The Relevance Vector Machine and Sparsity of Bayesian Learning
DD2435 Machine Learning, Advanced Course

Adrian Chmielewski-Anders amca3@kth.se
Bas Straathof btstr@kth.se
Clara Tump tump@kth.se
Leo Zeitler llze@kth.se

January 16, 2019

Abstract

The Relevance Vector Machine (RVM) is a regression and classification method similar in form to the popular Support Vector Machine (SVM). In this project, the original RVM method as described by Tipping [1, 2] is re-implemented, and the results of the original method reported by Tipping are reproduced and critically assessed. To deepen our understanding of RVMs and to be able to analyze their potential in more detail, we implement the fast RVM algorithm [3] and the RVM* method [4] as well. Tipping's performance comparison of RVMs and SVMs is enriched by testing on data sets that reflect general and edge cases that were not mentioned in the initial publication. The claimed advantages of RVMs such as sparsity during prediction, predictive distributions, and automatic nuisance parameter re-estimation are thoroughly analyzed and discussed. Also, the drawbacks of RVMs, such as poor scalability to large data sets due to a sub-optimal time-complexity during its training phase, are critically assessed. Special attention is devoted to the sparsity properties of the RVM, since these come at the expense of computational complexity and meaningfulness of the predictive distribution that the method provides. From these analyses, we conclude that in some cases the advantages of RVMs in comparison to SVMs are justified. However, there exist many edge cases in which such benefits are nullified; thus, the choice for RVMs is highly problem and data dependent.

1 Introduction

1.1 Description of the original article and method

The Relevance Vector Machine (RVM) was introduced by Tipping [1], and represents a probabilistic implementation of the Support Vector Machine (SVM) that can be used for both classification and regression. The advantages of RVMs in comparison to SVMs are that their solutions are often more sparse, and probabilistic predictions are produced instead of point estimates. Moreover, in contrast to SVMs, RVMs automatically estimate noise parameters from the data and are unconstrained when it it comes to the choice of kernel function. We will exemplify how the RVM as proposed by Tipping [1, 2] can be implemented for regression and classification. It is assumed that the reader is familiar with the SVM approach.

1.1.1 Regression

In probabilistic regression, it is standard practice to assume that the targets values are a function of the input variables, with some additive (random) noise (ϵ_n): $t_n = y(\mathbf{x}_n, \mathbf{w}) + \epsilon_n$, where $\mathbf{x}_n$ denotes the n-th input vector, $\mathbf{w}$ the weight vector, and $y(x_n) \triangleq \sum_{j=1}^{M} w_j \phi_j(x_n)$ the predictor function. Using the independence assumption of the targets t_n, the likelihood of the complete data set can be described as:

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

where $\mathbf{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)]^T$ is the $N \times (N+1)$ design matrix and $\boldsymbol{\phi}(\mathbf{x}_n) = [1, K(\mathbf{x}_n, \mathbf{x}_1), \ldots, K(\mathbf{x}_n, \mathbf{x}_N)]^T$ is the n-th basis function. To prevent over-fitting, we introduce *a prior* that encodes a preference for less complex functions:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1}) \tag{2}$$

An individual hyper-parameter α_i is introduced for every w_i, which is the core feature responsible for the sparsity of the RVM. Next, we need to set priors over the hyper-parameters α and σ^2, which define the variance of the distribution over the weights and the training input. To obtain uniform hyper-priors on a logarithmic scale, these should be Gamma distributions with their parameters set to zero [1, 5]. Having defined the priors, we can derive the inference of RVMs by computing the posterior over all unknowns:

$$p\left(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2 \mid \mathbf{t}\right) = \frac{p\left(\mathbf{t} \mid \mathbf{w}, \boldsymbol{\alpha}, \sigma^2\right) p\left(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2\right)}{p\left(\mathbf{t}\right)} \tag{3}$$

With the use of this posterior, target predictions t_* for unseen data points $\mathbf{x}_*$ can be made as follows:

$$p(t_*|\mathbf{t}) = \int p(t_*|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2) p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t}) d\mathbf{w} d\boldsymbol{\alpha} d\sigma^2 \tag{4}$$

Since the evidence $p(\mathbf{t})$ cannot be evaluated analytically, we have to employ an approximation procedure. The posterior can be split up into $p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t}) = p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t})$, where the first factor is given by:

$$\mathcal{N}(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = (2\pi)^{-(N+1)/2} |\mathbf{\Sigma}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{w} - \mu)^T \mathbf{\Sigma}^{-1} (\mathbf{w} - \mu)\right), \tag{5}$$

with the corresponding covariance and mean:

$$\mathbf{\Sigma} = (\sigma^{-2}\mathbf{\Phi}^T\mathbf{\Phi} + \mathbf{A})^{-1} \quad \text{and} \quad \boldsymbol{\mu} = \sigma^{-2}\mathbf{\Sigma}\mathbf{\Phi}^T\mathbf{t}, \tag{6}$$

and with $\mathbf{A} = \text{diag}\,(\alpha_0, \ldots, \alpha_N)$. In the RVM, the nuisance parameter σ^2 is implicitly set. Here, we see that it can be iteratively re-estimated from the data. We approximate the second factor in the decomposed posterior by the maximum likelihood estimates of $\boldsymbol{\alpha}_{MP}$ and σ^2_{MP}, which should not violate our predictive posterior (see Tipping [1]). The learning in RVM is the (local) maximization of the marginal likelihood function $p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t}) \propto p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}) p(\sigma^2)$ with respect to $\boldsymbol{\alpha}$ and σ^2. Since we use uniform hyper-priors, we only need to maximize $p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) = \int p(\mathbf{t}|\mathbf{w}, \sigma^2) p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w}$:

$$p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) = (2\pi)^{-N/2} |\sigma^2\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T|^{-1/2} \exp\left(-\frac{1}{2}\mathbf{t}^T(\sigma^2\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T)^{-1}\mathbf{t}\right) \tag{7}$$

Unfortunately, the values for $\boldsymbol{\alpha}$ and σ^2 cannot be computed in closed form, hence, we need an iterative re-estimation. The new values can be calculated either through direct differentiation of (7), or by adopting an EM approach in which the weights are treated as latent variables [1]. We choose to calculate the values using direct differentiation for it was observed by Tipping to converge faster [1]:

$$\alpha_i^{\text{new}} = \frac{\gamma_i}{\mu_i^2} \quad \text{and} \quad \sigma^{2^{\text{new}}} = \frac{||\mathbf{t} - \boldsymbol{\Phi}\mu||^2}{N - \sum_i \gamma_i}, \tag{8}$$

where $\gamma_i = 1 - \alpha_i \Sigma_{ii}$. We can now iteratively update $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ using α_i^{new} and $\sigma^{2^{\text{new}}}$ until our convergence criterion, which monitors differences in α_i and σ^2, is met. During re-estimation, many of the α_i approach infinity, which implies that the corresponding w_i approach zero and that we can can prune the corresponding kernel functions from $\boldsymbol{\Phi}$. Now we can compute the predictive posterior distribution for an unseen $\mathbf{x}_*$ with:

$$\begin{aligned} p(t_*|\mathbf{t}, \boldsymbol{\alpha}_{MP}, \sigma^2_{MP}) &= \int p(t_*|\mathbf{w}, \sigma^2_{MP}) p(\mathbf{w}|\boldsymbol{\alpha}_{MP}, \sigma^2_{MP}) d\mathbf{w} d\boldsymbol{\alpha} d\sigma^2 \\ &= \mathcal{N}(t_* \mid y_*, \sigma_*^2) \\ &= \mathcal{N}\left(\boldsymbol{\mu}^T \boldsymbol{\phi}(\mathbf{x}_*), \sigma^2_{MP} + \boldsymbol{\phi}(\mathbf{x}_*)^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\mathbf{x}_*)\right) \end{aligned} \tag{9}$$

In this expression, t_* can be interpreted as the basis functions weighted by the posterior mean weights, and σ_*^2 as the estimated noise in the data combined with the degree of unpredictability of the weights.

1.1.2 Classification

Let us now focus on the RVM two-class classification case. The previous linear model can be generalized by applying the logistic sigmoid function $\sigma(y) = 1/(1+e^{-y})$ to $y(\mathbf{x})$. However, the weights cannot be integrated out anymore. We can find an approximation by applying the Laplace method, which estimates a Gaussian centered around the mode of the posterior. For further information, we refer to Bishop [5]. The maximum likelihood weight $\mathbf{w}_{MP}$ can be found over $p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) \propto p(\mathbf{t}|\mathbf{w}) p(\mathbf{w}|\boldsymbol{\alpha})$:

$$\log\left(p(\mathbf{t}|\mathbf{w}) p(\mathbf{w}|\boldsymbol{\alpha})\right) = \sum_n (t_n \log(y_n) + (1 - t_n)\ \log(1 - y_n)) - \frac{1}{2}\mathbf{w}^T \mathbf{A} \mathbf{w} \tag{10}$$

Since this expression has no closed-from solution, a suitable maximization method is the "iteratively-reweighted least-squares" algorithm. This algorithm is accurate due to the fact that (10) is log-concave, and thus, unimodal. According to Laplace's method, the covariance matrix is the negative inverse of the second derivative of the posterior over the weights given the mode, which is given by:

$$\nabla_{\mathbf{w}} \nabla_{\mathbf{w}} log(p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}))|_{\mathbf{w}_{MP}} = -(\Phi^T \mathbf{B} \Phi + \mathbf{A}), \tag{11}$$

where $\mathbf{B}$ is a diagonal matrix with $B_{nn} = \sigma(y(\mathbf{x}_n))(1 - \sigma(y(\mathbf{x}_n)))$, so we obtain:

$$\boldsymbol{\Sigma} = (\Phi^T \mathbf{B} \Phi + \mathbf{A})^{-1} \quad \text{and} \quad \mathbf{w}_{MP} \equiv \boldsymbol{\mu} = \boldsymbol{\Sigma} \Phi \mathbf{B} \mathbf{t} \tag{12}$$

Note that $\mathbf{w}_{MP}$ is the same as $\boldsymbol{\mu}$ in the regression case while the variance is calculated according to logistic regression instead of setting $\mathbf{B} = \sigma^{-2}\mathbf{I}$. Using these approximations, the hyper-parameters $\boldsymbol{\alpha}$ and σ^2 can be recalculated identically to the regression case. In this way, the Laplace approximation and the "iteratively-reweighted least-squares" algorithm effectively map the classification to the regression problem.

1.2 RVM in a wider context

After the introduction of RVM in 2000, there have been several suggestions for improvements to the initial algorithm. Faul and Tipping [3, 6] suggested an alternative mathematical approach to overcome the main disadvantage of RVMs, namely that training generally takes significantly longer than for SVMs. Another significant disadvantage of the original RVM method is that when the basis functions are centred on the data points, the RVM will become increasingly certain of its predictions outside the domain of the training data. To solve this problem, Rasmussen and Quiñonero-Candela have presented a method called RVM* [4]. We have implemented and analyzed both methods, which will be described in more detail in later sections. Other work on RVMs that is not analyzed in this report, but worth mentioning, includes: a fully probabilistic RVM approach using variational inference [7], multi-class RVM [8], a smoothness prior extension of the RVM [9], and RVM with adaptive kernel learning [10].

1.3 Project scope and objective

In this project, we implement the original RVM method both for regression and classification, and validate Tipping's initial results [1, 2]. Additionally, we deploy RVM on a selection of regression and classification problems and compare the results and interpretations to the ones obtained by using an off-the-shelf SVM implementation. Moreover, the fast RVM approach introduced in [3] and RVM* algorithm [4] are implemented and critically discussed. We set our main objective to compare RVMs to SVMs, and thereby focus on computational performance, sparsity and interpretability of the prediction results obtained.

2 Methods

A lot of functionality is shared between the different RVM approaches, so we opt to implement an abstract RVM class which is inherited by several child classes. In particular, we agree on implementing the RVM regressor `RVR`, RVM* regressor `RVRS` and classifier `RVC` classes. Moreover, a flag passed to the constructor determines whether to use the fast RVM implementation. The approaches for the fast implementation and RVM* are explained below. To compare results, we test our implementation against the `sklearn.svm` SVM approach [11]. We use the `sklearn.model_selection.cross_validate` function to estimate the performance of the predictive models. All code is written in Python2.7.

2.1 Re-implementing fast RVM

The main issue concerning the original RVM method is that its training phase is approximately cubic with respect to the number of basis functions (see 4.1). Tipping and Faul [3] devised a sequential sparse Bayesian learning algorithm that solves this problem. Instead of starting with m basis functions, we can start with a model containing a single basis function (i.e. the bias) [3], in which we sequentially add, modify and remove basis functions to to modify their weights and to increase the marginal log-likelihood:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) = \log p\left(\mathbf{t} \mid \boldsymbol{\alpha}, \sigma^2\right) &= -\frac{1}{2}\left[N\log(2\pi) + \log|\mathbf{C}_{-i}| + \mathbf{t}^T\mathbf{C}_{-i}^{-1}\mathbf{t}\right] \\ &= \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \frac{1}{2}\left[\log\alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right] \\ &= \mathcal{L}(\boldsymbol{\alpha}) = \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \ell(\alpha_i),\end{aligned} \tag{13}$$

where $\mathbf{C} = \sigma^2\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T = \mathbf{C}_{-i} + \alpha_i^{-1}\phi_i\phi_i^T$, $s_i = \phi_i^T\mathbf{C}_{-i}^{-1}\phi_i$, $q_i = \phi_i^T\mathbf{C}_{-i}^{-1}\mathbf{t}$, and $_{-i}$ means that the contribution of basis vector i has been removed. $\mathcal{L}(\boldsymbol{\alpha})$ has a unique maximum with respect to α_i, namely:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}, \text{ if } q_i^2 > s_i, \quad \text{and} \quad \alpha_i = \infty, \text{ if } q_i^2 \leq s_i \tag{14}$$

For the complete derivation of these expressions we refer to Tipping [3]. We then compute the posterior $\mathbf{\Sigma}$ and $\boldsymbol{\mu}$, as well as the so-called sparsity s_m and quality q_m values for all M bases. From the set of all M bases, we select a random ϕ_m and we compute $\theta_i = q_i^2 - s_i$. Now, there are three possible scenarios: (1) $\theta_i > 0$ and $\alpha_i < \infty$, (2) $\theta_i > 0$ and $\alpha_i = \infty$, and (3) $\theta_i \leq 0$ and $\alpha_i < \infty$. If (1), then ϕ_i is already in the model and we re-estimate it; if (2), we add ϕ_i to the model with updated α_i; if (3), we delete ϕ_i from the model and set $\alpha_i = \infty$. Subsequently, (if regression and estimating the noise level) we update:

$$\sigma^2 = \frac{\left\|\mathbf{t} - \mathbf{y}\right\|^2}{N - M + \Sigma_m\alpha_m\mathbf{\Sigma}_{mm}} \tag{15}$$

Lastly, we recompute $\mathbf{\Sigma}$, μ s_m and q_m. We re-iterate this procedure till convergence.

2.2 Re-implementing RVM*

One of the problems of the RVM is that when using localized basis functions (for example the Radial Basis Function (RBF)), the predictive variance given by (9) becomes the noise in the data [2, 4, 5]. That is, since for the RBF at an input $\mathbf{x}_*$ that is far away from the training data will go to zero, the predictive variance σ_{MP}^2 and the predictive mean given by (9) go to zero as well. This has as its undesirable effect that the model uncertainty is maximized around the centers of the relevance vectors, and goes to zero when moving

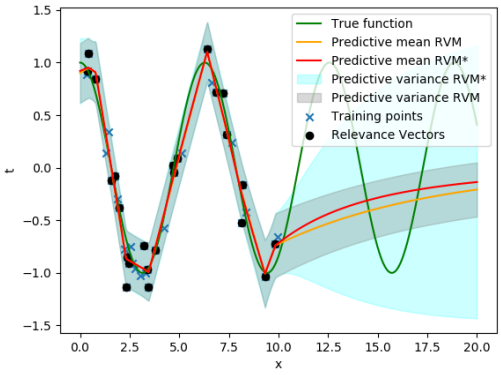

Figure 1: Plot of mean and variance of the RVM and RVM* applied to a toy problem of fitting $\cos \mathbf{x}$ with noisy observations. Here, $\sigma^2 = 0.15^2$, and $N = 40$ observations. This resulted in 24 relevance vectors. The shaded area is $\mu \pm 2\sigma$.

away from them. To fix this issue, Rasmussen and Quiñoero-Candel proposed the RVM* method [4], which introduces an additional basis function that is centered around the new input $\mathbf{x}_*$. The change happens only at test time, and training remains unchanged. $\mathbf{\Phi}$ is augmented so that $\mathbf{\Phi}_* \triangleq \begin{pmatrix}\mathbf{\Phi} & \boldsymbol{\phi}(\mathbf{x}_*)\end{pmatrix}^T$. This changes the predictive posterior's mean and variance by additive constants. As before, they are both Gaussian and so the convolution can be analytically computed as a Gaussian with mean and variance:

$$\mu^\dagger(\mathbf{x}_*) = \mu_*(\mathbf{x}_*) + \frac{eq}{\alpha^\dagger + s} \quad \text{and} \quad \sigma^{2\dagger}(\mathbf{x}_*) = \sigma_*^2(\mathbf{x}_*) + \frac{e^2}{\alpha^\dagger + s}, \tag{16}$$

where: $q = \sigma_{MP}^{-2}\boldsymbol{\phi}(\mathbf{x}_*)^T(\mathbf{t} - \mathbf{\Phi}\mu)$, $e = K(\mathbf{x}_*, \mathbf{x}_*) - \sigma_{MP}^{-2}\boldsymbol{\phi}_0(\mathbf{x}_*)\mathbf{\Sigma}\mathbf{\Phi}^T\boldsymbol{\phi}(\mathbf{x}_*)$, $s = \boldsymbol{\phi}(\mathbf{x}_*)^T(\sigma_{MP}^2 I + \mathbf{\Phi}A^{-1}\mathbf{\Phi}^T)^{-1}\boldsymbol{\phi}(\mathbf{x}_*)$, and $\alpha^\dagger$ corresponds to the precision of $\mathbf{t}$ since updating it according to (8) may lead to it being pruned. $\boldsymbol{\phi}_0(\mathbf{x}_*)$ is a vector of all kernel functions kept in the model evaluated at the new $\mathbf{x}_*$. An example is demonstrated in Figure 1, where it can be readily seen that the predictive variance for the RVM is close to being constant far from the training data, while expanding for the RVM*.

3 Results

3.1 Comparison with original results

To reproduce Tipping's RVM regression results, we apply our implementation to the `sinc` function, $t_n = f(x_n) = \frac{\sin(x_n)}{x_n}$ [1]. We use the linear spline kernel $K(x, y) = 1 + xy + xy \cdot \min(x, y) - \frac{x+y}{2} \cdot \min(x, y)^2 + \frac{\min(x,y)^3}{3}$ to test two cases, namely a 100 data points of `sinc` in $[-10, 10]$ with (1) fixed noise of 0.01^2, and (2) noise of $\sigma = 0.2$ which will be implicitly re-estimated σ^2. Similar to Tipping's results, in case (1) we end up with 10 relevance vectors (see Figure 2a), which clearly outperforms the SVM approach (39 support vectors [12]) in terms of sparsity. Case (2) yields a model with 11 relevance vectors, and the standard deviation is automatically estimated to be $\sigma = 0.205$ (see Figure 2b).

The RVM classification performance is tested based on a number of data sets including `Titanic`, `Banana`, `Breast Cancer` and `German`, which were also used by Tipping [2]. Figure 3 shows an example of RVM classification, including the learned decision boundary and the relevance vectors. In Table 1, the reader can compare the performance results of these data sets with Tipping's results [2]. The relative performance of SVM and RVM is highly similar, however, there are differences in the number of relevance vectors and the absolute errors. This is due to the fact that we fix the kernel, pruning thresholds and convergence thresholds to a constant value while Tipping reported the result that is the best result after experimentation with the parameters [2].

3.2 Extending on the original method

Referring to Table 1, we notice that SVM outperforms or equals RVM in all categories. Interestingly, the RVM scores closer to the SVM for classification tasks than for regression tasks; the difference in error is

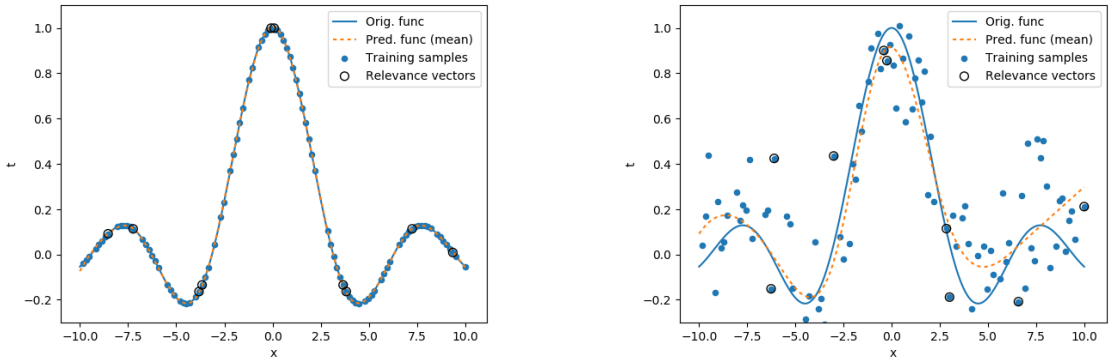

Figure 2: (left) RVM performed on the sinc(**x**) function with fixed noise $\sigma^2 = 0.01^2$; (right) RVM performed on the sinc(**x**) function with unfixed added noise $\sigma^2 = 0.2^2$.

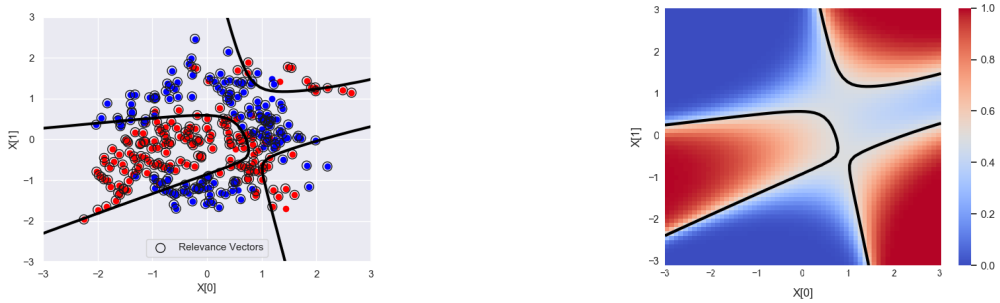

Figure 3: This figure shows an example of RVM classification on the `banana` data set [13]. (left) the training data, the relevance vectors and the decision boundary of the classification model; (right) the posterior probability given by the RVM model.

more noticeable. The RVM* performed similarly to the RVM and fast RVM, which is reasonable considering the goal of RVM* is to remove the degeneracy in predictive variance. The findings for classification mimic those in [2], however, the results are the opposite to [2] when looking to regression. What is most noticeable is the sparsity of the results. While RVM* must use all basis functions in prediction, RVM and its fast implementation are in some cases orders of magnitude more sparse than SVM. In some cases, for example in the linear data set, data is separated by a hyperplane in $\mathbb{R}^2$, so it makes sense that we should only need a few basis functions out of the original $N + 1$. The fast implementation gives similar results to the slower version, while being far more sparse. This is most likely due to hyper-parameter tuning (the threshold for pruning α) and the criteria for convergence. With more time, one would expect tuning these parameters to yield similar sparsity results.

Referring to Table 2, the increases in fitting time for the RVM and SVM can be explained in part by the complexity analysis in 4.1. Generally, the SVM is fastest of the three, followed by the fast RVM, and finally the original RVM. Although it is not fair to compare the two directly, since the SVM implementation most likely is written in low-level C and optimized well, the time increases reflect the time complexity (see 4.1).

4 Discussion

4.1 Complexity

One of the major disadvantages of the RVM is the computational complexity of the learning process. The computation of the posterior covariance matrix requires an inverse operation that is in $\mathcal{O}(M^3)$, where M is the number of basis functions. Although the initial number of $M = N + 1$ basis functions, where N denotes the number of data points, is drastically reduced through pruning, it is a costly operation. In contrast to the standard approach, the fast implementation of the RVM starts with a single basis function and iteratively adds, re-estimates, and deletes bases based on whether the newly introduced basis improves the marginal

Summary of Error Results and Number of Vectors

			errors				*vectors*			
Data set	**N**	**d**	**RVM**	**RVM***	**RVMF**	**SVM**	**RVM**	**RVM***	**RVMF**	**SVM**
cos	30	1	1.41	1.42	1.42	0.084	4.4	-	1.6	13.0
sinc	100	1	0.33	0.33	0.33	0.15	12.4	-	3.8	30.0
Friedman 2	300	4	5.2E8	1.8E10	1.2E10	1.6E5	117.0	-	41.6	240.0
Friedman 3	300	4	5.2E8	1.8E10	1.2E10	1.6E5	117.0	-	41.6	240.0
Boston	506	13	548.10	2267.51	3089.71	94.11	205.8	-	26.8	400.2
Airfoil	1503	5	9347.51	8534.35	7437.61	48.31	253.2	-	70.4	119.2
linear	100	2	0.38		0.26	0.09	80.0		1.8	71.0
Thyroid	140	5	0.28		0.16	0.04	112.0		3.4	63.4
Titanic	150	3	0.29		0.29	0.25	120.0		2.6	55.4
Banana	400	2	0.46		0.43	0.09	318.6		0.8	117.4
Waveform	400	21	0.08		0.19	0.33	316.4		4.4	320.0
Breast Cancer	569	30	0.37		0.37	0.37	452.0		11.4	455.2
German	700	20	0.31		0.3	0.31	553.6		4.4	560.0

Table 1: The six regression and seven classification data sets explored, along with the average Mean Squared Error (MSE), and the average number of support/relevance vectors for the first six, which are regression. The last seven are classification and the the error is $1 -$ accuracy. N is the number of samples, and d is the number of features. 5-fold cross validation was used to determine all results. The number of relevance vectors for the RVM* is identical to the number for the RVM. The RVM* was not tested on classification. For all data sets (and tables) the RBF kernel with $\gamma = 2$ was used, and the same threshold values for removing α and determining convergence were fixed.

			Fit Time		
Data set	**N**	**d**	**RVM**	**RVMF**	**SVM**
cos	30	1	0.042	0.027	0.00036
sinc	100	1	0.32	0.29	0.00068
Friedman 2	300	4	2.51	2.90	0.0024
Friedman 3	300	4	2.33	2.41	0.0019
Boston	506	13	6.59	6.93	0.0067
Airfoil	1503	5	67.86	64.20	0.047
linear	100	2	2.51	1.73	0.0013
Thyroid	140	5	5.33	6.43	0.00082
Titanic	150	3	5.59	3.44	0.00066
Banana	400	2	39.04	7.057	0.0019
Waveform	400	21	300.40	30.26	0.0094
Breast Cancer	569	30	80.54	123.26	0.010
German	700	20	130.01	47.55	0.031

Table 2: Fit time (in seconds) of our implementations of [1, 2] (RVM) and [3], compared to `sklearn.svm` (Fast RVM) [11]. The time is computed for both regression data sets (the upper 6) and the classification data sets (the bottom 7). In this table N is the number of samples, and d is the number of features.

log-likelihood. This significantly speeds up the training phase, but the downside of this algorithm is that the α_i can only be updated one after another, which results in a "greedy" optimization approach. For the classification case we have the additional "iteratively-reweighted least-squares" approximation procedure, which is run until convergence and requires additional cost-intense matrix operations (although it stays in the same complexity class).

The `sklearn` SVM implementation [11] is in the complexity class between $\mathcal{O}(N_{features} \times N_{samples}^2)$ and $\mathcal{O}(N_{features} \times N_{samples}^3)$ depending on the cache usage and hence it has a similar complexity [14]. However, due to the fact that the offset costs are smaller, on average SVM outperforms RVM on large data sets. From a computational point of view, the RVM has the advantage that it does not need to perform any

cross-validation to obtain the nuisance parameters C or ϵ which define the error margins in the classification and regression case for the SVM. However, this benefit vanishes for larger training input (for $N > 700$ [1]) and is thus not applicable for a large range of real-world data sets.

4.2 RVM in practice

The RVM results are relatively similar compared to the SVM results for classification, which is admirable, considering the RVM provides a full predictive distribution and not just a predictive point. While the results are not as good as in [2] for regression, it is most likely due to non-optimal stopping criteria as well as difficulty tuning hyper-parameters.

4.3 Concluding remarks

The RVM achieves respectable solutions in classification when compared to the SVM. In the literature, the RVM also achieves good if not better results than the SVM for regression tasks. Besides, the RVM provides sparse solutions, which for some applications is not only useful but required. In addition, the RVM provides a full predictive posterior and not just a single prediction point or class. Although the RVM has unwanted small variance for extrapolated test samples, it can be augmented with the RVM*.

However, the RVM is slow in training and the benefit of sparsity seems to come at the expense of meaningful variance and accuracy [4]. For larger data sets, training may become intractable. Furthermore, if the RVM* is used the sparsity is lost, as all basis functions are needed to compute the predictive distribution. This leads to an increased memory and time-complexity for computing predictions [4].

Ultimately, the RVM is suited best for small applications where a predictive distribution is valuable and it is known a priori that test inputs will not be too far from the center of training inputs to avoid a vanishing model variance. For larger problems, due to the large training time and sacrifice of accuracy for sparsity, other methods (such as the SVM) are preferable.

References

[1] M. E. Tipping, "The relevance vector machine," in *Advances in neural information processing systems*, pp. 652–658, 2000.

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

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

[4] 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.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

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

[7] C. M. Bishop and M. E. Tipping, "Variational relevance vector machines," in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, (San Francisco, CA, USA), pp. 46–53, Morgan Kaufmann Publishers Inc., 2000.

[8] I. Psorakis, T. Damoulas, and M. A. Girolami, "Multiclass relevance vector machines: sparsity and accuracy," *IEEE Transactions on neural networks*, vol. 21, no. 10, pp. 1588–1598, 2010.

[9] A. Schmolck and R. Everson, "Smooth relevance vector machine: a smoothness prior extension of the rvm," *Machine Learning*, vol. 68, no. 2, pp. 107–135, 2007.

[10] D. G. Tzikas, A. C. Likas, and N. P. Galatsanos, "Sparse bayesian modeling with adaptive kernel learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 6, p. 926, 2009.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] V. Vapnik, *Statistical learning theory. 1998*, vol. 3. Wiley, New York, 1998.

[13] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017.

[14] "Support vector machines." https://scikit-learn.org/stable/modules/svm.html#complexity. Accessed: 2019-01-14.