0053	8

Table 1: Root-mean-square error and number of basis functions required in approximation of the function $\sin(x_1)/x_1 + 0.1x_2$, using an SVM, and an RVM with additional linear input functions and optimised input scale parameters.

Parameter	**estimate**
w_{x_1}	0
w_{x_2}	0.102
$w_{x_1^2}$	0
$w_{x_2^2}$	0
$w_{x_1 x_2}$	0

Parameter	**estimate**
$\eta_1 \times 10^4$	997
$\eta_2 \times 10^4$	2

Table 2: LEFT: RVM parameter estimates for the weights associated with the additional basis functions. RIGHT: estimates of the two scale parameters.

'over-fit' and set all the η_i to large values (*i.e.* shrink all the 'widths' in the Gaussian towards zero). We offer some schematic insight into why this does not occur in Section 5.2.

Finally, we underline that these additional parameters have been successfully estimated *directly from the training set*, of 100 noisy samples, at the same time as estimating all other model parameters, including the data noise level. *No cross-validation was required.*

Although the above result is quite compelling, we must state some limitations to the approach:

- The interleaved two-stage training procedure leaves open the question of how exactly to combine optimisation over $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ (see Appendix C).
- The optimisation is computationally complex. While we can generally apply it to a single given data set, it was only practical to apply it to a single benchmark experiment in the following subsection.

4.4 Benchmark Comparisons

The tables which follow summarise regression and classification performance of the relevance vector machine on some example 'benchmark' data sets, comparing results for illustrative purposes with equivalent support vector machines. The number of training examples (N) and the number of input variables (d) are given for each data set, with further details regarding the data given in Appendix E. The prediction error obtained and the number of vectors (support or relevance) required, generally averaged over a number of repetitions, are given for both models. By way of summary, the RVM statistics were also normalised by those of the SVM and the overall average is displayed. A Gaussian kernel was utilised, and in all but a single case detailed shortly, its single input scale parameter chosen by 5-fold cross-validation.

4.4.1 Regression

			errors		*vectors*	
Data set	N	d	**SVM**	**RVM**	**SVM**	**RVM**
Sinc (Gaussian noise)	100	1	0.378	0.326	45.2	6.7
Sinc (Uniform noise)	100	1	0.215	0.187	44.3	7.0
Friedman #2	240	4	4140	3505	110.3	6.9
Friedman #3	240	4	0.0202	0.0164	106.5	11.5
Boston Housing	481	13	8.04	7.46	142.8	39.0
Normalised Mean			1.00	**0.86**	1.00	**0.15**

For the Friedman #1 data set, an additional benchmark result was obtained where the ten input scale parameters of the Gaussian kernel were optimised in the RVM: this is designated as '$\boldsymbol{\eta}$-RVM' below.

			errors			*vectors*		
Data set	N	d	**SVM**	**RVM**	η**-RVM**	**SVM**	**RVM**	η**-RVM**
Friedman #1	240	10	2.92	2.80	0.27	116.6	59.4	11.5

The dramatic improvement of $\boldsymbol{\eta}$-RVM is a consequence of the fact that the target function, as deliberately constructed by Friedman (1991), does not depend on input variables 6 to 10, and the influence of those distractor variables is suppressed by very low estimates for the corresponding η_k parameters. Unfortunately, computational resources limit the repetition of this extended $\boldsymbol{\eta}$-optimisation procedure for the complete set of benchmark experiments presented here. Typically, however, we do observe improvements in individual regression and classification experiments, although not generally as dramatic as that shown above.

4.4.2 Classification

Some examples of classification performance are given in the table below, and further details of the data are given in Appendix E. All problems are two-class, with the exception of the 'U.S.P.S.' handwritten digit set where, for computational reasons, we mirrored the SVM strategy of training ten separate dichotomous classifiers (rather than use the multinomial likelihood).

			errors		***vectors***	
Data set	N	d	**SVM**	**RVM**	**SVM**	**RVM**
Pima Diabetes	200	8	20.1%	19.6%	109	4
U.S.P.S.	7291	256	4.4%	5.1%	2540	316
Banana	400	2	10.9%	10.8%	135.2	11.4
Breast Cancer	200	9	26.9%	29.9%	116.7	6.3
Titanic	150	3	22.1%	23.0%	93.7	65.3
Waveform	400	21	10.3%	10.9%	146.4	14.6
German	700	20	22.6%	22.2%	411.2	12.5
Image	1300	18	3.0%	3.9 %	166.6	34.6
Normalised Mean			1.00	**1.08**	1.00	**0.17**

5. Perspectives on Sparsity

Both the illustrative and benchmark examples indicate that the presented Bayesian learning procedure is capable of producing highly sparse models. The purpose of this section is to offer further insight into the causes of this sparsity, and to this end we first look in more detail at the form of the weight prior distribution. Following that, we adopt a Gaussian process perspective in order to give a more graphical explanation.

5.1 The Prior over the Weights

From the Bayesian viewpoint, the relevance vector machine is sparse since most posterior probability mass is distributed over solutions with small numbers of basis functions, and the given learning algorithm finds one such solution. That sparse solutions are likely *a posteriori* relies of course on the prior: there must also be significant probability on sparse models *a priori*. Given the Gaussian specification of $p(\mathbf{w}|\boldsymbol{\alpha})$, though, it does not appear that we are utilising such a prior. However, the hierarchical nature of the prior disguises its overall character, and we need to integrate out the hyperparameters to discover the true identity of the prior over the weights.

In Section 2, the marginal likelihood (15) was obtained by marginalising over the weights. Alternatively, for a Gamma prior over the hyperparameters, it is also possible to integrate out α instead, independently for each weight, to obtain the marginal, or what might be considered the 'true', weight prior:

$$
\begin{aligned}
p(w_i) &= \int p(w_i|\alpha_i)p(\alpha_i)\; d\alpha_i, \\
&= \frac{b^a \Gamma(a+\frac{1}{2})}{(2\pi)^{\frac{1}{2}}\Gamma(a)}\; (b + w_i^2/2)^{-\left(a+\frac{1}{2}\right)},
\end{aligned} \tag{32}
$$

where $\Gamma(\cdot)$ is the gamma function as defined earlier. Equation (32) corresponds to the density of a Student-t distribution, and so the overall marginal weight prior is a product of independent Student-t distributions over the w_i. A visualisation of this Student-t prior, alongside a Gaussian, is given in Figure 6. For the case of the uniform hyperprior, with $a = b = 0$, we obtain the improper prior $p(w_i) \propto 1/|w_i|$. Intuitively, this looks very much like a 'sparse' prior since it is sharply peaked at zero like the popular Laplace prior $p(w_i) \propto \exp(-|w_i|)$, which has been utilised to obtain sparsity both in Bayesian contexts (Williams, 1995), and, taking the negative log, as the 'L_1' regulariser $\sum_i |w_i|$ elsewhere (*e.g.* Chen et al., 1995; Grandvalet, 1998; Smola et al., 1999). The implication is that although superficially we appear to be utilising a non-sparse Gaussian prior over the weights, in truth the hierarchical formulation implies that the real weight prior is one which can be clearly recognised as encouraging sparsity.

