DD2434 Project Report: Sparse Bayesian Learning and the Relevance Vector Machine Revisited

Group 48:
Povel Forsare Källman, Martin Köling, Sonja Horn, Daniel Wass

October 24, 2020

Abstract

This study aims to reproduce parts of the 2001 paper 'Sparse Bayesian Learning and the Relevance Vector Machine' by Michael E. Tipping, where a probabilistic sparse learning method is introduced and compared with the popular support vector machine. The study compares the models for new data sets as well as data sets used in Tipping's paper in an attempt to validate his results. The study concludes that the main advantage of the relevance vector machine stated by Tipping, i.e. that it produces similar results to the support vector machine while being vastly sparser and thus more efficient for classification, can be validated.

Acknowledgements

The authors wish to thank Negar Safinianaini for helpful discussions regarding implementation.

[*]POVELK@KTH.SE, KOLING@KTH.SE, SONJAHO@KTH.SE, DWASS@KTH.SE

1 Introduction

This study aims to reproduce parts of the paper 'Sparse Bayesian Learning and the Relevance Vector Machine' by Michael E. Tipping, first published in 2001 in the 'Journal of Machine Learning Research' [1]. Tipping introduces the Relevance Vector Machine (RVM) and compares its performance to its source of inspiration - the Support Vector Machine (SVM), first presented by Vapnik in 1995 [2] - both for regression and classification. The results show that the RVM on an average is performing at a similar level to the SVM for classification, and on a higher level regarding regression, while using a fraction of the amount of vectors when predicting, which regulates the prediction time complexity. A less RVM superiority is showed for classification problems than for regression. To further investigate this, the study is delimited to binary classification. The SVM and the RVM are implemented, evaluated, and compared based on a selection of data sets from Tipping's paper and two other data sets.

1.1 Scope

The study intends to evaluate and validate Tipping's results, by reproducing results from the original publication and complementing them with new evidence. In detail, the study will focus on the comparison between the SVM and the RVM binary classifiers.

2 Theory

This section describes the RVM and the fundamentals of the SVM. Since this project is delimited to classification, some specific details regarding regression of RVM are excluded. For these we instead refer to Tipping [1]. For more details of the SVM, we refer to Vapnik [2]. The data is $\{x_n, t_n\}_{n=1}^{N}$, the weights $\mathbf{w}$ are $[w_0, \dots w_N]$, and the precision over the weights $\boldsymbol{\alpha}$ are $[\alpha_0, ..., \alpha_N]$.

2.1 The Support Vector Machine

The binary SVM classifier is making predictions by mapping a sample x to a prediction of its target class $t \in \{-1, 1\}$ by a linear model, applying a non-linear Kernel function, $y(x; \mathbf{w}) : \mathbf{x} \to \mathbf{y}$:

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

The sign of y is the class estimate for the sample. The model is sparse, as many weights are pushed to zero during the learning process, leaving only the subset of training samples with corresponding weights $w > 0$ to impact the prediction. These samples are called *support vectors.* The non-linear Kernel maps the input data into a high dimensional feature space. A decision boundary, separating the two classes, is constructed in this space, allowing separation of data that is non-separable in its original space [2].

2.2 The Relevance Vector Machine

The mapping of the RVM is similar to the one of the SVM, but with a probabilistic approach, introducing a prior over the weights and a distribution over the targets. In contradiction to the SVM, the RVM is more intuitive for regression tasks than for classification, and is thus explained in that order. For the regression case, the targets $\mathbf{t}$ are assumed to be Gaussian distributed, according to:

$$p(\mathbf{t}|\mathbf{w}, \sigma^2) = \prod_{i=1}^{N} \mathcal{N}(t_i|y(x_i;\mathbf{w}), \sigma^2), \text{ where} \tag{2}$$

$$t_i = y(x_i;\mathbf{w}) + \epsilon, \text{ and } \epsilon \sim \mathcal{N}(0, \sigma^2), \text{ and } \sigma^{-2} \sim \text{Gamma}(\sigma^{-2}|c, d). \tag{3}$$

For binary classification, $t_n \in \{0, 1\}$, it is sufficient to predict the probability of a sample belonging to one of the classes. There is no noise and the linear model $y(x;\mathbf{w})$ is generalised by the sigmoid link function $\sigma(y) = \frac{1}{1+e^{-y}}$. Thus, following the Bernoulli distribution, the likelihood of $\mathbf{t}$ is written by

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma(y(x_n;\mathbf{w}))^{t_n} \left(1 - \sigma(y(x_n;\mathbf{w}))\right)^{1-t_n}. \tag{4}$$

In this report, as well as in Tipping's paper, a zero-mean Gaussian prior with individual precision for each weight is considered:

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

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N} \text{Gamma}(\alpha_i|a, b). \tag{6}$$

We assume uniform hyperpriors, i.e. $a = b = c = d = 0$, leaving us with results independent of unit measurement, but, conveniently, with scale-invariance [1].

