Sparse Bayesian Learning and Relevance Vector Machine

Sumit Kumar Pandey(180797), Wasif Jawad Hussain(180882) and Divij Jain(180251)

Abstract

The paper presents a Bayesian framework to obtain sparse solutions for regression and classification problems. The paper uses a particular specialisation of the conventional Support Vector Machine(SVM) which known as the ”Relevance Vector Machine” (RVM).The original SVM suffers from a number of disadvantages. However the RVM suffers from none of those disadvantages. At the same time the probabilistic Bayesian framework provides a number of additional advantages. The framework uses lesser basis functions and at the same time gives probabilistic predictions. The paper also illustrates some examples of the RVM application along with some comparative benchmarks.

Index Terms

Support Vector Machines, Linear Regression, Classification, Bayes Theory, Kernels, Inner Product Spaces, Posterior, Likelihood, Prior, Conjugate Distributions

I. Introduction

In the supervised learning problem we are given a training data which comprises of a set of examples of input vectors $\{\mathbf{x_n}\}_{n=1}^{N}$ and the corresponding targets $\{t_n\}_{n=1}^{N}$. Our goal is to come up with a model which captures the relationship between the target and the input vectors and helps us in making predictions about the target $\{t_{new}\}$ for a new value of input $\{\mathbf{x_{new}}\}$. The predictions are modelled as a certain function of the input vectors. The task of inferring this function is done during the training process.

These functions $\mathbf{y}(\mathbf{x})$, can be expressed in general form as mentioned below.

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

The authors are with the Department of Electrical Engineering, IIT Kanpur, Kanpur 208016, India. email: {sumitkrp, wasif, jaind}@iitk.ac.in.

Where $y(\mathbf{x};\mathbf{w})$ is a weighted sum of M, generally non-linear and fixed basis functions $\phi(\mathbf{x})$ given as

$$\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))^T \tag{2}$$

Tipping presented a Bayesian probabilistic framework [1] for training models like (1). The key feature of this method is that, most of the parameters w_i are set to 0 during the training process. As a result only the most relevant parameters remain. Equations of the form (1) represent wide range of models but the paper laid more emphasis on a specific type denoted as the *Relevance Vector Machine*(RVM) whose implementation is similar to that of *Support Vector Machine*(SVM).

The *SVM* model makes predictions as described below

$$y(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x};\mathbf{x_i}) + w_0 \tag{3}$$

where $K(\mathbf{x};\mathbf{x_i})$ is a *kernel* function. During classification type problems, SVM attempts to minimise the error while also trying to maximise the margin between the classes.This is greatly helpful in avoiding over-fitting, making good generalisation and most importantly it results in a sparse model dependent only on a subset of kernel functions(those associated with support vectors).

In spite of these advantages there are some drawbacks of the SVM methodology, listed as-

1) Despite being relatively sparse, SVMs make liberal use of kernel functions. Since the number of basis functions grows linearly with the training size, it often brings the necessity for post-processing.
2) Ideally we desire to estimate the conditional distribution in order to capture the uncertainty in our predictions. However, due to the non-probabilistic nature of SVMs, this uncertainty is not captured.
3) The necessity to estimate the margin trade-off parameter *C*, brings us to the requirement of a cross-validation procedure which is computationally expensive.
4) The kernel function $K(\mathbf{x};\mathbf{x_i})$ must satisfy Mercer's condition

The relevance vector machine (RVM) is a Bayesian implementation of (3), it does not suffer from the above limitations. We have one prior associated with each weight and it is governed by a set of hyper-parameters. Sparsity is obtained because posterior distributions of many of the weights are peaked around zero. The training vectors associated with the remaining non-zero weights are termed as *relevance vectors*. The most enthralling feature of RVM is that even though its performance is similar to an SVM, it generally uses significantly lesser kernel functions.

II. Sparse Bayesian Learning for Regression

A. Model Specification

We are given a data set of input-target pair $\{\mathbf{x_n}, \mathbf{t_n}\}_{n=1}^{N}$. We consider the target functions to be scalar valued. We formulate a probabilistic model assuming the targets to be samples from model with additive noise. The likelihood of the dataset can be written as

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

where

$$\mathbf{t} = (t_1, t_2, \ldots, t_n)^T$$

$$\mathbf{w} = (w_1, w_2, \ldots, w_n)^T$$

$$\boldsymbol{\phi}_{N\times N+1} = (\phi(\mathbf{x_1}), \phi(\mathbf{x_2}), \ldots, \phi(\mathbf{x_M}))^T$$

$$\boldsymbol{\phi}(\mathbf{x_n}) = [1, K(\mathbf{x_n}, \mathbf{x_1}), \ldots, K(\mathbf{x_n}, \mathbf{x_N})]^T$$

Since maximum likelihood estimation can lead to over-fitting we define the following prior over the weights.

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

here $\boldsymbol{\alpha}$ is a *N+1* dimensional vector of hyper-parameters. We then set out to define hyper-prior on the remaining parameters as follows

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

$$p(\beta) = Gamma(\beta|c, d);\ \ \beta = \sigma^{-2}$$

B. Inference

We apply Bayes rule to obtain the posterior over all the unknown given data written as

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)}{p(\mathbf{t})} \tag{6}$$

We can use this obtained posterior to calculate the predictive distribution given as

$$p(t_*|\mathbf{t}) = \int p(\mathbf{t}_*|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)|\mathbf{t})d\mathbf{w}d\boldsymbol{\alpha}d\sigma^2 \tag{7}$$

Computing posterior in (6) is not possible due to the presence of the intractable integral in the denominator. So instead we decompose it as follows

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

Since the normalising integral of (8) is a convolution of two Gaussians we can analytically calculate the posterior distribution.

The posterior over weights is given as follows

$$
\begin{aligned}
p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha},\sigma^2) &= \frac{p(\mathbf{t}|\mathbf{w},\sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2)} \\
&= (2\pi)^{-\frac{N+1}{2}}|\Sigma|^{-1/2}\exp\left(\frac{-1}{2}(\mathbf{w}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{w}-\boldsymbol{\mu})\right) \qquad (9) \\
\mathbf{A} &= diag(\alpha_0,\alpha_1,\dots,\alpha_N); \boldsymbol{\Sigma} = (\sigma^{-2}\boldsymbol{\phi}^{\boldsymbol{T}}\boldsymbol{\phi}+\mathbf{A})^{-1}; \boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\phi}^{\boldsymbol{T}}\mathbf{t}
\end{aligned}
$$

We approximate the hyperparameter posterior $p(\boldsymbol{\alpha},\sigma^2|\mathbf{t})$ by a delta function at its mode(most probable values). Our belief is that the functions generated from the point estimate are very good approximation from those obtained by sampling from posterior. Also for predictive purposes rather than requiring the exact expression for $p(\boldsymbol{\alpha},\sigma^2|\mathbf{t})$ we only need the integral

$$\int p(t_*|\boldsymbol{\alpha},\sigma^2)p(\boldsymbol{\alpha},\sigma^2|\mathbf{t})d\boldsymbol{\alpha}d\sigma^2 \sim \int p(t_*|\boldsymbol{\alpha},\sigma^2)\delta(\boldsymbol{\alpha},\sigma^2|\mathbf{t})d\boldsymbol{\alpha}d\sigma^2 \qquad (10)$$

which is again also approximated by the mode approximation.

Summarising Relevance Vector "Learning" has thus become the search for hyper-parameter posterior mode. *i.e* maximisation of $p(\boldsymbol{\alpha},\sigma^2|\mathbf{t})$. For the case of uniform hyper-priors this boils down to maximising the term $p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2)$, which is computable and given by

$$
\begin{aligned}
p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2) &= \int p(\mathbf{t}|\mathbf{w},\sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})d\mathbf{w} \\
&= (2\pi)^{-N/2}|\sigma^2\mathbf{I}+\boldsymbol{\phi}\mathbf{A}^{\mathbf{-1}}\boldsymbol{\phi}^{\boldsymbol{T}}|^{-1/2}exp\left(\frac{-1}{2}\mathbf{t}^{\mathbf{T}}(\sigma^2\mathbf{I}+\boldsymbol{\phi}\mathbf{A}^{\mathbf{-1}}\boldsymbol{\phi}^{\boldsymbol{T}})^{-1}\mathbf{t}\right)
\end{aligned} \qquad (11)
$$

This quantity is known as the marginal likelihood and its maximisation comes under type-II maximum likelihood method.

C. Optimising the Hyperparameters

Values of $\boldsymbol{\alpha}$ and σ^2 which maximise (11) can't be obtained in closed form and are obtained by various numerical techniques.

For $\boldsymbol{\alpha}$ we differentiate (11) and equate it with zero. We get the following result.

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2} \qquad (12)$$

here μ_i is the i^{th} posterior mean from (9). γ_i is a measure of how well-determined it's corresponding parameter w_i is by the data and it is given as

$$\gamma_i \equiv 1 - \alpha_i \Sigma_{ii} \ ; \ \gamma_i \epsilon [0, 1]$$

here Σ_{ii} is the i^{th} diagonal element of the posterior weight co-variance from (9).

For the noise variance σ^2, differentiation of (11) leads us to the estimate:

$$(\sigma^2)^{new} = \frac{||\mathbf{t} - \boldsymbol{\phi\mu}||}{N - \Sigma_{ii}\gamma_i} \text{ ; N: number of data examples} \tag{13}$$

Learning algorithm proceeds by repeated application of (12) and (13) and subsequently updating posterior statistics $\mathbf{\Sigma}$ and $\boldsymbol{\mu}$ from (9) until some suitable convergence criteria is realised.

D. Making Predictions

After the convergence of the hyper-parameter estimation procedure, we can make new predictions by using the posterior distribution over weights, conditioned on the maximising values $\boldsymbol{\alpha_{MP}}$ and σ^2_{MP}. We can then compute the predictive distribution for a new datum $\mathbf{x}_*$ from (10) as follows

$$p(t_*|\mathbf{t}, \boldsymbol{\alpha_{MP}}, \sigma^2_{MP}) = \int p(t_*|\mathbf{w}, \sigma^2_{MP}) p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha_{MP}}, \sigma^2_{MP}) d\mathbf{w} \tag{14}$$

with

$$\begin{aligned} y_* &= \boldsymbol{\mu^T \phi(x_*)} \\ \sigma^2_* &= \sigma^2_{MP} + \boldsymbol{\phi(x_*)^T \Sigma \phi(x_*)} \end{aligned} \tag{15}$$

We infer the following two remarks from our obtained results:

- Predictive mean is intuitively the basis functions weighted by the posterior mean weights, many of which are typically zero.
- The predictive variance comprises of sum of two variance components. One from the estimated noise from the data and the other that arises due to uncertainty in the prediction of the weights.

III. Sparse Bayesian Classification

Relevance vector classification follows same approach as of regression which we have discussed in the previous section. In classification, where it is desired to predict the posterior probability of class membership given the input $\mathbf{x}$, we generalise the linear model by applying the logistic sigmoid function $\sigma(y) = 1/(1+e^{-y})$ to $y(\mathbf{x})$, so our likelihood will be written as:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma\{y(\mathbf{x_n};\mathbf{w})\}^{t_n}[1-\sigma\{y(\mathbf{x_n};\mathbf{w})\}]^{1-t_n} \tag{16}$$

But unlike the regression case we can't integrate out the weights to obtain closed form expressions for either the marginal likelihood $P(\mathbf{t}|\mathbf{w})$ or the weight posterior $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$, so we apply an iterative procedure [2], based on Laplace's method:

1) For the current, fixed values of $\boldsymbol{\alpha}$ we find the most probable weights $\mathbf{w_{MP}}$ (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 equal to finding the maximum, over $\mathbf{w}$, of:

$$\log\{P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})\} = \sum_{n=1}^{N}[t_n \log y_n + (1-t_n)\log(1-y_n)] - \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} \tag{17}$$

with $y_n = \sigma\{y(\mathbf{x_n};\mathbf{w})\}$. This is a standard optimization of a regularised logistic model and we use efficient iteratively-reweighted least-squares algorithm to find $\mathbf{w_{MP}}$.

2) We compute the Hessian at $\mathbf{w_{MP}}$ by differentiating (17) twice to give:

$$\nabla_{\mathbf{w}}\nabla_{\mathbf{w}} \log p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})\big|_{\mathbf{w_{MP}}} = -\left(\boldsymbol{\phi}^{\boldsymbol{T}}\mathbf{B}\boldsymbol{\phi} + \mathbf{A}\right) \tag{18}$$

where $\mathbf{B} = diag(\beta_1, \beta_2, ..., \beta_N)$ is a diagonal matrix with $\beta_n = \sigma\{y(\mathbf{x_n})\}[1-\sigma\{y(\mathbf{x_n})\}]$. This is then negated and inverted to give the covariance $\boldsymbol{\Sigma}$ for a Gaussian approximation to the posterior over weights centered at $\mathbf{w_{MP}}$. At the mode of $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})$, using (18) and the also the fact that $\nabla_{\mathbf{w}} \log p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha})\big|_{\mathbf{w_{MP}}} = 0$ we can write:

$$\boldsymbol{\Sigma} = \left(\boldsymbol{\phi}^{\boldsymbol{T}}\mathbf{B}\boldsymbol{\phi} + \mathbf{A}\right)^{-1} \tag{19}$$

$$\mathbf{w_{MP}} = \boldsymbol{\Sigma}\boldsymbol{\phi}^{\boldsymbol{T}}\mathbf{B}\mathbf{t} \tag{20}$$

3) Using the statistics $\boldsymbol{\Sigma}$ and $\mathbf{w_{MP}}$ (in place of $\boldsymbol{\mu}$) of the Gaussian approximation, the hyper-parameters $\boldsymbol{\alpha}$ are updated using (12). There is no term of 'noise' variance σ^2 here.

This procedure is repeated until convergence criteria is satisfied. Compared with (9), it can be seen that the Laplace approximation effectively maps the classification problem to a regression one with data-dependent noise, with the inverse noise variance for ϵ_n given by β_n. In the Bayesian treatment of multi-layer 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 $\mathbf{w_{MP}}$ can often be unrepresentative of the overall posterior mass. But here in this linearly-parametrised model (RVM), we know that $p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha})$ is log-concave (as the Hessian is negative-definite everywhere), which gives us considerably more confidence in the Gaussian approximation. For polychotomous classification, where the number of classes is greater than two, the likelihood is written in the form:

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \sigma\{y_k(\mathbf{x_n}; \mathbf{w_k})\} \tag{21}$$

Here the classifier has multiple outputs $y_k(\mathbf{x}; \mathbf{w_k})$, each with it's own parameter vector $\mathbf{w_k}$ and associated hyper-parameters $\boldsymbol{\alpha}_k$. The modified Hessian is computed from [3], and inference proceeds as shown above.

IV. Relevance Vector Examples

The paper applies both regression and classification on synthetic data sets.

A. Relevance Vector Regression: the 'sinc' function

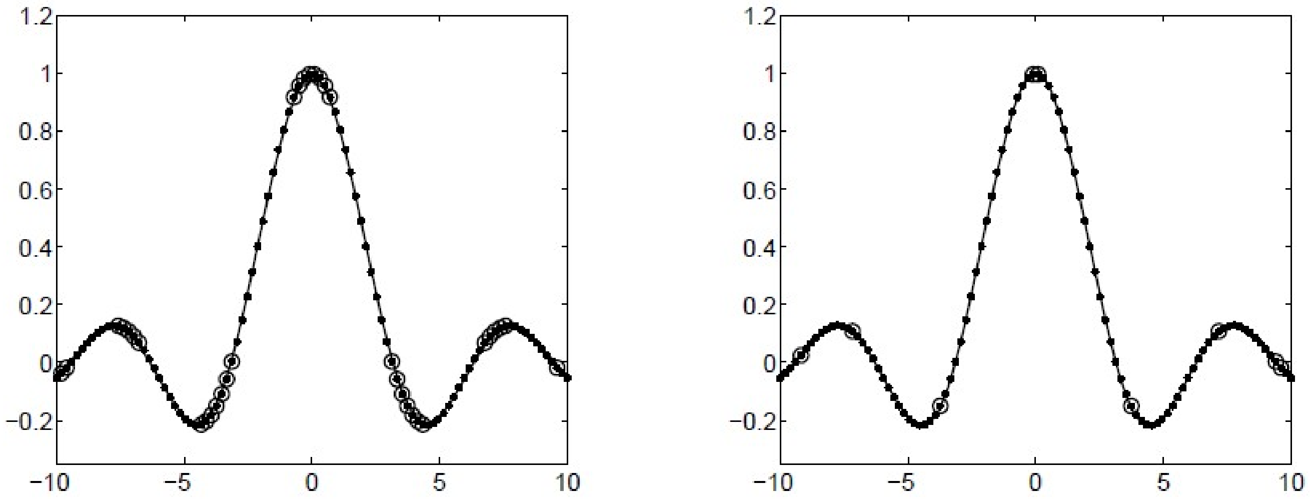

Fig. 1: Support (left) and relevance (right) vector approximations to sinc(x) from 100 noise-free examples using 'linear spline' basis functions. The estimated functions are drawn as solid lines with support/relevance vectors shown circled

The function sinc(x) = sin(x)/x is used to demonstrate support vector regression. We have a tube of $\pm\epsilon$ region inside which the errors are not penalized. The support vectors lie on the edge of, or outside the tubular region.

Now by using a univariate 'linear spline' kernel:

$$K(x_m, x_n) = 1 + x_m x_n + x_m x_n min(x_m, x_n) - \frac{x_m + x_n}{2} min(x_m, x_n)^2 + \frac{min(x_m, x_n)^3}{3} \quad (22)$$

and taking ϵ = 0.01, we approximate sinc(x) using 100 uniformly-spaced noise-free samples in [-10; 10] which uses 36 support vectors as illustrated in *Fig. 1(left)* [1]. In the case of RVM, we model the same data using the same kernel(22). For the purpose of comparison in SVM example, we model the 'sinc' function with a relevance vector machine. However, we set the noise-variance to be 0.01^2 and we re-estimate α alone. Using the above value of σ, the RVM approximator uses only relevance vectors and is plotted in *Fig. 1(right)* [1]. The maximum error for RVM is 0.007 and that for SVM is 0.01. Thus RVMs have advantages both in terms of higher accuracy and greater sparsity in comparison to SVMs. We use the same kernel even when a uniform noise is added to the targets in the interval [-0.2;0.2] as illustrated in *Fig. 1* [1]. In this case too RVM performs better than SVM as it uses lesser number of relevance vectors. The RMS deviation in case of RVM is just 0.0245 in comparison to 0.0291 for a SVM. Also in case of RVMs, the parameters $\boldsymbol{\alpha}$ and σ^2 are automatically computed in the training process

B. Relevance Vector Classification: Ripley's synthetic data

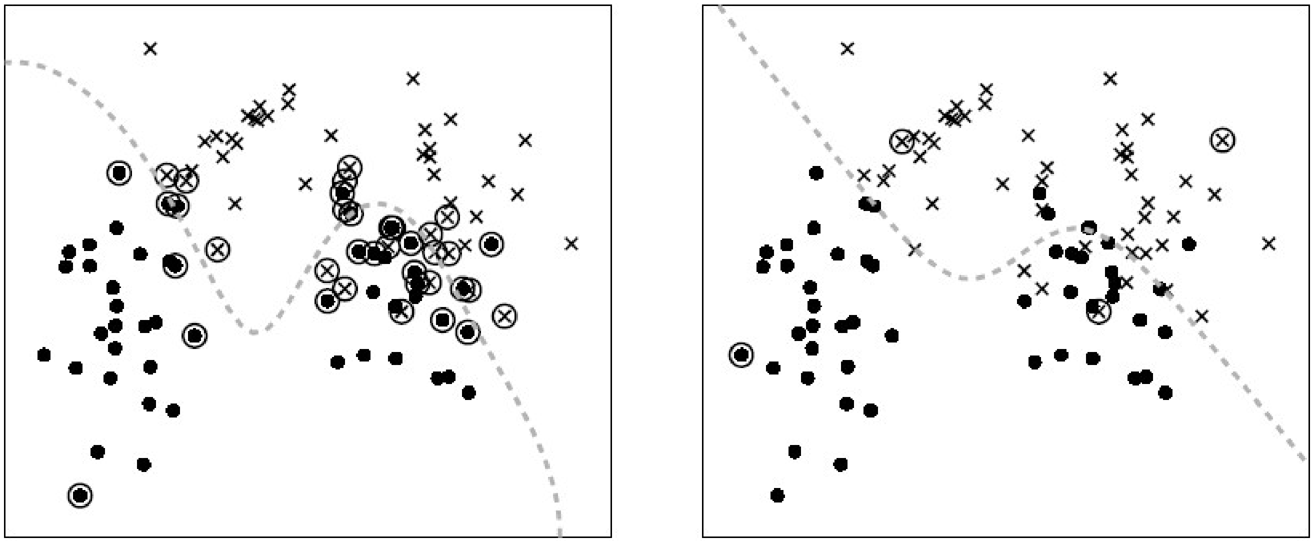

Fig. 2: Analysis of performance of SVM(on the left) and RVM(on the right) on 100 examples from Ripley's GMM dataset. Dashed line is the decision boundary and circled points are the respective support/relevance vectors.

We have class 1(denoted by $'\times'$) and class 2(denoted by $'\bullet'$) generated artificially(refer *Fig 2*).

We use the following Gaussian kernel:

$$K(x_m, x_n) = exp(-r^{-2}\|\mathbf{x_m} - \mathbf{x_n}\|^2) \tag{23}$$

where 'r' is the 'width' parameter (taken to be 0.5 for this case). The test error of RVM is superior to that of a SVM (9.3% error vs 10.6% error) over the chosen test set. The RVM uses only 4 kernel functions in comparison to 38 kernel functions used by SVM which greatly reduces the complexity of the RVM classifier. Another interesting feature of RVM is that, unlike SVM, the relevance vectors are at some distance from the decision boundary.

C. Extensions

Tipping uses another synthetic example to illustrate the following two favourable features of the Sparse Bayesian approach: -

1) It makes use of arbitrary basis functions.
2) It can directly optimize the kernel parameters.

It is important to determine the kernel parameters both in SVMs and RVMs. Now based on 100 examples added with Gaussian noise of standard deviation 0.1, we have to determine the following 2 dimensional function: -

$$y(x_1, x_2) = sinc(x_1) + 0.1x_2$$

Since the function is linear in x_2, it is not good to model the above function using the superposition of non-linear functions. Also for the non-linear function sinc(x_1), the term $0.1x_2$ simply acts as noise. These factors make the SVM learning process very difficult. We bring in two modifications to improve the above result.

1) We append 2 extra column to the design matrix ϕ containing x_1 and x_2 and also introduce the corresponding weights and hyper-parameters. We also introduce 3 additional quadratic terms x_1^2 , x_2^2 and x_1x_2.
2) We directly optimise the marginal likelihood w.r.t the kernel parameters. We use the following kernel function with parameters η_1 and η_2: -

$$K(x_m, x_n) = exp(-\eta_1(x_{m1} - x_{n1})^2 - \eta_2(x_{m2} - x_{n2})^2) \tag{24}$$

The above two modifications bring in much more accurate RVM approximating function. The RMS error of RVM is much lower than that of SVM (0.0053 vs 0.0194). With only 8 basis functions, the RVM is much more sparse than SVM(75 basis functions). The obtained values of η_1 and η_2 do not lead to over-fitting. No cross-validation required and we get all the additional parameters directly from the training set. These results are very interesting, yet there are certain limitations of it.

1) Since the two-staged training process is interleaved, it becomes difficult to explain as to how to combine optimization over $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$.
2) The optimization process is computationally expensive.

D. Benchmark Comparisons

The paper summarises the performance of RVM on both classification and regression. The number of training examples(N) and the number of input variables(d) is known for each data set. The RVM statistics were normalised by those of the SVM. A Gaussian kernel was used and the input scale parameter chosen by 5-fold cross-validation. The following observations were made: -

1) The errors obtained for RVM is lesser than that obtained in SVM in case of regression but are very close in case of classification task.
2) The number of support/relevance vectors in both classification and regression is much lesser for RVMs in comparison to SVMs.

REFERENCES

[1] Michael E. Tipping *Sparse Bayesian Learning and the Relevance Vector Machine*. Journal of Machine Learning Research 1 (2001) 211-244

[2] D. J. C Mackay *The evidence framework applied to classification networks*. Neural Computation, 4(5):720-736, 1992b

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