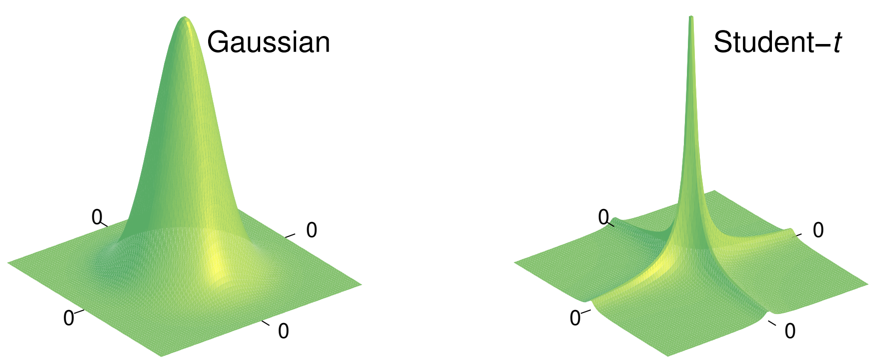

Figure 6: LEFT: an example Gaussian prior $p(\mathbf{w}|\boldsymbol{\alpha})$ in two dimensions. RIGHT: the prior $p(\mathbf{w})$, where the hyperparameters have been integrated out to give a product of Student-t distributions. Note that the probability mass is concentrated both at the origin and along 'spines' where one of the two weights is zero.

Unfortunately, we cannot continue the Bayesian analysis down this route to compute $p(\mathbf{w}|\mathbf{t})$, since the marginal $p(\mathbf{w})$ is no longer Gaussian. However, we might pose the question: why not integrate out $\boldsymbol{\alpha}$ explicitly and maximise over $\mathbf{w}$ — *i.e.* find the mode of $p(\mathbf{w}|\mathbf{t})$ — instead of *vice-versa* as detailed in Section 2? This alternative approach would be equivalent to maximisation of a penalised likelihood function of the form:

$$\mathcal{L}(\mathbf{w}) = -\frac{\beta}{2}\sum_{n=1}^{N}[t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)]^2 - \sum_{i=0}^{N}\log|w_i|, \tag{33}$$

where we note that the presence of the log differentiates (33) from L_1 regularisation. In fact, we must discount this alternative inference strategy since we typically find that $\mathcal{L}(\mathbf{w})$, and so $p(\mathbf{w}|\mathbf{t})$, is significantly multi-modal, often extremely so. These modes occur where the likelihood, which has the form of a Gaussian in $\mathbf{w}$-space, overlaps the 'spines' (see Figure 6, right) of the prior. (We remind the reader here that the $\boldsymbol{\alpha}$-conditional posterior, $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$, which we maximise in step 1 of the classification case in Section 3, is log concave and unimodal.) The implication therefore is that the marginalised weight posterior mode is highly unrepresentative of the distribution of posterior probability mass. A good illustration of this phenomenon is given by MacKay (1999) in the context of single-hyperparameter models. Conversely, as discussed in Section 2.2, all the experimental evidence for relevance vector learning suggests that $\boldsymbol{\alpha}_{\mathrm{MP}}$ *is* representative of the posterior $p(\boldsymbol{\alpha}|\mathbf{t})$.

We finally note here that a model with a *single* hyperparameter governing the inverse prior variance of *all* weights would place less probability on sparse models than the 'relevance' prior we use here. Such a prior does not specify independence over the weights, the magnitudes of which are therefore coupled to all other weights through the common hyperparameter (so, *a priori*, two large weights would be more probable than one large and one small, for example).

5.2 A Gaussian Process View

We first note that relevance vector learning in regression is maximising the probability of the N-dimensional vector of target values $\mathbf{t}$ under the model of (15). This is a *Gaussian process* model (Rasmussen, 1996; MacKay, 1998; Williams, 1999): *i.e.* $p(\mathbf{t}) = \mathcal{N}(\mathbf{t}|\mathbf{0},\mathbf{C})$ where we re-write $\mathbf{C}$ from (15) as:

$$\mathbf{C} = \sigma^2\mathbf{I} + \sum_{i=0}^{N}\alpha_i^{-1}\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}, \tag{34}$$

and $\mathbf{v}_i = (\phi_i(\mathbf{x}_1), \phi_i(\mathbf{x}_2), \ldots, \phi_i(\mathbf{x}_N))^{\mathrm{T}}$ is an N-vector containing the output of basis function i evaluated at all the training examples (whereas $\boldsymbol{\phi}(\mathbf{x})$ earlier denoted the vector of all the basis functions evaluated at a single datum). This implicit, 'weightless', formulation of the Bayesian model can be adopted explicitly in other contexts to realise sparsity, for example in 'sparse kernel PCA' (Tipping, 2001). In Figure 7 we illustrate schematically an idealised two-dimensional 'projection' of the Gaussian process.

The basis functions $\mathbf{v}_i$ specify directions in $\mathbf{t}$-space whose outer product contribution to the overall covariance is modulated by the inverse hyperparameters α_i^{-1}. Adjustment of each α_i thus changes both the size and shape of $\mathbf{C}$, while adjusting σ^2 grows or shrinks it equally in all directions. The RVM learning procedure iteratively updates the noise level along with all the contributions of the vectors $\mathbf{v}_i$ in order to make the observed data set $\mathbf{t}$ most probable.