We now desire to maximise our parameter posterior, given the data:

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

but since directly computing the denominator of the r.h.s. is computationally infeasible, we decompose the posterior into [1]:

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

Optimal parameters cannot be obtained in closed form from (8), and we thus require an iterative approximation of each term of the r.h.s.

Maximising the second term of the r.h.s. of (8), $p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t}) \propto p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha})p(\sigma^2)$, gives us the optimal hyperparameters $\boldsymbol{\alpha}$ (and $\sigma^{2(new)}$ for regression) for the current $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. In the case of uniform hyperpriors, it is sufficient to just maximise the target likelihood $p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)$. Differentiation of this likelihood, and setting to zero, gives the optimal $\boldsymbol{\alpha}$ (see Tipping [1]):

$$\alpha_i^{new} = \frac{1 - \alpha_i \Sigma_{ii}}{\mu_i^2}. \tag{9}$$

For classification tasks, the maximisation of the first term of the r.h.s. of (8) can be approximated using the same approximation as MacKay, using Laplace's method [3]. By keeping the current values of $\boldsymbol{\alpha}$ fixed and maximising the weight posterior, we obtain the mean, i.e. the most probable weights $\mathbf{w_{MP}}$, and covariance matrix of the weight posterior $\boldsymbol{\Sigma}$. Since $p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) \propto p(\mathbf{w}|\mathbf{t})p(\mathbf{w}|\boldsymbol{\alpha})$, it is equivalent to maximise the following with respect to the weights:

$$\log\left(p(\mathbf{w}|\mathbf{t})p(\mathbf{w}|\boldsymbol{\alpha})\right) = \sum_{n=1}^{N}\left[t_n \log\sigma(y(x_n;\mathbf{w})) + (1-t_n)\log(1-\sigma(y(x_n;\mathbf{w})))\right] - \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w}, \tag{10}$$

where $\mathbf{A} = \text{diag}(\alpha_0, ..., \alpha_N)$. Note that the prior over $\mathbf{w}$ acts like a regularisation term, pushing down the weights. Following MacKay's approximation, as described by Tipping [1], gives us

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^T\mathbf{B}\boldsymbol{\Phi} + \mathbf{A})^{-1}, \tag{11}$$

$$\mathbf{w_{MP}} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{B}\mathbf{t}. \tag{12}$$

where $\boldsymbol{\Phi}$ is the design matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}(x_1), ..., \boldsymbol{\phi}(x_N)]$, where $\boldsymbol{\phi}(x_n) = [1, K(x_n, x_1), K(x_n, x_2), ..., K(x_n, x_N)]$. The Laplace approximation maps our classification problem into a regression problem, with the matrix $\mathbf{B}$ being a diagonal matrix where the diagonals $\beta_1, ..., \beta_N$ are the inverse data-dependent noise variance, $\beta_n = \sigma(y(x_n;\mathbf{w}))(1-\sigma(y(x_n;\mathbf{w})))$ [1].

The results from (11) and (12) can then be used in (9), with $\mathbf{w_{MP}}$ as μ, to update a new, more accurate $\boldsymbol{\alpha}$. In turn, this new $\boldsymbol{\alpha}$ can then be used to derive a new mean and a new covariance matrix, again using (11) and (12). As we shall see, many of the α-values will go towards infinity, whereas the weights approach zero, leading to a very sparse model. This is desirable as the computationally heavy Kernel function, used for classifying new samples, only needs to be applied to the training samples corresponding to the non-zero weights - i.e. the relevance vectors. The predictive computation scales cubic and the memory required scales squared to the number of basis functions used, i.e. the number of relevance vectors [1].

