KTH Computer Science and Engineering

Machine Learning Advanced

DD2434

Supervisor-Pawel Herman

Project Report

Sriram Venkatakrishnan – sriramve@kth.se
Evgeny Genov – genov@kth.se
Andreas Jönsson – ajon6@kth.se
Morris Groot Beumer – morrisgb@kth.se

Abstract

In this paper we explore the general Bayesian framework for obtaining sparse solutions first suggested by Michael E. Tipping. A special case application in this framework known as relevance vector machine (RVM) was implemented. The method is compared to a well-known support vector machine technique. The results are compared for classification and regression problems for artificially generated and real data sets and compared to equivalent results reported in the original paper by Tipping. Though our results generally aligned with Tipping's, we offer possible causes for the discrepancies in the results, whether they are regarding precision, or in the general features of RVM.

February 3, 2020

1 Introduction

In this paper we recreate the results reported in the 2000 and 2001 academic papers authored by Michael E. Tipping on 'relevance vector machines' (RVM for short)[1]. RVM is a special case of Bayesian sparse modelling methodology in linearly parameterized models, which is interesting because of its similarities and distinctions to another popular methodology, namely support vector machines (SVM). RVM has an identical functional form to SVM but offers a handful of advantages over it. The predictions in RVM have a probabilistic form, so the output would have conditional distribution $p(t|\boldsymbol{x})$, which shows the uncertainty in the predictions. Another advantage that RVM has over SVM, according to Tipping, is that RVM has no need for cross-validation of hyper parameters, which is typically required for validation of C (and ϵ for regression) in SVM models. RVM also stands out with the a fewer number of vectors during training, since in SVM the number of support vectors increases linearly with the size of the training set. RVM succeeds in reducing this amount by pruning the basis functions that correspond with the large number of weights that are sharply peaked at zero during training.

We start the exercise with making the implementation of the studied methods by ourselves. These techniques are an SVM classifier, support vector regression (SVR), RVM classifier and RVM regression. The entire algorithms are implemented 'from scratch' with minimal use of external libraries. To validate the algorithms, we repeat some of the tests done in the paper for assessing performance of RVM compared to SVM. The data sets used in the benchmark tests are the following

1. Synthetic data sampled from a sinc(x) function with Gaussian noise (Regression)
2. Synthetic data sampled from a sinc(x) function with Uniform noise (Regression)
3. Boston Housing market data (Regression)
4. Ripley's synthetic data (Classification)
5. Pima diabetes (Classification)
6. Banana data set (Classification)

2 Model specification of the RVM

The following section roughly follows the model specification in Tippings article [1]. The relevance vector machine for regression implements the support vector machine model in a Bayesian manner. Given input data points and corresponding targets $\{x_n, t_n\}_{n=1}^N$, the RVM aims to model a relation between t_n and x_n, such that new points t_n^* can be predicted using new inputs x_n^*, and the relation between t_n and x_n. It is assumed the targets follow the model with additive noise, i.e.

$$t_n = y(x_n; w) + \epsilon_n$$

Here it is assumed that the noise points ϵ_n are independent samples from the same normal distribution with mean 0 and variance σ^2, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ so that

$$p(t_n|x) = \mathcal{N}(y(x_n; w), \sigma^2)$$

Relevance vector machines use a similar model for $y(x_n; w)$ as the support vector machines do, described by

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

Here $K(x_i, x)$ is defined as the kernel function between x_i and x. It is assumed that t_n given x_n is conditionally independent, and as a consequence the likelihood can be written as

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

Here $\Phi = [\phi(x_1), ..., \phi(x_N)]^T$, where $\phi(x_n) = [1, K(x_n, x_1, ..., K(x_n, x_N)]^T$. Note that the conditioning on x_n is left out, and will be left out in the future to improve clarity.

The Bayesian approach is characterized by using a prior distribution over the weights in order to calculate the posterior distribution, rather than using maximum likelihood estimates for w and σ^2. Tipping [1] chooses a zero-mean Gaussion prior distribution over w. This is a popular choice, as the Gaussian prior is a conjugate prior for the Gaussian likelihood, leading to a posterior distribution of the same family as the prior, decreasing the computational complexity [2]. The prior distribution is then defined, with the help of the hyperparameter α, as

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

Again, using conjugate priors over the remaining parameters α and σ^2, the hierarchical Bayesian model can be completed. A conjugate prior for the Gaussian distribution with fixed mean is the $Gamma(a, b)$ distribution, defined by $Gamma(x|a, b) = \frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx}$, where $\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt$ is the Gamma function. This results in the following priors over α and $\beta \equiv \sigma^{-2}$