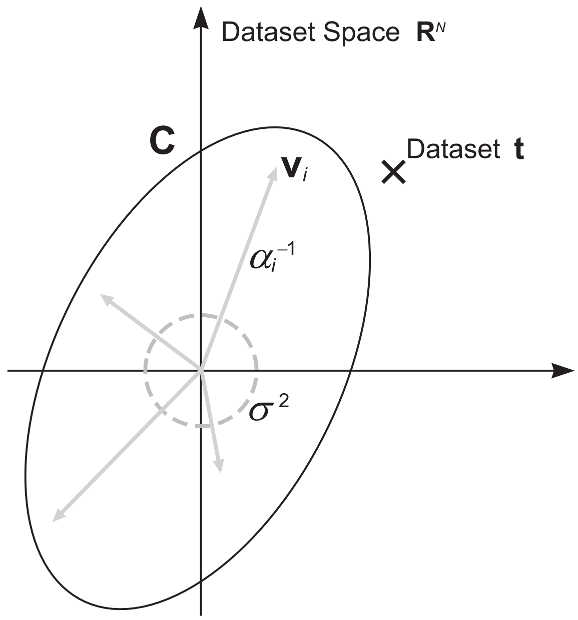

Figure 7: A Gaussian process view of the regression RVM, showing a two-dimensional 'projection' of the N-dimensional data set space. The data set to be modelled is marked with the cross, the weighted directions of some of the N basis vectors are shown by grey arrows emanating from the origin, the noise component of the covariance is denoted by a dashed grey circle, and the form of the overall covariance $\mathbf{C}$ is illustrated by the ellipse.

To see how sparsity can arise, consider, then, a trivial example with just a single basis function $\mathbf{v}_1$, which is not well 'aligned' with $\mathbf{t}$. In Figure 8 (left), one possible $\mathbf{C}$ is shown where $\mathbf{v}_1$ contributes significantly. In Figure 8 (right), $\mathbf{t}$ is explained by the noise alone. In both cases, the normalisation term $|\mathbf{C}|^{-1/2}$ is the same, but based on the unit Mahalanobis covariance ellipses, the noise explanation is more probable. Intuitively, this occurs since it 'wastes' less probability mass in the direction of $\mathbf{v}_1$.

Of course, in practice there will be $N+1$ $\mathbf{v}$-vectors all competing with each other and the noise to explain the data in an N-dimensional space. The general point though still holds: if we consider increasing the contribution of a given $\mathbf{v}_i$, and if the 'spread' in $\mathbf{C}$ orthogonal to $\mathbf{t}$ is greater than that in the direction of $\mathbf{t}$, then the data can be better explained by increasing the noise variance σ^2, as this increases $\mathbf{C}$ identically in all directions (and so increases $|\mathbf{C}|$ less). Thus, at convergence of the $\boldsymbol{\alpha}$-optimisation, all the deleted basis functions are those for which the noise level (whether also optimised or set in advance) can better explain the data.

A further intuition that may be gleaned from this pictorial perspective is that we might expect $\mathbf{v}_i$ that lie more 'in the direction of' $\mathbf{t}$ to be more relevant[10], and so in classification,

10. Unfortunately, as it would simplify learning if so, the R relevance vectors that are found do *not* correspond to the R most closely aligned $\mathbf{v}_i$ vectors.

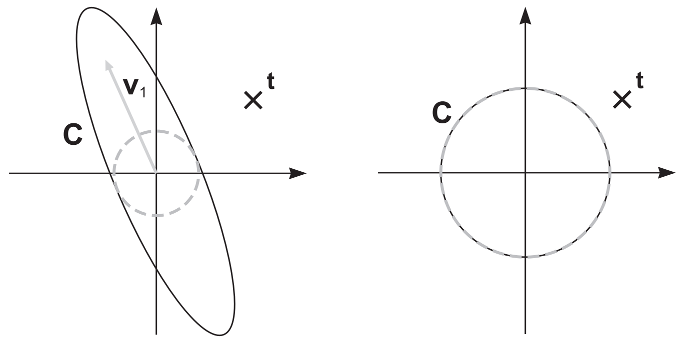

Figure 8: Two potential explanations of a data set $\mathbf{t}$. Left: using a basis function in conjunction with noise. Right: using noise alone. Covariance ellipses depicting unit Mahalanobis distances are shown. For both Gaussians, $|\mathbf{C}|$ is the same.

basis functions centred near the decision boundary are unlikely to be retained. Note that this does not rule out functions located significantly on the 'wrong' side of the boundary since $\mathbf{v}_i$ contributes to $\mathbf{C}$ in (34) as an outer product and so can also be relevant if it is well aligned with $-\mathbf{t}$. In fact, an example of such a relevance vector may be seen in Figure 3 (right) earlier.

Finally here, we may glean some understanding of why it may be profitable to optimise the marginal likelihood with respect to input scale parameters as was outlined in Section 4.3. Figure 9 provides an approximate representation (again projected to two dimensions) of the marginal likelihood model which results from using a Gaussian kernel. For a very narrow kernel width, it can be seen from (34) that the Gaussian process covariance becomes diagonal. At the other extreme, a very large width implies that the data set is modelled only by the (isotropic) noise. It follows that the data set can become more probable at some intermediate width. While it is not necessarily the case that this is the 'optimal' width in terms of model accuracy, it can at least be seen why the marginal likelihood is not optimised by inappropriately shrinking the width of the kernel to zero, as would occur if maximising the conventional likelihood.

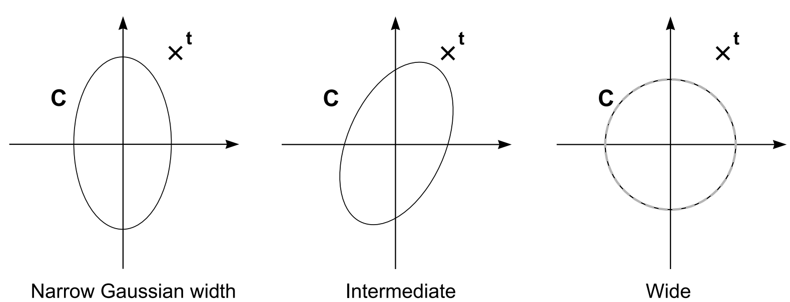

Figure 9: Effect of the width of a Gaussian kernel on the marginal probability of a data set $\mathbf{t}$. Covariance ellipses depicting lines of equal probability are shown, with the intermediate Gaussian width (centre) giving the greatest likelihood for $\mathbf{t}$.

6. Discussion

In this paper we have sought to provide the technical detail and experimental justification for a Bayesian approach to sparse learning in linearly-parameterised models, and we have endeavoured to offer some understanding of the mechanism through which sparsity is realised. Although we have concentrated on comparison with the popular 'support vector machine', we would underline once more that the presented method is a general one.