3 Method

3.1 Implementing the Relevance Vector Machine

The RVM was implemented by first initialising $\alpha_{0-N} = 0.1$ and $\mathbf{w} = \vec{0}$, after reasoning with the course teaching assistant Negar Safinianaini [4]. The iterative process, equivalent to Tipping's method, was started by approximating the optimal $\boldsymbol{\Sigma}$ and $\mathbf{w_{MP}}$ according to (11) and (12), followed by updating $\boldsymbol{\alpha}$ according to (9). For each iteration, all basis functions with corresponding α-values exceeding a set tolerance were 'pruned away'. This method was also used by Tipping in the original paper, ensuring that only relevant α-values, their corresponding weights, and data points were considered in the next iteration. By utilizing this method, we "avoid ill-conditioning" and enjoy a "considerable and advantageous acceleration of the learning algorithm" [1], while also ensuring that only relevance vectors remain once the algorithm converges. The iterative process will then proceed until convergence, which is defined by the change in the $\boldsymbol{\alpha}$ vector. The largest difference in individual values between the old and the updated $\boldsymbol{\alpha}$ vectors is computed, and when the largest difference is less than 10^{-3}, the algorithm has reached convergence.

The tolerance boundary for accepting relevance vectors (i.e. the condition for pruning), was set to α-values exceeding 10^{12}, following Tipping's method. As a reminder, α-values exceeding 10^{12} are equivalent to corresponding weight very close to zero (9). The only exception was for the German data set, where reasonable results could not be obtained by this boundary. The boundary was lowered to 10^9, which resulted in more satisfying results.

The kernel used for the basis functions ϕ was the following radial basis function (RBF), also used by Tipping [1]:

$$K(x_m, x_n) = exp(-r^{-2}||x_m - x_n||^2). \tag{13}$$

In order to determine the length-scale parameter r for the kernel, an iterative approach was implemented. For each data set, the algorithm was run for a range of values, from which the best results were chosen. These estimated optimal parameters are presented in Table 1. The ranges of the values were centered around the estimated optimas for the SVM.

3.2 Implementing the Support Vector Machine

The same kernel (13) was used for basis functions for the SVM. When determining the length-scale parameter r in case of the SVM and its penalization term scalar C, a cross fold validation was used for each data set, as recommended by Bishop [5]. The ranges of the cross-fold validation was then based on the benchmarked values used by Rätsch [6]. The best parameters can be seen in Table 1.

Since the tolerance boundary for accepting relevance vectors was set to $\alpha = 10^{12}$ in the RVM implementation, the following reasoning was used to set the tolerance boundary for accepting support vectors in the SVM implementation:

$$\begin{gathered} \alpha = \frac{\gamma}{w^2} \propto \frac{1}{w^2}, \\ \frac{1}{w^2} > 10^{12} \Leftrightarrow w^2 < \frac{1}{10^{12}} \Rightarrow w < \frac{1}{\sqrt{10^{12}}} \Rightarrow w < 10^{-6}. \end{gathered} \tag{14}$$

Thereby, the boundary for accepting support vectors was set to 10^{-6}.

3.3 Validation and Comparison

To validate Tipping's results, the RVM and the SVM were applied to a selection of data sets that were used in the original paper. Important properties of these data sets such as the input dimension d, the number of test samples N_{test} and the number of training samples N_{train} are presented in Table 1. All outputs for the data sets used are one-dimensional and binary. The reason for this selection was the desire to validate Tipping's result on different cases, as some data sets are low dimensional and some data sets are high dimensional. The data sets also differ in the number of samples used for training and testing. To extend Tipping's results, two additional data sets were included, the *Heart* data set from Rätsch's benchmark data base [6], and *Haberman's survival data set* [7], which both are also presented in Table 1 together with their respective properties, marked with *. These data sets were chosen mainly for their input dimensions, i.e. one of low dimension and one of higher, in order to allow for investigation of replicability.

The data sets from Rätsch's benchmark database are pre-split into 100 test and training subsets. As in Tipping's paper, the ten first of these were utilised and the error rates and number of vectors presented in Table 2 are the averages over these. Haberman's data set was shuffled, split into five subsets and then five-fold cross-validated. One split was used for testing and the rest for training. The values in Table 2 are the averages over these.

Table 1: Data sets.

Data set	$\mathbf{N_{train}}$	$\mathbf{N_{test}}$	**d**	**C**	$\mathbf{r_{SVM}}$	$\mathbf{r_{RVM}}$
Banana	400	4900	2	296.2	0.9	0.7
Breast Cancer	200	77	9	11.19	5.4	4.6
German	700	300	20	2.862	5.1	5.1
* Haberman	245	61	3	3.162	5.25	5.3
* Heart	170	100	13	3.162	2.45	4.2

4 Results

Table 2 presents the results of applying the SVM and the RVM on the original and the additional data sets. Column 2 to 5 showcase Tipping's results, while column 6 to 9 showcase the results of this study, in regards to the average error over test data and average number of support vectors or relevance vectors. The normalised means were computed on the mutual data sets while the *normalised means were computed on the data sets exclusive to this study.

