DD2434 Machine Learning, Advanced Course
Sparse Modelling - The Relevance Vector Machine

Walid Abdul Jalil
Shadman Ahmed
Oguzhan Ugur

walidaj@kth.se
shadmana@kth.se
Oguzhanu@kth.se

January 23, 2018

Course responsible: Pawel Herman and Jens Lagergren

Introduction

Advances in computing power together with cheaper and more widely available electronics have resulted in an explosion of different applications that handle massive amounts of data, such as facial recognition software. As such, people focusing on research and development are always trying to find faster and more memory-efficient models. This is where sparse models come in. Sparse models are faster and more memory-efficient because, among all the coefficients/parameters that describe a model, only a few are non-zero.

One method is the RVM (Relevance Vector Machine). RVM is a method that uses the same form as the widely-known SVM (Support Vector Machine), but provides probabilistic outputs. It supposedly does not suffer from the same disadvantages as the SVM, such as the requirement to estimate a trade-off parameter or the need to use "Mercer" kernel functions. Our task is to try to validate some of the results by M.E. Tipping in his 2000 article ("The Relevance Vector Machine") and to critically examine Tipping's arguments in favor RVMs.

Support Vector Machines

In supervised learning we are given a set of input vectors $\{\mathbf{x}_n\}_{n=1}^N$ along with corresponding targets $\{t_n\}_{n=1}^N$. Which can either be real valued in the case of regression or integers in the case of classification. Usually some function $y(\mathbf{x})$ is used to predict the output. A common form is:

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

where $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_M(\mathbf{x}))^T$ are general non-linear basis functions. This form is linear in the parameters $\mathbf{w} = (w_1, w_2, ..., w_M)^T$.

When dealing with real world data, the presence of noise means that the main challenge to a good model is to avoid overfitting. The SVMs (Support Vector Machines) are supervised learning models that are quite resistant to overfitting since it uses regularization. The very reason SVMs were created was because Perceptron learning overfitted when cutting a plane through the data in some higher dimension without a margin. By contrast the support vector machine makes predictions based on functions of the form

$$y(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{M} w_i K(\mathbf{x}, \mathbf{x}_i) + w_0 \tag{2}$$

where $K(\mathbf{x}, \mathbf{x}_i)$ is a kernel function, which represents a dot product of input data points mapped into the higher dimensional feature space by some transformation. In that higher dimensional feature space, a hyperplane will maximize the margin between the the classes(in case of classification), while minimizing the number of classification errors. This results in a relatively sparse model because it is only dependent on a subset of the kernels. The hyperplane is defined by the training points that are either on the margin or on the wrong side of it.

Relevance Vector Regression

Tipping adopts a Bayesian approach to learning and introduces something called the Relevance Vector Machine. A probabilistic sparse kernel that is identical in form to the SVM. Tipping introduces a prior over the weights w_i defined by a set of hyperparameters. The most probable values of the weights are then iteratively estimated/updated from the data. Tipping argues that sparsity is achieved because in practice the posterior distributions of many weights will be infinitely peaked around zero. Finally, he argues that the RVMs are capable of the roughly the same performance to an equivalent SVM but with far fewer kernel functions. Some of the mathematical steps that are necessary for RVM regression are described below.

Setting the prior

The first thing Tipping does, is to introduce a prior that encodes a preference for smoother, less complex functions as a way of avoiding overfitting. He chooses a zero-mean Gaussian prior over $\mathbf{w}$

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathcal{N}(w_i|0, \alpha_i) \tag{3}$$

where α is a vector of N + 1 hyperparameters. The main difference with RVM regression is the fact that there is a seperate hyperparameter α_i for each weight parameter w_i. This has a moderating effect on the prior.

Tipping also introduces a hyperprior over $\boldsymbol{\alpha}$ and over the noise variance $\beta = \sigma^2$:

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N} Gamma(\alpha_i|a, b) \tag{4}$$

$$p(\beta) = Gamma(\beta|c, d) \tag{5}$$

He then proceeds by setting these parameters to zero, $a = b = c = d = 0$. Thus getting uniform hyperpriors. This means that results will be independent of linear scaling of $\boldsymbol{t}$ and basis function outputs.