We do not seek to state that this approach is definitively superior to any other; rather that it is a regression and classification tool worthy of serious consideration and one which offers some quite compelling and advantageous features, some of which we summarise here:

- Generalisation is typically very good. While it was not the intention to present a comprehensive comparison with other methods, it was demonstrated in Section 4.4 that results comparable with the state-of-the-art can be obtained.
- 'Learned' models are typically *highly* sparse. Indeed, for the visualisable examples given in Section 4, the models appear almost optimally compact.
- In classification, the model gives estimates of the posterior probability of class membership. This is a highly important, but often overlooked, feature of any practical pattern recognition system.
- There are no 'nuisance' parameters to validate, in that the type-II maximum likelihood procedure automatically sets the 'regularisation' parameters, while the noise variance can be similarly estimated in regression.
- There is no constraint over the number or type of basis functions that may be used, although we note below the computational implications of using a large number thereof.
- We can optimise 'global' parameters, such as those which moderate the input variable scales. This is a very powerful feature, as it is impossible to set such scale parameters by cross-validation. On a single benchmark task and on isolated problems, we have found that this can be highly effective, though for computational reasons, we cannot yet present a more extended experimental assessment of the technique.

Another potential advantage of a Bayesian approach is that the fully marginalised probability of a given model (basis set) $\mathcal{M}$, $p(\mathbf{t}|\mathcal{M}) = \int p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2, \mathcal{M})p(\boldsymbol{\alpha})p(\sigma^2)\ d\boldsymbol{\alpha}\ d\sigma^2$, may be considered a measure of the merit of the model (*e.g.* could provide a criterion for selection of the kernel). However, we have already indicated that this quantity cannot be computed analytically, and the simple and practical numerical approximations that we have employed have proved ineffective. In practice, we would use a cross-validation approach, and finding a criterion (which should require significantly less computational overhead) based on an effective approximation to $p(\mathbf{t}|\mathcal{M})$ remains an open research question.

We finally note that the primary disadvantage of the sparse Bayesian method is the computational complexity of the learning algorithm. Although the presented update rules are very simple in form, their required memory and computation scale respectively with the square and cube of the number of basis functions. For the RVM, this implies that the algorithm presented here becomes less practical when the training examples number several thousand or more. In light of this we have recently developed a much more efficient strategy for maximising the marginal likelihood, which is discussed briefly in Appendix B.2, and we intend to publish further details shortly.

Acknowledgments

The author wishes to thank Chris Bishop, Chris Williams, Neil Lawrence, Ralf Herbrich, Anita Faul, John Platt and Bernhard Schölkopf for helpful discussions, and the latter two again for the use of their SVM code. In addition, the author was appreciative of many excellent comments from the reviewers.

Appendix A. Further Details of Relevance Vector Learning

Relevance vector learning involves the maximisation of the product of the marginal likelihood and the priors over $\boldsymbol{\alpha}$ and σ^2 (or $\beta \equiv \sigma^{-2}$ for convenience). Equivalently, and more straightforwardly, we maximise the log of this quantity. In addition, we also choose to maximise with respect to $\log \alpha$ and $\log \beta$ — this is convenient since in practice we assume uniform hyperpriors over a logarithmic scale, and the derivatives of the prior terms vanish in this space. So, retaining for now general Gamma priors over $\boldsymbol{\alpha}$ and β,we maximise:

$$\log p(\mathbf{t}|\log \boldsymbol{\alpha}, \log \beta) + \sum_{i=0}^{N} \log p(\log \alpha_i) + \log p(\log \beta), \tag{35}$$

which, ignoring terms independent of $\boldsymbol{\alpha}$ and β, and noting that $p(\log \alpha) = \alpha p(\alpha)$, gives the objective function:

$$\begin{aligned}\mathcal{L} = -\frac{1}{2}\left[\log|\beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\mathrm{T}}| + \mathbf{t}^{\mathrm{T}}(\beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\mathrm{T}})^{-1}\mathbf{t}\right] + \\ \sum_{i=0}^{N}(a \log \alpha_i - b\alpha_i) + c \log \beta - d\beta.\end{aligned} \tag{36}$$

Note that the latter terms disappear with a, b, c, d set to zero.

We now consider how to robustly and efficiently compute $\mathcal{L}$, outline the derivation of the hyperparameter updates, and consider the numerical difficulties involved.

A.1 Computing the Log Objective Function

The matrix $\beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\mathrm{T}}$, which appears in the first two terms in $\mathcal{L}$, is of size $N \times N$. However, computation of both of the terms of interest may be written as a function of the posterior weight covariance $\mathbf{\Sigma} = (\mathbf{A} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}$. This matrix is $M \times M$, where M is the number of basis functions in the model. While initially $M = N + 1$, in practice many basis functions will be 'deleted' during optimisation (see B.1 shortly) and M will decrease considerably giving significant computational advantages as optimisation progresses.

We compute the first term by exploiting the determinant identity (see, *e.g.*, Mardia et al., 1979, Appendix A):

$$|\mathbf{A}||\beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\mathrm{T}}| = |\beta^{-1}\mathbf{I}||\mathbf{A} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}|,$$

giving

$$\log|\beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\mathrm{T}}| = -\log|\mathbf{\Sigma}| - N\log\beta - \log|\mathbf{A}|. \tag{38}$$

Using the Woodbury inversion identity:

$$(\beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\mathrm{T}})^{-1} = \beta\mathbf{I} - \beta\mathbf{\Phi}(\mathbf{A} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\beta,$$

the second, data-dependent, term may be expressed as

$$\begin{aligned}\mathbf{t}^{\mathrm{T}}(\beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^{\mathrm{T}})^{-1}\mathbf{t} &= \beta\mathbf{t}^{\mathrm{T}}\mathbf{t} - \beta\mathbf{t}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Sigma}\mathbf{\Phi}^{\mathrm{T}}\beta\mathbf{t},\\ &= \beta\mathbf{t}^{\mathrm{T}}\left(\mathbf{t} - \mathbf{\Phi}\boldsymbol{\mu}\right), \end{aligned} \tag{40}$$

with $\boldsymbol{\mu} = \beta\mathbf{\Sigma}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$, the posterior weight mean. Note that (40) may be also be re-expressed as:

$$\begin{aligned}\beta\mathbf{t}^{\mathrm{T}}\left(\mathbf{t} - \mathbf{\Phi}\boldsymbol{\mu}\right) &= \beta\|\mathbf{t} - \mathbf{\Phi}\boldsymbol{\mu}\|^2 + \beta\mathbf{t}^{\mathrm{T}}\mathbf{\Phi}\boldsymbol{\mu} - \beta\boldsymbol{\mu}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\boldsymbol{\mu},\\ &= \beta\|\mathbf{t} - \mathbf{\Phi}\boldsymbol{\mu}\|^2 + \boldsymbol{\mu}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu} - \beta\boldsymbol{\mu}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\boldsymbol{\mu},\\ &= \beta\|\mathbf{t} - \mathbf{\Phi}\boldsymbol{\mu}\|^2 + \boldsymbol{\mu}^{\mathrm{T}}\mathbf{A}\boldsymbol{\mu}, \end{aligned} \tag{41}$$

which corresponds to the penalised log-likelihood evaluated using the posterior mean weights. The terms in (38) are sometimes referred to as the "Ockham factors".