Generally, Tipping's SVM has a lower average error than our SVM, while our SVM performs slightly better in the number of support vectors. As for the RVM, Tipping's results and the results of this study are very similar with no significant superiority on either average error or number of relevance vectors.

Table 2: Results for RVM & SVM.

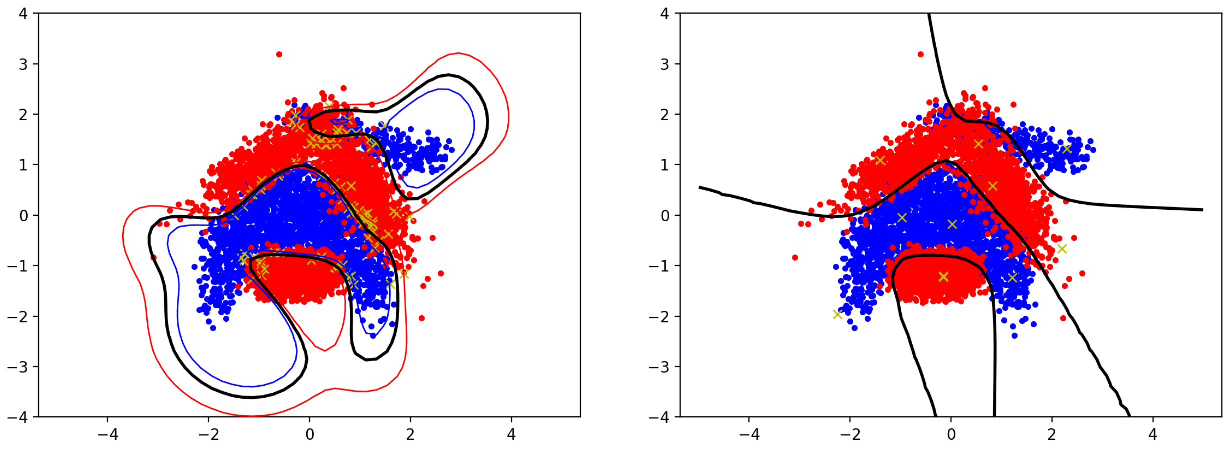
	Tipping				***New***			
	errors		**vectors**		**errors**		**vectors**	
Data set	**SVM**	**RVM**	**SVM**	**RVM**	**SVM**	**RVM**	**SVM**	**RVM**
Banana	10.9%	10.8%	135.2	11.4	12.7%	10.8%	93.2	11.5
Breast Cancer	26.9%	29.9%	116.7	6.3	33.8%	29.2%	115.7	4.8
German	22.6%	22.2%	411.2	12.5	24.9%	24.4%	409.7	17.7
Normalised Mean	1	**1.03**	1	**0.02**	**1.13**	**1.05**	**0.89**	**0.06**
* Haberman					23.2%	25.5%	138.6	10.4
* Heart					19.3%	18.2%	100.1	6.6
***Normalised Mean**					1	**1.02**	1	**0.07**

Figure 1 shows the SVM and RVM classifiers for the first split of the Banana data set. As Table 2 also suggests, the number of support vectors (yellow crosses) greatly exceeds the number of relevance vectors. Figure 1 also shows that the decision boundary for the SVM classifier is more complex than the decision boundary for the RVM classifier.

5 Discussion

5.1 Error rates

The results of this study (Table 2) show error rate superiority in favor of the RVM for all data sets evaluated, except for Haberman, where the SVM outperformed the RVM. Furthermore, by inspecting the normalised means for the mutual data sets in Table 2, we could see that Tipping's average SVM performance was proportionally closer to the performance of his RVM, compared to the difference of



(a) SVM, $C = 296.2$, $r = 0.9$.

(b) RVM, $r = 0.7$.

Figure 1: The SVM and RVM for data set Banana 1. (a) Shows the contours for the decision boundary of the SVM in black together with the margins in blue and red. The test data from the two classes are colored in blue and red, respectively. (b) shows the contours for the decision boundary of the RVM together with test data from the two classes colored in blue and red, respectively. The support vectors and the relevance vectors are displayed by yellow crosses in both (a) and (b).

our models. However, looking at the *Haberman* and *Heart* data sets, the *normalised mean error rates show very similar proportions to Tipping's study, with the SVM and RVM performing very similar.