Maximizing the evidence w.r.t. to these hyperparameters, most of them go to infinity. That means that the corresponding weight parameters will have posterior distributions concentrated around zero. The α will then go to zero and their corresponding basis functions will not be used (deeming them to be irrelevant). So they are "pruned" out and this is what results in a sparse model. This is a type of something termed "ARD" (Automatic Relevance Determination).

After defining the prior, the posterior is computed over the parameters $\boldsymbol{w}$ given $\boldsymbol{t}, \boldsymbol{\alpha}, \sigma^2$:

$$\begin{aligned} p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}, \sigma^2) &= \frac{p(\boldsymbol{t}|\boldsymbol{w}, \sigma^2)p(\boldsymbol{w}|\boldsymbol{\alpha})}{p(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma^2)} \\ &= (2\pi)^{-(N+1)/2}|\boldsymbol{\Sigma}|^{-1/2}\ exp\Big[-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{\mu})\Big] \end{aligned} \tag{6}$$

where the posterior covariance and mean are:

$$\boldsymbol{\Sigma} = (\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A})^{-1} \tag{7}$$

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t} \tag{8}$$

with: $\boldsymbol{A} = diag(\alpha_1, ..., \alpha_N)$.

By approximating $p(\boldsymbol{\alpha}, \sigma^2|\boldsymbol{t}) \approx \delta(\boldsymbol{\alpha}_{MP}, \sigma^2_{MP})$. The parameters that need to be learned can be calculated by the maximization of $p(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma^2)$. Also known as type-II maximum likelihood method(evidence approximation):

$$\begin{aligned} p(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma^2) &= \int p(\boldsymbol{t}|\boldsymbol{w}, \sigma^2)p(\boldsymbol{w}|\boldsymbol{\alpha})\ d\boldsymbol{w} \\ &= (2\pi)^{-N/2}|\sigma^2\boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T|^{-1/2}\ exp\Big[-\frac{1}{2}\boldsymbol{t}^T(\sigma^2\boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T)^{-1}\boldsymbol{t}\Big]. \end{aligned} \tag{9}$$

Finally the hyperparameters must be obtained by iterative methods. α is given by:

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2} \tag{10}$$

where μ_i is the i-th posterior mean weight. and γ_i is:

$$\gamma_i = 1 - \alpha_i\Sigma_{ii}, \tag{11}$$

where Σ_{ii} the i-th diagonal element of the posterior covariance. The noise variance is obtained by the iterative method:

$$(\sigma^2)^{new} = \frac{||\boldsymbol{t} - \boldsymbol{\Phi\mu}||^2}{N - \Sigma_i \gamma_i} \tag{12}$$

When doing these calculations, we see that a lot of α_i tend to become so numerically large that computers cant manage them. For those α_i, w_i will be set to zero, thus producing a sparse model.

Results

The 'sinc' function

The example of the sinc function mentioned in Tipping's paper, *"The Relevance Vector Machine"* ,was implemented into our own algorithm and a result close to Tipping's were obtained. The x-values used in this implementation was generated in python by using numpy.linspace where 100 uniformly-spaced noise free samples were generated on a interval [-10,10].The target variables were obtained by using the sinc function $t = \frac{1}{|x|} sin(|x|)$. Both support vector regression and relevance vector regression were performed with a spline kernel and tested with and without noise. The noise type we used was a Gaussian of standard deviation 0.1. The noise variance for the relevance vector machine was fixed at 0.01^2 and the needed parameters were iteratively estimated for RVM.