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

Bayesian inference now proceeds by using the Bayes' rule to compute the posterior distribution over the unknowns, given the data

$$p(w, \alpha, \beta|t) = \frac{p(t|w, \alpha, \beta)p(w, \alpha, \beta)}{p(t)} \tag{3}$$

From (3), a new point t_* can be computed given t, with the use of the posterior predictive distribution, integrating out w, α, β

$$p(t_*|t) = \int p(t_*|w,\alpha,\beta)p(w,\alpha,\beta|t)\,dw\,d\alpha\,d\beta \tag{4}$$

The problem now is that this integral is intractable, and the posterior distribution in (3) can not be solved analytically. Hence, effective approximations have to be made. To do this, the posterior distribution can be rewritten in the following way

$$p(w,\alpha,\beta^2|t) = p(w|t,\alpha,\beta)p(\alpha,\beta|t) \tag{5}$$

Tipping [1] shows that $p(w|t,\alpha,\beta)$ in (5) follows a Gaussian distribution and can be solved analytically, and gives valid reasons to approximate $p(a,\beta|t)$ by the delta function at the most probable values of β and α, so that $\delta(\alpha_{MP},\beta_{MP}) \approx p(\alpha,\beta|t)$. To find this delta function, Bayes rule is used once again to conclude this is similar to maximizing

$$p(\alpha,\beta|t) \propto p(t|\alpha,\beta)p(\alpha)p(\beta) \tag{6}$$

In the case of uniform hyperpriors α and β, only the term $p(t|\alpha,\beta)$ in (6), which is often referred to as the marginal likelihood and is analytically computable, has to be maximized. However, values of α and β that maximize $p(t|\alpha,\beta)$ can not be obtained in closed form, and are estimated in an iterative way, in order to effectively approximate the posterior distribution in (3) in the end. Tipping elaborates these iterative procedures in further detail in Sparse Bayesian Learning [1]. Using the approximated posterior distribution, the predictive distribution from (4) can be estimated by a Gaussian distribution, where again the point estimates α_{MP} and β_{MP} were used to approximate

$$p(t_*|w_{MP},\alpha_{MP}) \approx p(t_*|w,\alpha,\beta)$$

For specifics on the delta function approximation, the iterative process to find α and β that maximize (6), the final closed form estimates for the posterior and predictive distributions and the conversion to the RVM classification model, we refer to Tipping [1].

3 Results

3.1 SVM and RVM classification

3.1.1 Ripley's synthetic data

In accordance with the paper by Tipping, we test the algorithm's performance by applying it to artificially-generated data in two dimensions. The data is binary classified, with both classes generated from the mixture of two Gaussians and overlapping with a Bayes error of 8% [3]. The classifier uses the kernel trick, where the mapping function is a 'Gaussian kernel'

$$K(\boldsymbol{x}_m,\boldsymbol{x}_n) = exp(-r^{-2}||\boldsymbol{x}_m - \boldsymbol{x}_n||^2) \tag{7}$$

where r is a width parameter equal to 0.5. The best value for parameter C in SVM is selected from the best result of 5-fold cross-validation ($C = 2$). The training data set contains 100 points randomly selected from Ripley's original 250. The test set contains 1000 points. The results achieved with self-made implementation are compared to results reported in the paper by Tipping in the table below

	test error	# functions
Original SVM	10.6%	38
Reproduced SVM	11.5%	42
Original RVM	9.3%	4
Reproduced RVM	10.4%	4

In the plot we can verify that support vectors in RVM show a different characteristic with relevance vectors located further from the decision boundary. These points represent the middle of the class groups rather than border points.