Computation of $\mathbf{\Sigma}$, its determinant and $\boldsymbol{\mu}$ is achieved robustly through Cholesky decomposition of $\mathbf{A} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$, an $O(M^3)$ procedure.

A.2 Derivatives and Updates

A.2.1 The hyperparameters

The derivatives of (36) with respect to $\log\boldsymbol{\alpha}$ are:

$$\frac{\partial\mathcal{L}}{\partial\log\alpha_i} = \frac{1}{2}\left[1 - \alpha_i(\mu_i^2 + \Sigma_{ii})\right] + a - b\alpha_i. \tag{42}$$

Setting this to zero and solving for α_i gives a re-estimation rule:

$$\alpha_i^{\mathrm{new}} = \frac{1 + 2a}{\mu_i^2 + \Sigma_{ii} + 2b}. \tag{43}$$

This is equivalent to an *expectation-maximisation* (EM) update (see A.3 below) and so is guaranteed to locally maximise $\mathcal{L}$. However, setting (42) to zero, and following MacKay (1992a) in defining quantities $\gamma_i \equiv 1 - \alpha_i\Sigma_{ii}$, leads to the following update:

$$\alpha_i^{\mathrm{new}} = \frac{\gamma_i + 2a}{\mu_i^2 + 2b}, \tag{44}$$

which was observed to lead to much faster convergence although it does not benefit from the guarantee of local maximisation of $\mathcal{L}$. In practice, we did not encounter any optimisation difficulties (*i.e.* 'downhill' steps) utilising (44).

A.2.2 The noise variance

Derivatives with respect to $\log \beta$ are:

$$\frac{\partial \mathcal{L}}{\log \beta} = \frac{1}{2}\left[\frac{N}{\beta} - \|\mathbf{t} - \mathbf{\Phi}\boldsymbol{\mu}\|^2 - \mathrm{tr}\left(\mathbf{\Sigma}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)\right] + c - d\beta, \tag{45}$$

and since $\mathrm{tr}\left(\mathbf{\Sigma}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)$ can be re-written as $\beta^{-1}\sum_i \gamma_i$, setting the derivative to zero and rearranging and re-expressing in terms of σ^2 gives:

$$(\sigma^2)^{\mathrm{new}} = \frac{\|\mathbf{t} - \mathbf{\Phi}\boldsymbol{\mu}\|^2 + 2d}{N - \sum_i \gamma_i + 2c}. \tag{46}$$

A.3 Expectation-Maximisation (EM) Updates

Another strategy to maximise (36) is to exploit an EM formulation, treating the weights as the 'hidden' variables and maximise $E_{\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\beta}\left[\log p(\mathbf{t}|\mathbf{w},\beta)p(\mathbf{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})p(\beta)\right]$, where the operator $E_{\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\beta}[\cdot]$ denotes an expectation with respect to the distribution over the weights given the data and hidden variables, the posterior $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\beta)$.

For $\boldsymbol{\alpha}$, ignoring terms in the logarithm independent thereof, we equivalently maximise:

$$E_{\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\beta}\left[\log p(\mathbf{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})\right], \tag{47}$$

which through differentiation gives an update:

$$\alpha_i = \frac{1 + 2a}{\langle w_i^2 \rangle + 2b}, \tag{48}$$

and where, from the posterior (11), $\langle w_i^2 \rangle \equiv E_{\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\beta}\left[w_i^2\right] = \Sigma_{ii} + \mu_i^2$. This is therefore equivalent to the (less efficient) gradient update (43) earlier.

Following the corresponding procedure for the noise level σ^2 we maximise;

$$E_{\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\beta}\left[\log p(\mathbf{t}|\mathbf{w},\beta)p(\beta)\right], \tag{49}$$

which gives

$$(\sigma^2)^{\mathrm{new}} = \frac{\|\mathbf{t} - \mathbf{\Phi}\boldsymbol{\mu}\|^2 + (\sigma^2)^{\mathrm{old}}\sum_i \gamma_i + 2d}{N + 2c}. \tag{50}$$

Appendix B. Computational Considerations

B.1 Numerical Accuracy

The posterior covariance $\mathbf{\Sigma}$ is computed as the inverse of the 'Hessian' matrix $\mathbf{H} = \mathbf{A} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$ which, although positive definite in theory, may become numerically singular in practice. As optimisation of the hyperparameters progresses, the range of α-values typically becomes highly extended as many tend towards very large values. Indeed, for $a, b, c, d = 0$, many α's typically would tend to infinity if machine precision permitted. In fact, ill-conditioning of the Hessian matrix becomes a problem when, approximately, the ratio of the smallest to largest α-values is in the order of the 'machine precision'.

Consider the case of a single $\alpha_i \to \infty$, where for convenience of presentation we choose $i = 1$, the first hyperparameter. Using the expression for the inverse of a partitioned matrix, it can be shown that:

$$\boldsymbol{\Sigma} \to \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & (\mathbf{A}_{-i} + \beta \boldsymbol{\Phi}_{-i}^{\mathrm{T}} \boldsymbol{\Phi}_{-i})^{-1} \end{bmatrix} \tag{51}$$

where the subscript '$-i$' denotes the matrix with the appropriate i-th row and/or column removed. The term $(\mathbf{A}_{-i} + \beta \boldsymbol{\Phi}_{-i}^{\mathrm{T}} \boldsymbol{\Phi}_{-i})^{-1}$ is of course the posterior covariance computed with basis function i 'pruned'. Furthermore, it follows from (51) and (13) that $\mu_i \to 0$ and as a consequence of $\alpha_i \to \infty$, the model intuitively becomes exactly equivalent to one with basis function $\phi_i(\mathbf{x})$ excluded.

We may thus choose to avoid ill-conditioning by pruning the corresponding basis function from the model at that point (*i.e.* by deleting the appropriate column from $\boldsymbol{\Phi}$). This sparsification of the model during optimisation implies that we typically experience a very considerable and advantageous acceleration of the learning algorithm. The potential disadvantage is that if we believed that the marginal likelihood might be increased by re-introducing those deleted basis functions (*i.e.* reducing α_i from infinity) at a later stage, then their permanent removal would be suboptimal. For reassurance, at the end of optimisation we can efficiently compute the sign of the gradient of the marginal likelihood with respect to *all* α's corresponding to deleted basis functions. A negative gradient would imply that reducing an infinite α, and therefore reintroducing the corresponding basis function, would improve the likelihood. So far, no such case has been found.

The intuitive and reliable criterion we used for deletion of basis functions and their weights at each iteration in regression was to remove those whose 'well-determinedness' factors γ_i fell below the machine precision ($\approx 2.22 \times 10^{-16}$ in our case, the smallest ϵ such that $1 + \epsilon \neq 1$). As a result, presumably, of inaccuracies introduced by the Laplace approximation step, in classification a slightly more robust method was required, with weights deleted for which $\alpha_i > 10^{12}$. Note that the action of deleting such a basis function should not in theory change the value of the objective function $\mathcal{L}$ (in practice, it should change only negligibly) since in (38), the infinite part of the terms $\log|\boldsymbol{\Sigma}| = \log|\boldsymbol{\Sigma}_{-i}| - \log\alpha_i$ and $\log|\mathbf{A}| = \log|\mathbf{A}_{-i}| + \log\alpha_i$ cancel.

B.2 Algorithm Complexity

The update rules for the hyperparameters depend on computing the posterior weight covariance matrix, which requires an inverse operation (in fact, Cholesky decomposition) of order $O(M^3)$ complexity and $O(M^2)$ memory storage, with M the number of basis functions. Although, typically, the pruning discussed above rapidly reduces M to a manageable size in most problems, $M = N + 1$ for the RVM model at initialisation, and N may be very large. This of course leads to extended training times, although the disadvantage of this is significantly offset by the lack of necessity to perform cross-validation over nuisance parameters, such as for C and ϵ in the SVM. So, for example, with the exception of the larger data sets (*e.g.* roughly $N > 700$) the benchmark results in Section 4.4 were obtained more quickly for the RVM (this observation depends on the exact implementations and cross-validation schedules of course).

Even so, for large data sets, with computation scaling approximately in $O(N^3)$, the full RVM algorithm becomes prohibitively expensive to run. We have therefore developed an alternative algorithm to maximise the marginal likelihood which is 'constructive'. It starts with a single basis function, the bias, and both adds in further basis functions, or deletes current ones, as appropriate, rather than starting with all possible candidates and pruning. This is a much more efficient approach, as the number of basis functions included at any step in the algorithm tends to remain low. It is, however, a more 'greedy' optimisation strategy, although our preliminary results show little, if any, loss of accuracy compared to the standard algorithm. This appears a very promising mechanism for ensuring the sparse Bayesian approach remains practical even for very large basis function sets.

Appendix C. Adapting Input Scale Parameters

In Section 4.3 we considered how parameters within the kernel specification could be adapted to improve the marginal likelihood. We note that in general, we may still prefer to select an individual kernel parameter (for example, the 'width' parameter of a Gaussian) via cross-validation. Here, however, we consider the case of multiple input scale parameters, where cross-validation is not an option and where by optimising such parameters considerable performance gains may be achieved. The benefit is typically most notable in regression problems, as exemplified by the Friedman #1 data set benchmark in Section 4.4.1.

Assume that the basis functions take the form

$$\phi_m(\mathbf{x}) = \phi_m(\sqrt{\eta_1}x_1, \sqrt{\eta_2}x_2, \ldots, \sqrt{\eta_d}x_d), \tag{52}$$

where the input vector $\mathbf{x}$ comprises d variables $(x_1, \ldots, x_d)$ and the corresponding scale (inverse squared 'width') parameters are $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_d)$. We notate $\phi_{nm} = \phi_m(\mathbf{x}_n; \boldsymbol{\eta})$ to be the elements of the design matrix $\boldsymbol{\Phi}$ as before. The gradient of the likelihood $\mathcal{L}$ (36) with respect to the k-th scale parameter is first written in the form:

$$\frac{\partial \mathcal{L}}{\partial \eta_k} = \sum_{n=1}^{N}\sum_{m=1}^{N} \frac{\partial \mathcal{L}}{\partial \phi_{nm}} \frac{\partial \phi_{nm}}{\partial \eta_k}. \tag{53}$$

Note that $m = 0$ refers to the bias, and so, being independent of $\mathbf{x}$, does not enter into (53). The first term in (53) is independent of the basis function parameterisation and it is convenient to collect all the terms into a matrix $\mathbf{D}$ such that $\mathbf{D}_{nm} = \partial\mathcal{L}/\partial\phi_{nm}$. Evaluating these derivatives then gives:

$$\mathbf{D} = (\mathbf{C}^{-1}\mathbf{t}\mathbf{t}^{\mathrm{T}}\mathbf{C}^{-1} - \mathbf{C}^{-1})\boldsymbol{\Phi}\mathbf{A}^{-1}, \tag{54}$$

$$= \beta\left[(\mathbf{t} - \mathbf{y})\boldsymbol{\mu}^{\mathrm{T}} - \boldsymbol{\Phi}\boldsymbol{\Sigma}\right], \tag{55}$$

where $\mathbf{C} = \sigma^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}$. The form of (55) is intended to be a more intuitive re-writing of (54).

So for a set of Gaussian basis functions with shared scale parameters:

$$\phi_{nm} = \exp\left\{-\sum_{k=1}^{d} \eta_k (x_{mk} - x_{nk})^2\right\}, \tag{56}$$

and we have

$$\frac{\partial \mathcal{L}}{\partial \eta_k} = \sum_{m=1}^{N} \sum_{n=1}^{N} -\mathbf{D}_{nm} \mathbf{\Phi}_{nm} (x_{mk} - x_{nk})^2. \tag{57}$$

The expression (57) can thus be used as the basis for a gradient-based local optimisation over $\boldsymbol{\eta}$. Exactly how this is performed represents the primary implementation difficulty. A joint nonlinear optimisation over $\{\boldsymbol{\alpha}, \boldsymbol{\eta}\}$, using, for example, conjugate gradient methods, is prohibitively slow. Since the α update equation (44) is so effective, we chose to interleave those updates with a few cycles (*e.g.* 1 to 5) of optimisation of $\boldsymbol{\eta}$ using a simple hill-climbing method. The exact quality of results is somewhat dependent on the ratio of the number of $\boldsymbol{\eta}$ to $\boldsymbol{\alpha}$ updates, although in nearly all cases a significant improvement (in generalisation error) is observed.

Clearly it would be more satisfactory to offer a definitive method for optimising $\boldsymbol{\eta}$, and we would expect that there is a better mechanism for doing so than the one we have employed. Nevertheless, the results given for the Friedman #1 task and the 'toy' example of Section 4.3 indicate that, even if simplistically implemented, this is a potentially very powerful extension to the method.

Appendix D. Probabilistic Outputs

D.1 Error Bars in Regression

Note that, as alluded to earlier, care must be exercised when interpreting the error bars given by equation (22) since they are predicated on the prior over functions, which in turn depends on the basis. In the case of the RVM, the basis functions are data-dependent. To see the implication of this, we re-visit the link with Gaussian process models that was developed in Section 5.2.

A Gaussian process model is specified as a (usually zero mean) multivariate Gaussian distribution over observations, *i.e.* $p(\mathbf{t}) = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}_{\mathrm{GP}})$ with

$$\mathbf{C}_{\mathrm{GP}} = \sigma^2 \mathbf{I} + \mathbf{Q}, \tag{58}$$

where $\mathbf{Q}_{mn} = Q(\mathbf{x}_m, \mathbf{x}_n)$ is the *covariance function*, defined over all $\mathbf{x}$-pairs. There are no 'weights' in the model. Instead, predictions (in terms of predictive distributions) are obtained from Bayes' rule using $p(t_*|\mathbf{t}) = p(t_*, \mathbf{t})/p(\mathbf{t})$. The covariance function must give rise to a positive $\mathbf{Q}$, so like the SVM kernel, must adhere to Mercer's condition. The 'Gaussian' (or negative squared exponential) is a common choice: $Q(\mathbf{x}_m, \mathbf{x}_n; \boldsymbol{\theta}) = \theta_1 \exp\left(-\theta_2 \|\mathbf{x}_m - \mathbf{x}_n\|^2\right)$.

Clearly, the RVM is also a Gaussian process, with $\mathbf{C}$ defined as in (34), and indeed we could also make predictions without recourse to any model 'weights'. The corresponding covariance function, read off from (34), for the RVM is:

$$Q_{\mathrm{RVM}}(\mathbf{x}_m, \mathbf{x}_n) = \sum_{i=0}^{N} \alpha_i^{-1} K(\mathbf{x}_m, \mathbf{x}_i) K(\mathbf{x}_n, \mathbf{x}_i), \tag{59}$$

where $K(\mathbf{x}, \mathbf{x}_0) = 1$ denotes the bias. A feature of this prior is that it is data-dependent, in that the prior covariance between any two points $\mathbf{x}_m$ and $\mathbf{x}_n$ depends on all those $\{\mathbf{x}_i\}$

in the training set. This has implications for computing error bars in RVM regression. To see this, consider that we use a Gaussian covariance function, and compute the predictive error bars σ_*^2 *away from the data*, where we would expect them to be significant. For the Gaussian process (Williams, 1999):

$$\begin{aligned}\sigma_*^2 &= \sigma^2 + Q(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^{\mathrm{T}} \mathbf{C}_{\mathrm{GP}}^N \mathbf{k}, \\ &\approx \sigma^2 + \theta_1, \end{aligned} \tag{60}$$

where $(\mathbf{k})_i = K(\mathbf{x}_*, \mathbf{x}_i)$, $\mathbf{C}_{\mathrm{GP}}^N$ is (58) evaluated at the training data points, and we assume the test point to be sufficiently distant from the training data such that all $K(\mathbf{x}_*, \mathbf{x}_i)$, the elements of $\mathbf{k}$, are very small. Using the identical kernel in a relevance vector model gives a predictive variance from (22) of:

$$\begin{aligned}\sigma_*^2 &= \sigma^2 + \mathbf{k}^{\mathrm{T}} \boldsymbol{\Sigma} \mathbf{k}, \\ &\approx \sigma^2, \end{aligned} \tag{61}$$

and thus depends only on the noise variance. There is no contribution from the function prior which may seem odd, but is of course consistent with its specification.

D.2 Posterior Probabilities in Classification

When performing 'classification' of some test example $\mathbf{x}_*$, we would prefer our model to give an estimate of $p(t_* \in \mathcal{C}|\mathbf{x}_*)$, the posterior probability of the example's membership of the class $\mathcal{C}$ given the features $\mathbf{x}_*$. This quantity expresses the uncertainty in the prediction in a principled manner while facilitating the separation of 'inference' and 'decision' (Duda and Hart, 1973). In practical terms, these posterior probability estimates are necessary to correctly adapt to asymmetric misclassification costs (which nearly always apply in real applications) and varying class proportions, as well as allowing the rejection of 'uncertain' test examples if desired.

Importantly, the presented Bayesian classification formulation incorporates the Bernoulli output distribution (equivalent in the log-domain to the 'cross-entropy' error function), which in conjunction with the logistic sigmoid squashing function, enables $\sigma\{y(\mathbf{x})\}$ to be interpreted as a consistent estimate of the posterior probability of class membership. Provided $y(\mathbf{x})$ is sufficiently flexible, in the infinite data limit this estimate becomes exact (see, *e.g.*, Bishop, 1995).

By contrast, for a test point $\mathbf{x}_*$ the SVM outputs a real number which is thresholded to give a 'hard' binary decision as to the class of the target t_*. The absence of posterior probabilities from the SVM is an acknowledged deficiency, and a recently proposed technique for tackling this involves the *a posteriori* fitting of a sigmoid function to the fixed SVM output $y(\mathbf{x})$ (Platt, 2000) to give an approximate probability of the form $\sigma\{A.y(\mathbf{x}) + B\}$. While this approach does at least produce predictive outputs in the range $[0, 1]$, it is important to realise that this does not imply that the output of the sigmoid is necessarily a good approximation of the posterior probability. The success of this post-processing strategy is predicated on the original output $y(\mathbf{x})$ of the SVM, with appropriate shifting and rescaling, being a good approximation to the 'inverse-sigmoid' of the desired posterior probability. This is the quantity $\log\{p(t \in \mathcal{C}_{+1}|\mathbf{x})/p(t \in \mathcal{C}_{-1}|\mathbf{x})\}$, referred to as the 'log-odds'. As a result of the nature of the SVM objective function, $y(\mathbf{x})$ cannot be expected to be a reliable model of this.

The simple example which follows in Figure 10 underlines this. The figure shows the output of trained classifiers on a simple two-class problem in one dimension. Class $\mathcal{C}_{+1}$ is uniform in $[0, 1]$ and class $\mathcal{C}_{-1}$ in $[0.5, 1.5]$, with the overlap implying a minimal Bayes error of 25%. The correct log-odds of $\mathcal{C}_{+1}$ over $\mathcal{C}_{-1}$ is shown in the figure. It should be zero in $[0.5, 1.5]$ and infinite outside this region. The output $y(\mathbf{x})$ of both SVM and RVM (*i.e.* without the sigmoid function in the latter case) is shown for classifiers trained on a generous quantity of examples (1000) using a Gaussian kernel of width 0.1. Note that both models are optimal in the generalisation sense, since the decision boundary may lie anywhere in $[0.5, 1]$.

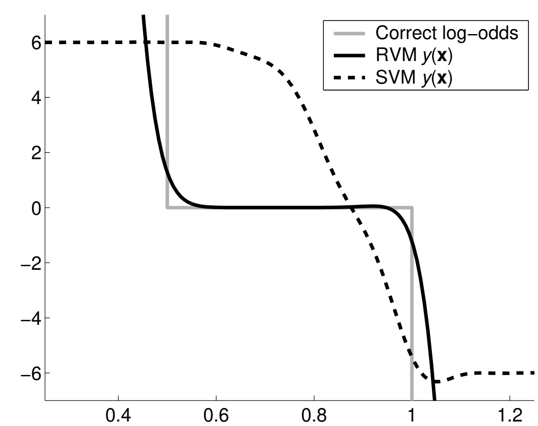

Figure 10: Output of an RVM and SVM trained on overlapping, uniformly distributed data, along with the true log-odds of membership of $\mathcal{C}_{+1}$ over $\mathcal{C}_{-1}$. The SVM output has been rescaled without prejudice for ease of comparison with the RVM.

The key feature of Figure 10 is that while the RVM offers a reasonable approximation, the SVM output is highly *un*representative of the true log-odds. It should be evident that no setting of the sigmoid parameters A and B can ever correct for this. This is exemplified in the region $x \in [0.5, 1]$, in the vicinity of the decision boundary where posterior accuracy is arguably most important, and where the log-odds should be constant. Practically, we note that if this were a real application with asymmetric misclassification losses, for example medical diagnosis or insurance risk assessment, then the absence of, or lack of accuracy in, posterior estimates could be very costly.

Appendix E. Benchmark Data Sets

E.1 Regression

Noisy sinc. Data was generated from the function $y(x) = \sin(x)/x$ at 100 equally-spaced x-values in $[-10, 10]$ with added noise from two distributions: first, Gaussian of standard deviation 0.1 and second, uniform in $[-0.1, 0.1]$. In both cases, results were averaged over 100 random instantiations of the noise, with the error being measured to the true function over a 1000-example test set of equally spaced samples in $[-10, 10]$.

Friedman functions. Synthetic functions taken from Friedman (1991):

$$y_{\#1}(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - 1/2)^2 + 10x_4 + 5x_5 + \sum_{k=6}^{10} 0.x_k, \tag{62}$$

$$y_{\#2}(\mathbf{x}) = [x_1^2 + (x_2 x_3 - 1/x_2 x_4)^2]^{1/2}, \tag{63}$$

$$y_{\#3}(\mathbf{x}) = \tan^{-1}\left[\frac{x_2 x_3 - 1/x_2 x_4}{x_1}\right]. \tag{64}$$

For #1, inputs were generated at random from the 10-dimensional unit hypercube (note that input variables x_6 through x_{10} do not contribute to the function) and Gaussian noise of unit standard deviation added to $y(\mathbf{x})$. For #2 and #3, random input samples were generated uniformly in the intervals $x_1 \in [0, 100]$, $x_2 \in [40\pi, 560\pi]$, $x_3 \in [0, 1]$ and $x_4 \in [1, 11]$. Additive Gaussian noise was generated on the output of standard deviation one-third of that of the signal $y(\mathbf{x})$. For all Friedman functions, average results for 100 repetitions were quoted, where 240 randomly generated training examples were utilised, along with 1000 noise-free test examples.

Boston housing. This popular regression benchmark data set was obtained from the *StatLib* archive at `http://lib.stat.cmu.edu/datasets/boston`. It comprises 506 examples with 14 variables, and results were given for the task of predicting the median house value from the remaining 13 variables. Averages were given over 100 repetitions, where 481 randomly chosen examples were used for training and the remaining 25 used for testing.

E.2 Classification

Pima Diabetes. This data was utilised as an example set by Ripley (1996), and re-used by Williams and Barber (1998). We used the single split of the data into 200 training and 332 test examples available at `http://www.stats.ox.ac.uk/pub/PRNN/`.

U.S.P.S. The U.S. Postal Service database of 16×16 images of digits '0' through '9' is that commonly utilised as a benchmark for SVM-related techniques, *e.g.* as used by Schölkopf et al. (1999b). The data utilised here was obtained from those authors, and comprises 7291 training examples along with a 2007-example test set.

Banana, Breast Cancer, Titanic, Waveform, German, Image. All these data sets were taken from the collection recently utilised by Rätsch et al. (2001). More details concerning the original provenance of the sets are available in a highly comprehensive online repository at `http://ida.first.gmd.de/~raetsch/`. A total of 100 training/test splits are provided by those authors: our results show averages over the first 10 of those.

References

J. O. Berger. *Statistical decision theory and Bayesian analysis.* Springer, second edition, 1985.

C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

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.

B. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 71–77, Bari, Italy, 1996. Morgan Kaufmann.

C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381. MIT Press, 1997.

S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, 1995.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley, 1973.

J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–141, 1991.

Y. Grandvalet. Least absolute shrinkage is equivalent to quadratic penalisation. In L. Niklasson, M. Bodén, and T. Ziemske, editors, *Proceedings of the Eighth International Conference on Artificial Neural Networks (ICANN98)*, pages 201–206. Springer, 1998.

T. S. Jaakkola and M. I. Jordan. Bayesian logistic regression: a variational approach. In D. Madigan and P. Smyth, editors, *Proceedings of the 1997 Conference on Artificial Intelligence and Statistics*, Ft Lauderdale, FL, 1997.

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

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

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

D. J. C. MacKay. Bayesian methods for backpropagation networks. In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks III*, chapter 6, pages 211–254. Springer, 1994.

D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–165. Springer, 1998.

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

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Probability and Mathematical Statistics. Academic Press, 1979.

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.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.

J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2000.

C. E. Rasmussen. *Evaluation of Gaussian processes and other methods for non-linear regression*. PhD thesis, Graduate Department of Computer Science, University of Toronto, 1996.

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

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

B. Schölkopf. The kernel trick for distances. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1999a.

B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999b.

M. Seeger. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 603–609. MIT Press, 2000.

A. J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.

A. J. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN99)*, pages 575–580, 1999.

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.

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.

M. E. Tipping. Sparse kernel principal component analysis. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

V. N. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

V. N. Vapnik, S. E. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation and signal processing. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*. MIT Press, 1997.

C. K. I. Williams. Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 599–621. MIT Press, 1999.

C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

P. M. Williams. Bayesian regularisation and pruning using a Laplace prior. *Neural Computation*, 7(1):117–143, 1995.