Additionally, the average error for the mutual data sets of our SVM was significantly higher than that of Tipping's. Considering that our RVM was implemented almost identically to the one in Tipping's paper, whereas the SVMs were implemented independently, the larger differences in errors for the SVMs were to be expected. This would imply weaknesses in our SVM implementation, rather than that our RVM is a more robust classifier than the SVM in general, which would be contradictory to Tipping's very similar results for the two algorithms.

For the *Haberman* data set, some notable behaviour was observed for the error rate. The RVM performed worse than the SVM, something which is not reflected in the other data sets for our models. This data set is the only one that was not pre-split, and its results were thus five-fold cross-validated, instead of averaged over a fraction of the predetermined splits. Additionally, it is also the data set with the highest N_{train}/N_{test}-ratio. That the SVM is superior for this data set could be caused by such data set characteristics, but could also be from some entirely different reason. However, we do not find any of the above statements to be compelling enough, and thus, no conclusions regarding the SVM-superior results can be drawn. Note that Tipping's results for the *Breast Cancer* data set shows similar results.

By isolated comparison of our models, a somewhat weak correlation can be observed between dimensionality and proportional difference in error rate, i.e. $\epsilon_{RVM}/\epsilon_{SVM}$. *Banana* (2d) and *Haberman* (3d) differ by 15 and 10 percent respectively, whereas *Heart* (13d) and *German* (20d) differ by 6 and 2 percent respectively. The *Breast Cancer* (9d) data set is somewhat contradicting to such correlation, with a difference of 14 percent. However, even if this correlation would be stronger, since our SVM performed at a lower level than Tipping's, implying poor implementation, and considering the low number of data sets evaluated, no conclusions of causation could be drawn.

5.2 Amount of vectors

One of the main advantages of the RVM compared to the SVM, according to Tipping, is the statement that the RVM is a much sparser model [1]. Table 2 clearly shows the RVM superiority in the number of relevance vectors in comparison to the number of support vectors. The superiority in sparsity is validated by our results on the original data sets, but also extends to the new data sets, *Heart* and *Haberman*.

Further examining Table 2, we acknowledge the vector ratio between the number of relevance vectors and the number of support vectors for a given data set, i.e. $relevancevectors/supportvectors$. The data sets *German* (20d), *Heart* (13d) and *Breast Cancer* (9d) have a ratio of 4.3, 6.6, and 4.1 percent respectively, while data sets *Haberman* (3d) and *Banana* (2d) have ratios 7.5 and 12.3 percent respectively. The ratio is slightly lower for the higher dimensional data sets, which could imply that high dimensional data might draw greater prediction complexity advantages for the RVM than low dimensional data. However, such small complexity improvements would be considered as highly secondary, as the prediction complexity of the RVM is already vastly advantageous for all data sets examined. Further examination of very high dimensional input data to strengthen or discard these speculations would be interesting.

An interesting observation done by Tipping is the placement of the support vectors in regards to the relevance vectors in input space. Typically, the support vectors are located close to the decision bound, whereas the relevance vectors instead can be found further away from the bound. Looking at it from a Bayesian framework, the behavior can be explained by the heavier penalisation of output vectors aligned closer to the bound by the RVM. This was also the case in our study, which can be shown in Figure 1. However, as Tipping further states, this does not implicate whether or not output vectors closely aligned with the bound are more correct than "prototypically"-located output vectors, but merely presents an interesting reflection of the results between the models [1].

5.3 Conclusion

Finally regarding classification, this study could validate Tipping's claim of the RVM being vastly sparser than its popular competitor, the SVM, while also producing highly competitive results. In the experiments conducted for this study, the RVM out-performed the SVM regarding error rate in four out of five cases. Regarding sparsity, the RVM exceeded the SVM across all the data sets.

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. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[3] D. J. MacKay, "The evidence framework applied to classification networks," *Neural computation*, vol. 4, no. 5, pp. 720–736, 1992.

[4] N. Safinianaini, "Discussions regarding project implementation." 2020.

[5] C. M. Bishop, *Pattern recognition and machine learning.* Springer Science+ Business Media, 2006.

[6] G. Rätsch, "Ida benchmark repository." [Online]. Available: http://www.raetschlab.org/Members/raetsch/benchmark

[7] UCI Machine Learning Repository, "Haberman's survival data set." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival