Sparse Modelling with Relevance Vector Machines

Group 3

Athira Athira athira@kth.se
Alexandros Ferles ferles@kth.se
Erifili Ichtiaroglou erifili@kth.se
George Zervakis zervakis@kth.se

January 21, 2018

Abstract

Support Vector Machines (SVM) are a state of the art method in supervised Machine Learning, making use of model weights and kernel functions. Since they use a subset of kernels, they are considered *sparse* models, and they have good generalisation properties. However, a typical SVM procedure does not handle uncertainty over the results, as no probabilistic computation is involved. The *Relevance Vector Machine*, introduced in Tipping's work, is a sparse model which follows similar techniques with the SVM, in addition with handling uncertainty under a full Bayesian approach. We present a re-implementation of the original RVM, dealing with both the Relevance Vector Regression and Classification problems. We employ various techniques such as the *Expectation-Maximization* and the *Iterative Reweighted Least Squares* algorithms for estimating coefficients that are used in parts of both problems. We evaluate our approach in toy problems that were introduced in the original work. We obtain results of comparative accuracy in both the error and the relevance vector space.

1 Introduction

Tipping's paper [1] introduces the "relevance vector machine" (RVM) models, which is suggested as a probabilistic approach of the "support vector machine" SVM models[2].

In the SVM approach of the supervised learning, either regression or classification, we are given a training test with the input vectors x_n and their corresponding targets t_n and we are trying to learn the correlation between the targets and the inputs, so that we can predict correctly the target of an unseen input point x. To do so, SVM uses the equation

$$y(\mathbf{x}) = \Sigma_{n=1}^{N} w_n K(\mathbf{x}, \mathbf{x_n}) + w_0 \tag{1}$$

where w_n are the weights specified by the model and K(.,.) is the kernel function (estimates how "similar" the input points are). SVM provides a sparse model, which prevents over-fitting since only some of the kernels functions are used to predict the target of a new data point, those that are proven by the model to be the most significant ones.

However SVMs appear to have some important disadvantages:

- Since the kernel functions are dependent on the input points, the size of the kernel functions increases with the size of the input points, which is not computationally efficient.
- The predictions of the model are not probabilistic. Instead SVM provides "hard" targets for the input points and doesn't capture the uncertainty in the prediction.
- The trade-off parameter "C" for classification and the insensitivity parameter ϵ have to also be tuned, in order to make the model more general, compensate for not capturing the uncertainty in the predictions and further avoid overfitting.

To overcome these shortcuts of SVMs, Tipping proposed a probabilistic framework, where a prior over the weights is introduced, governed by some hyper-parameters (one for each weight)[1]. The hyper-parameters are estimated iteratively from the data and the ones with values closed to infinity are pruned, as they correspond to weights peaked around zero. Thus, the sparsity of the model is achieved and the non-zero remaining vectors, "the relevance vectors", are used to predict the targets of the new input points. The main advantages of the RVMs over SVMs is that RVMs do not exhibit any of the limitations above, since they are probabilistic and thus the uncertainty in the predictions is being taken into consideration and also they use much fewer kernel functions in comparison to SVMs.

1.1 Literature Review

An RVM algorithm with a variational approach (patented [3]) was developed where the posterior distribution is approximated as the product of distribution of hyperparameters, distribution of weights and distribution of predetermined additional parameters[4]. This uses a predetermined reference criterion.

A faster implementation of the RVM was introduced in the paper [5]. The acceleration of the optimization process is accomplished by a 'principled and efficient sequential addition and deletion of candidate basis functions'. A paper [6] published in 2005 points out that the sparsity of the RVM arises from uncertainty in the predictions and suggest using augmentation to solve the problem of uncertainty increasing in proportion to the distance from training samples. Schmolck and Everson [7] points out that the since there is no explicit prior defined over the variances of weight, the degree of sparsity is highly dependent on the kernel function used (type of the kernel and the kernel functions). They suggest incorporating a flexible 'smoothness prior' dependent on the noise to the RVM and demonstrates that it results in improved fit, sparsity and computational performance.

1.2 Scope and Objective

The scope of this project was to gain a deeper insight to the RVMs model by implementing the methods proposed in the paper "Sparse Bayesian learning and the relevance vector machine" by Tipping, Michael E. The suggested methods concern both the regression and classification task and we had to test our implementation, by performing the same evaluation test as in [1]. In section 2 we explain the implementation method we followed and we present the results in section 3. Finally in section 4 we reason about our results and compare them to the ones from Tippings paper.

2 Relevance Vector Regression

2.1 The Model

The task of Regression, similarly with the Support Vector Machine method, is based upon a linear model, where our target function $y(x)$ is a linear combination of some basis(kernel) functions $\phi_i(x)$ as shown in the following equation:

$$y(\mathbf{x}) = \sum_{n=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) \tag{2}$$

Suppose that we have a dataset of input-output pairs $(x_n, t_n)_{n=1}^{N}$ then $\mathbf{w} = (w_0, ..., w_N)$ is the weight vector corresponding to the input data and Φ is the *design* matrix of N x *(N+1)* dimensions with $\Phi_{nm} = K(x_n, x_{m-1})$ and $\Phi_{n1} = 1$.

Assuming a Gaussian distribution for the likelihood and making use of the above equation, we define the likelihood of the dataset as:

$$p(t|w, \sigma^2) = (2\pi\sigma^2)^{-\frac{N}{2}} exp[-\frac{1}{2\sigma^2}||t - \Phi w||^2] \tag{3}$$

Unfortunately, the Maximum-Likelihood-Estimation of the parameters w, σ^2 is not a well established method, because it will, in general, lead to overfitting.

To deal with this issue, we introduce a set of *N+1* hyperparameters α each one associated independently with a weight. This modification, corresponds to a Gaussian prior over the weights. It is important to mention that this is exactly the reason why the sparse characteristics of the RVM are obtained.

Next, from the Baye's Rule, we calculate the posterior over the weights. $p(w|t, \alpha, \sigma^2) \sim \mathcal{N}(\mu, \Sigma)$ which we be a Gaussian with its second order statistics given by:

$$\Sigma = (\Phi^\top B \Phi + A)^{-1}$$
$$\mu = \Sigma \Phi^\top B t \tag{4}$$

where A is a *N+1* x *N+1* diagonal matrix with all **a** in its diagonal and $B = \sigma^2 \mathcal{I}_N$.

The procedure follows by marginalising out the weights and calculating the *evidence* for the hyperparameters, that is:

$$p(t|\alpha, \sigma^2) = (2\pi)^{-fracN2}|B^{-1} + \Phi A^{-1}\Phi^\top|^{-frac12} exp[-\frac{1}{2}t^\top(B^{-1} + \Phi A^{-1}\Phi^\top)^{-1}t] \quad (5)$$

In a fully Bayesian treatment, we should integrate out the hyperparameters **a** and σ^2 but unfortunately, such approach will lead to a non closed-form for the marginalisation. Alternatively, we use the *type II maximum likelihood method* to optimise the marginal likelihood with respect to the hyperparameters and hence, we make predictions based on these optimum values.

2.2 Parameter Estimation

The optimisation of the hyperparameters **a**, σ^2 can be achieved with two iterative methods. The former is an Expectation Maximisation (EM) approach, which updates the **a** using this rule:

$$\alpha_i^{new} = \frac{1}{< w_i^2 >_{p(w|t,\alpha,\sigma^2)}} = \frac{1}{\Sigma_{ii} + \mu_i^2} \quad (6)$$

The latter method, which it is considered to converge faster, performs a differentiation of (5) and produces the following re-estimation rule:

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2} \quad (7)$$

where $\gamma_i = 1 - \alpha_i \Sigma_{ii}$

Both methods conclude to the same estimate for the noise variance, which is obtained by:

$$(\sigma^2)^{new} = ||t - \Phi\mu||^2/(N - \sum_i \gamma_i) \quad (8)$$

Finally, during the training phase, many of the α go to infinity, which means that the posterior over the weights becomes infinitely peaked at zero. Thus, we can actually omit the kernel functions associated with these α.

3 Relevance Vector Classification

3.1 The Model

RVM classification assumes a generalized linear model and a sigmoid function is applied to the linear model output y_x to obtain categorical (here binary) outputs of classification. This allows the classification task to be treated as a logistic regression task. The model is summarized in equation (9).

$$\begin{gathered}\text{Bernoulli likelihood: } P(\mathbf{t}|\mathbf{x}) = \prod_{n+1}^{N} \sigma\{y(x_n)\}^{t_n}\{1 - \sigma y(x_n)\}^{1-t_n} \\ \text{sigmoid function: } \sigma(y(x)) = \frac{1}{1+e^{-y}}\end{gathered} \quad (9)$$

Here the likelihood for the model is in the form of a Bernoulli distribution[8]. The corresponding maximum likelihood does not have a quadratic form and hence the need to resort to iterative methods[1].

3.2 Parameter Estimation

The iterative procedure for parameter estimation is outlined in this section. Similar to RVM regression, each iteration aims to improve the estimate of the hyper-parameters (prior over weights) and thus find the relevance vectors. However, unlike the regression case, in classification, noise variance σ^2 is not accounted for[1].

The iterative procedure alternates between two steps: estimating the posterior mode W_{MP} from the current estimate of hyper-parameters (α) and updating the hyper-parameters using the new estimate of W_{MP}. Hence the algorithm needs an initialization of the weights (initialized to 0) and alpha hyper-parameters (initialized to 1). The former step is equivalent to solving a logistic regression problem. Among the many possible optimization techniques, the method suggested in the reference paper [1][9]. This optimization technique is called iterative re-weighted least squares algorithm (IRLS)[8] and has been described in the next section.

Iterative Re-weighted Least Squares (IRLS) Algorithm

IRLS is an optimization algorithm for solving L_p approximation problem by an iterative approach. It has been proved to be very efficient in finding maximum likelihood estimates for generalized linear models[8].

In the current work, we use IRLS to estimate the posterior mode $\mathbf{w}_{MP}$ iteratively. For a logistic regression model, the sigmoid function is non-linear and the maximum likelihood formulation does not have a quadratic form. Hence a closed form solution does not exist for the logistic regression model. IRLS algorithm finds an optimal L_p approximation by iteratively updating the weights which are estimated as a analytical solution of a weighted least squares (quadratic) approximation of the original problem[10].

Each iteration in IRLS can be divided into two sections, the first section in which the weights are updated and the second section in which the convergence criteria is tested on the objective function calculated for updated weights. In the weight update section, IRLS uses the Netwon-Raphson (or Newton) method for minimization of an objective function. The objective function for the logistic regression is a negative log likelihood function where is likelihood is in the form of a Bernoulli distribution. Since the Newton-Raphson method uses second-derivative, the weight update section comprises of computation of the gradient(first derivative, refer (10)), Hessian(second derivative, refer (11)) of the objective function and use of these values in calculating the new weights ((12)) (approximate posterior mode). The second part of each iteration of the algorithm updates $\mathbf{y}$ and B. The updated $\mathbf{y}$ values are used to update the negative log-likelihood (objective) function (Note: In the equations (10), (11) and (13), we define $y_n = \sigma((x_n)) = \sigma(\phi(x_n)w^*)$).

$$\nabla E(\mathbf{w}) = \Sigma_{n=1}^{N}(y_n - t_n)\phi_n = \mathbf{\Phi}^T(\mathbf{y} - \mathbf{t}) \tag{10}$$

$$H = \nabla\nabla E(\mathbf{w}) = \Sigma_{n=1}^{N} y_n(1 - y_n)\phi_n\phi_n^T = \mathbf{\Phi}^T B \mathbf{\Phi} \tag{11}$$

$$\mathbf{w}^{new} = \mathbf{w}^{old} - H^{-1}\nabla E(\mathbf{w}) \tag{12}$$

$$E(\mathbf{w}) = -\Sigma_{n=1}^{N}\{t_n \ln(y_n) + (1-t_n)\ln(1-y_n)\} \tag{13}$$

Hyper-parameter estimation

Estimation of hyper-parameters in the second step of RVM classification uses the same approach as in regression. The α parameters are computed using the equation (6) or (7), as in regression. The key idea is that the covariance Σ is the inverse of the Hessian at w_{MP}. Thus, w_{MP} is used to compute the B matrix using the equation (14), which is in turn used in computing the Hessian.

$$\begin{gathered} B_{nn} = \sigma(y(x_n))\{(1-\sigma(y(x_n)))\} \\ \text{where,} y(x_n) = \phi(x_n)w^* \end{gathered} \tag{14}$$

Convergence Criterion

Here, the Bayesian parameter estimation is approximated by a Laplacian approximation, which considers a second order (hence Hessian) Taylor expansion of the log-posterior around its mode. This results in a Gaussian approximate of the posterior[11]. Since the Hessian is a positive definite matrix at all points, the error function is log-concave (there exists an optimal minimum) [12].

3.3 Prediction

The prediction (estimating the targets for previously unseen inputs) is similar to that in RVM regression, but with one major difference. The output values are computed using the estimated parameters (weights) and the relevance vectors in the same way as in regression. Further, the logistic sigmoid function is applied on these outputs to give a classification output that expresses the probability of the new data belonging to either of the two classes(15).

$$t^* = \sigma(\phi(x^*, x^{rel})w^{RVM}) \tag{15}$$

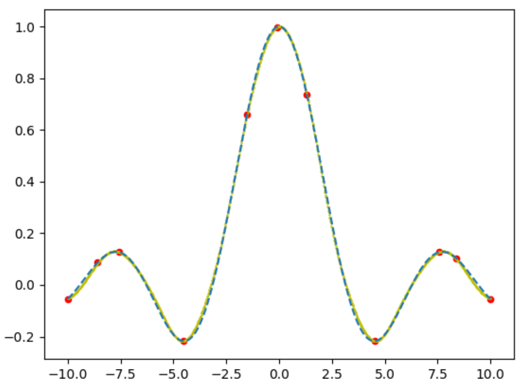
Since the sigmoid function results to an output varying from 0 to 1, we deploy a simple class definition, similar to the classes assigned as zeros and ones in the training sets:

$$Class(t^*) = \begin{cases} 0 & t^* \leqslant 0.5 \\ 1 & t^* > 0.5 \end{cases} \tag{16}$$

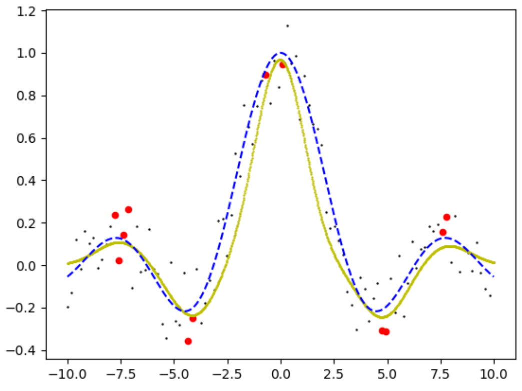
4 Results

4.1 Regression

To evaluate our implementation for our regression RVM model we first reproduced the synthetic example: the 'sinc' function [1]. The results we have obtained for both cases were quite similar to the ones illustrated in the original paper, confirming indeed all the features of the RVM compared to the SVM method. More specifically, by looking at Figure 1(a), we estimated the sinc function utilising an RBF kernel with $\gamma = 0.6$ (inverse of standard deviation). The error obtained was 0.000107 while requiring 11 relevance vectors. For case (b), we added gaussian noise of standard deviation 0.2 to the targets while again using an RBF kernel with $\gamma = 0.4$. The error obtained was 0.002 while we used 12 relevance vectors.



(a)



(b)

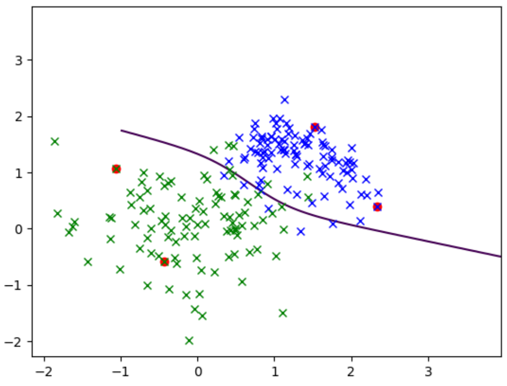
Figure 1: $Sinc(x)$ function with no added noise (a) and with added Gaussian noise $\sigma = 0.2$ (b)

Apart from the synthetic data, results were also obtained by testing RVM to real datasets. In particular, we evaluated its performance on the Boston Housing dataset by averaging over 100 repetitions. We used overall 100 samples and split the dataset into 90% training and 10% testing. The prediction error was estimated to 16.2 while the relevance vectors required were 45.7.
The results for Regression are summarized on the table below:

	errors		**kernels**	
Datasets	SVR	RVR	SVR	RVR
sinc(noise-free)	0.215	0.0001	39	11
sinc(Gaussian noise)	0.378	0.326	45.2	12
Boston Housing	8.04	16.2	142.8	45.7

4.2 Classification

For the evaluation of our classification model, we implemented a test on synthetic data exactly as it was described in paper [1]. 2-Dimensional input data were generated for our 2 classes as they appear in Figure 2. The number of relevance vectors found are 4 and as it is stated in the [1], they are not close to the boundary, but more close to the center of the Gaussians, as they are representative of each class. The error value that we got for 250 test sample was around 20%.



(a)

Figure 2: Boundary between the 2 Gaussian classes. The number of relevance vectors found was 4 (red dots).

Relevance Vector Classification was also applied in a real world toy problem, the *Pima Indians Diabetes* dataset. Before presenting the results over an average of 30 simulations, we should note that the dataset that was available to us, differs from the original one presented in Tipping's paper, as it includes more examples.

	errors		**kernels**	
Datasets	SVC	RVC	SVC	RVC
Pima Indians Diabetes	67	100	109	70

5 Discussion and Conclusion

Through this work, the original version of the Relevance Vector Machine was re-implemented and applied in both synthetic and real-world datasets. This implementation involves the formulation of algorithms for both the regression and the classification problems. Although our results do not thoroughly replicate the original ones, the deviations remain between a reasonable doubt, except from the Pima Indians dataset, where there is a significant deviation from the original results. This deviation, among other reasons, can be explained to the different form of the dataset. This is reasonable, because except from understanding the theoretical implementation and adjusting it to source code, a lot of initial parameters have to be tuned, and the right convergence and terminating conditions have to be defined. The scientific process that creates a fully functioning model from theory to practice, demands both time and a great number of simulations.
As a result, our future work can focus on:

- Fine tuning of the initial conditions, coefficients, and convergence conditions for both Regression and Classification algorithms
- Introduction of more simulations in both the datasets that were used in this work, and new datasets that can be found in modern literature

- Coarse search to define the optimality of each algorithm regarding the available datasets, in combination with avoiding the phenomenon of overfitting in any of these datasets

References

[1] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[2] M. Tipping, "Relevance vector machine," Oct. 14 2003. US Patent 6,633,857.

[3] M. Tipping and C. Bishop, "Variational relevance vector machine," Apr. 12 2005. US Patent 6,879,944.

[4] C. M. Bishop and M. E. Tipping, "Variational relevance vector machines," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 46–53, Morgan Kaufmann Publishers Inc., 2000.

[5] M. E. Tipping, A. C. Faul, *et al.*, "Fast marginal likelihood maximisation for sparse bayesian models.," in *AISTATS*, 2003.

[6] C. E. Rasmussen and J. Quinonero-Candela, "Healing the relevance vector machine through augmentation," in *Proceedings of the 22nd international conference on Machine learning*, pp. 689–696, ACM, 2005.

[7] A. Schmolck, "Smooth relevance vector machines," 2008.

[8] I. T. Nabney, "Efficient training of rbf networks for classification," 1999.

[9] C. Bishop, "Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn," *Springer, New York*, 2007.

[10] C. S. Burrus, "Iterative reweighted least squares," *OpenStax-CNX Web site. http://cnx. org/content/m45285/1.12*, 2012.

[11] S. Srihari, "Lecture notes on iterative re-weighted least squares," 2017. URL: `http://www.cedar.buffalo.edu/~srihari/CSE574/Chap4/4.3.3-IRLS.pdf`. Last visited on 2018/01/21.

[12] A. C. Faul and M. E. Tipping, "Analysis of sparse bayesian learning," in *Advances in neural information processing systems*, pp. 383–389, 2002.