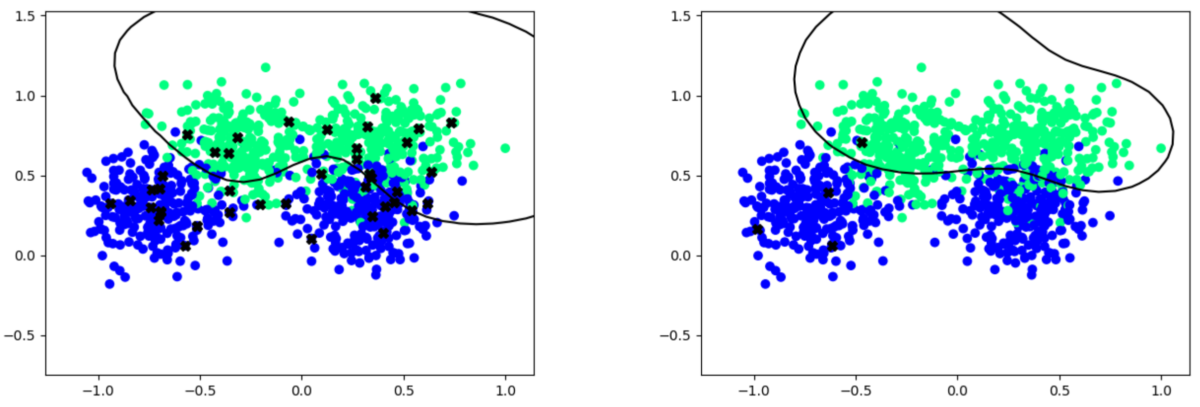

Figure 1: SVM classification (Left) and RVM classification (Right) of Ripley's data. The decision boundary is in black. Support and relevance vectors are marked as black crosses

3.1.2 Benchmark tests

For further comparison we use tests on some of the benchmark data sets used by Tipping. We focused only on binary classification problems to avoid the added complexity of multi-class set. The Pima Diabetes data set comprises 200 examples of training data and 332 examples of test data. The Banana data set is divided into 100 different data sets, each containing 400 items in the training set and 4900 data points in the testing set. We applied our model on the first 10 of these 100 data sets and averaged out the results, which is the same approach as Tipping[1].

We apply a Gaussian kernel, same as Tipping, with an input scale parameter selected on the basis of 5-fold cross-validation. Instead of using the notation with a width parameter, like we did in (7), we used the notation with variance (8).

$$K(\boldsymbol{x}_m, \boldsymbol{x}_n) = exp\left(-\frac{||\boldsymbol{x}_m - \boldsymbol{x}_n||^2}{2\sigma^2}\right) \tag{8}$$

For Pima Diabetes, we found a value for $C = 0.9$ and $\sigma^2 = 5$, (or $r = \sqrt{10}$) for the SVM, and used the same value for σ^2 in the RVM. In the case for the Banana data set, we found a value for $C = 1$ and $\sigma^2 = 0.5$ (or $r = 1$), and for RVM we used the same σ^2. The RVM results are compared by a normalized mean to the SVM. This makes the findings easier to compare to the findings of Tipping.

			Original				**Reproduction**			
			Errors		**Vectors**		**Errors**		**Vectors**	
Dataset	**N**	**d**	**SVM**	**RVM**	**SVM**	**RVM**	**SVM**	**RVM**	**SVM**	**RVM**
Pima Diabetes	200	8	20.1%	19.6%	109	4	28.0%	32.8%	150	3
Banana	400	2	10.9%	10.8%	135.2	11.4	16.6%	11.0%	141.5	15.9
Normalized mean			1.00	**0.98**	1.00	**0.06**	1.00	**0.91**	1.00	**0.07**

3.2 SVM and RVM regression

Following the methodology proposed in the paper of Tipping, we replicate the algorithm for SVM regression as well as for RVM regression. Although sparse, SVM makes unnecessary use of basis functions due to the fact that the number of training vectors increases linearly with the increase in the size of the training. This limitation is overcome while using RVM model. To present a visual output of the both the regression models, we use a synthetic dataset for both regressions, the $sinc(x)$ function. In place of the classification margin, an ϵ-insensitivity region is introduced. Thereby we know that the support vectors lie on the edge or outside of this region. We replicate the kernel mentioned in the paper, a univariate 'linear spline' kernel.

$$K(x_m, x_n) = 1 + x_m x_n + x_m x_n \min(x_m, x_n) - \frac{x_m + x_n}{2} \min(x_m, x_n)^2 + \frac{\min(x_m, x_n)^3}{3} \tag{9}$$

This kernel is used to define a set of basis functions $\phi(x) = K(x, x_n), n = 1, ..., N$. Also, for comparing the outputs of both SVM and RVM regression, we fix the noise variance to 0.01^2 in the Relevance Vector Machine. This is comparable to the ϵ-insensitivity in the SVM. The implementation is compared with Tipping's results provided in the paper.

	Original		**Implemented**	
	SVM	**RVM**	**SVM**	**RVM**
Support/Relevance Vectors	35	9	52	37
Largest error.	0.0100	0.007	0.066	0.0003

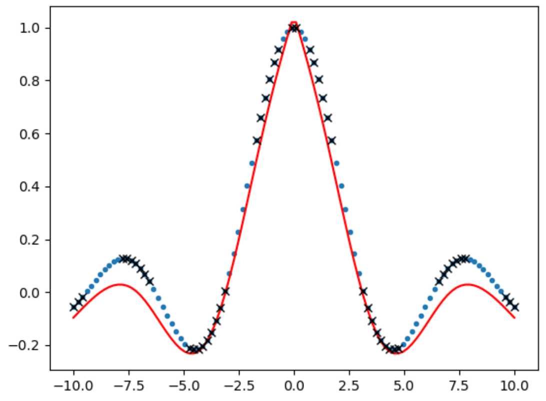
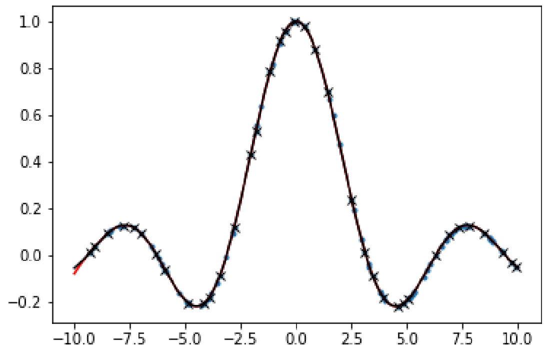

Figure 2: SVM Regression (Left) and RVM Regression (Right) of sinc(x) function.The Red line depicts the SVM/RVM fit, blue is the true function and the black dots are support/relevance vectors

When considering the plot without noise in figure 2, the largest error obtained by us is 0.066 for SVM, whereas the result reported by Tipping is 0.01. This discrepancy can be related to lack of information on the range of C used in the paper.

Also, to picture the a real world example, we add noise to the dataset, a uniform noise of $[-0.2, 0.2]$ and a gaussian noise with a standard deviation of 0.1, is added to the targets. For SVM, the parameters C and ϵ were tuned using a 5-fold cross validation, whereas in RVM the α's and σ^2 are automatically estimated using the learning procedure. The test our model to a real world example, we use the Boston Housing dataset, to measure the performance of RVM in such scenario.

The table below compares the results of Tippings paper with our own implementation. It can be seen that the number of support vectors both SVM and RVM produce are of similar quantity as is reported by Tipping. The root mean squared errors for the $sinc(x)$ function, however, are significantly lower in the reproduced case compared to Tipping. Especially the RVM root mean squared error is a factor 10 lower than this error reported by Tipping. This is likely due to the fact that some model settings were not described by Tipping, such as the cost function C and the maximum amount of iterations. Also, the ϵ value used by us is 0.1 in order for us to replicate the results in the paper, whereas it is specified as 0.01 in the paper. Figure 3 and figure 4 show the fitted SVM function (left) and the fitted RVM function (right) plotted against the true $sinc(x)$ function, with the support vectors marked as $\times$.

	Original		**Implemented**	
	SVM	**RVM**	**SVM**	**RVM**
Support/Relevance Vectors(Uniform noise)	44.3	7	25	7
RMS deviation(Uniform noise)	0.215	0.187	0.153	0.019
Support/Relevance Vectors(Gaussian noise)	45.2	6.7	22	7
RMS deviation(Gaussian noise)	0.378	0.326	0.209	0.025
Support/Relevance Vectors(Boston Housing)	142.8	39	-	53
RMS deviation(Boston Housing)	8.04	7.46	-	8.009

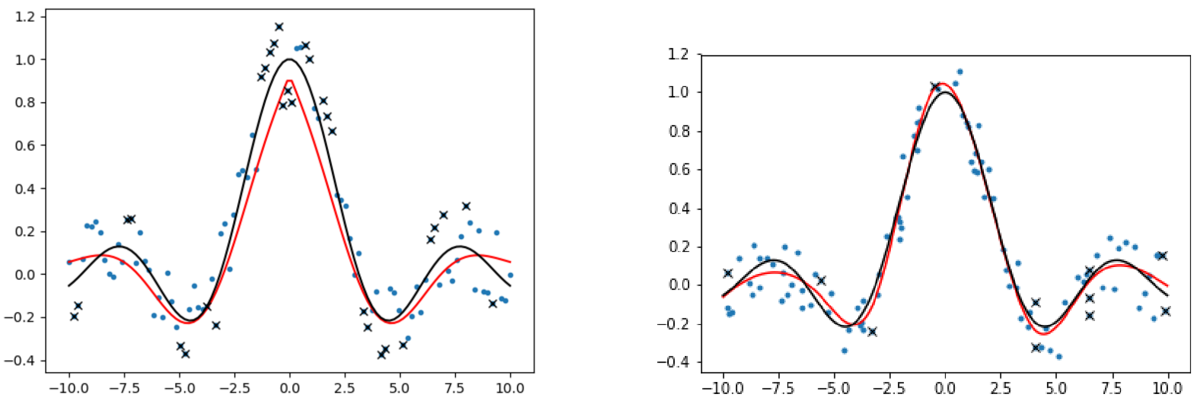

Figure 3: SVM Regression (Left) and RVM Regression (Right) of sinc(x) function with uniform noise.

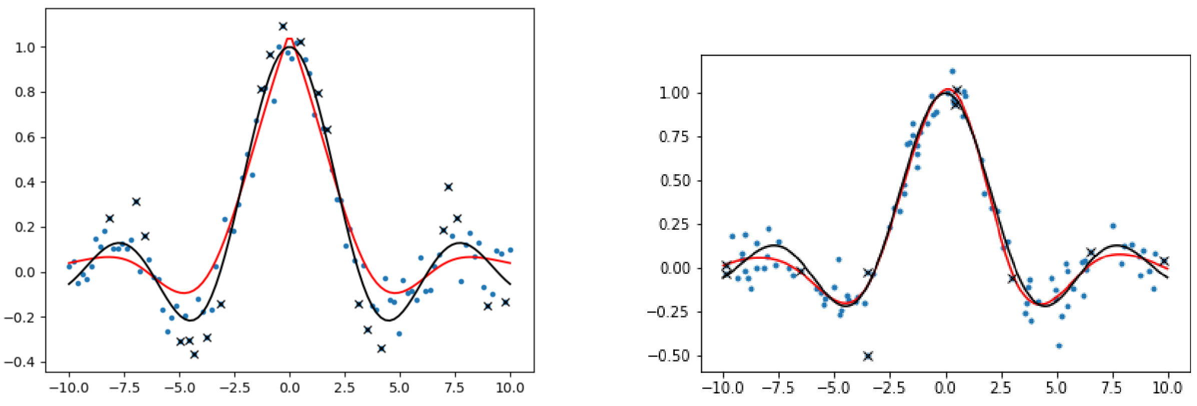

Figure 4: SVM Regression (Left) and RVM Regression (Right) of sinc(x) function with gaussian noise.

4 Discussion

In this project we intend to reproduce the results from the Tipping paper with the best matching accuracy. However, it can be seen that there are still discrepancies in our results. We point these discrepancies to a number of ambiguities regarding what hyper parameters were used in the experiments in the paper. In SVM, for example, the Lagrangian variable α is constrained with a bound value C. While we can perform type-2 optimization for the hyper parameters, i.e. 5-fold cross-validation, the range of C values reviewed in such process remains unspecified in the original paper. There is also some uncertainty with values for pruning thresholds in SVM and RVM, as well as stopping criteria in RVM.

The findings for the Benchmark Tests in section 3.1.2 varied to those of Tipping. We had larger errors for Pima Diabetes, and for our SVM we used a larger amount of support vectors. These discrepancies might be explained by us not finding good enough hyper parameters (C and σ^2). The 2001 paper did not make it clear what hyper parameters Tipping used, so we had to try and find our own. In the case of the Banana data set, our SVM had a larger error, but fewer support vectors, compared to Tipping's findings, whereas our RVM had lower error, but more support vectors. Again, this is probably caused by us using different hyper parameters. However, our normalized means were very similar to Tipping's. From this, we argue that our findings in this section support Tipping's claims about the sparsity of RVM with comparable precision to SVM.

The findings in section 3.2 about Regression also varied to those found in Tipping's paper. We have a larger error for sinc(x) function without noise and also we use a higher number of support vectors. This discrepancy can also be likely related to the fact that we were not able to find good hyperparameters(C). Also, our kernel used absolute values of x_m, x_n for computation, in order for us to obtain the output as desired. This implementation has not been specified in the 2001 paper of Tipping, adding ambiguity to our findings. The outputs sinc(x) function with Uniform and Gaussian noise have modelled well to that in the paper, although deviations are found. This can be attributed to the issue we face in the no noise model. We tried our implementation for regression on a real data set (Boston housing), since this provides a good way to validate the advantageous features of RVM. The output obtained closely matches the outputs in the paper, validating the paper's findings.

Our findings align fairly well with the advantages associated with using RVM method, as claimed by Tipping [1]. Firstly, there is a considerably smaller number of basis functions used in RVM predictions compared to SVM predictions. This feature improves the sparsity of the model, and because this can be reduced while training, it reduces computational complexity when running the training. In the experiments, training RVM models generally took less running time, also benefiting from no cross-validation optimization needed. However, this reduced training time could be additionally affected by differences in the code between our SVM and RVM implementation.

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] C. M. Bishop, *Pattern recognition and machine learning.* springer, 2006.

[3] B. Ripley, "Neural networks and pattern recognition," *Cambridge University*, vol. 7, 1996.