The plots and the results are depicted below. Figure 1 and Figure 2 represents the plots without noise for SVM and RVM, Figure 3 and Figure 4 represent the plots with Gaussian noise. The SVM approximator on Figure 1 requires 41 support vectors, while RVM only requires 9 relevance vectors, the errors are 0.079 and 0.0029 respectively. The RVM gives a sparser model than the support vector machine which matches Tipping's statement,*"The most compelling feature of the RVM is that, while capable of generalization performance comparable to an equivalent SVM, it typically utilizes dramatically fewer kernel functions"*
The noisy regression of SVM requires 32 support vector and has an error of 0.164364 while RVM requires 6.61 relevance vectors and has an error of 0.0936. The parameters C and ϵ were tuned by using a grid search for SVM. For the RVM, the noise variance(σ^2) and alpha(α) were estimated iteratively.

As told earlier, the relevance vector machine utilizes fewer kernel functions which is also possible to see from the results we obtained and the main reason of this is the ARD.The posterior distribution of many of the parameters is infinitely peaked around zero and gets pruned.

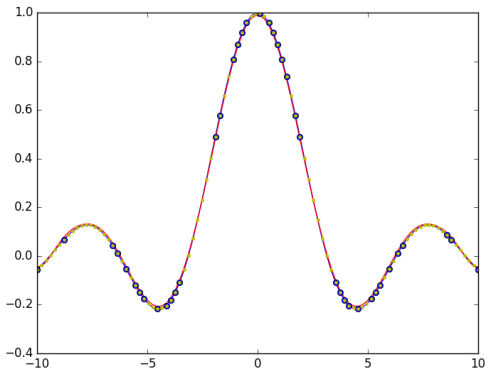

Figure 1: SVM

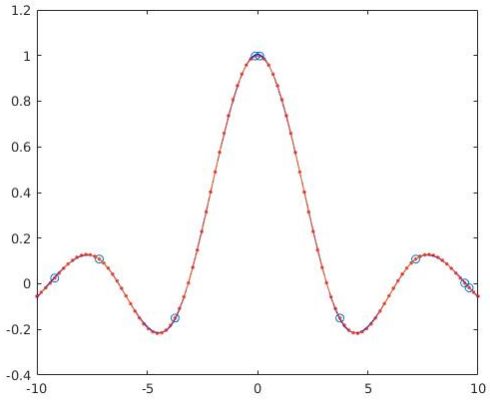

Figure 2: RVM

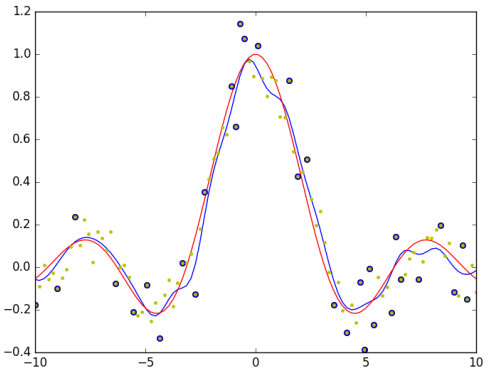

Figure 3: SVM Noise

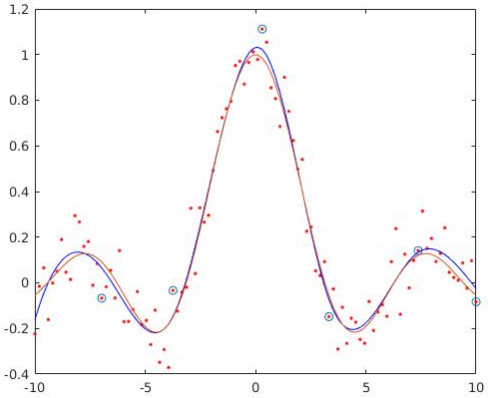

Figure 4: RVM Noise

The blue line on the plots are the estimated function, the red line is the true function and the bold circles in blue are the vectors. The data points are yellow on the figures at left hand-side and red on those on the right hand-side

Benchmark comparison

The Benchmark section in Tipping's paper compares different 'benchmark' datasets but we decided to look into Friedman 1, sinc(Gaussian noise) and a decaying-sinusoidal function. The Friedman dataset was generated by using sklearn.datasets with a Gaussian noise of unit standard deviation added to $y(\mathbf{x})$. The results for Friedman 1 were obtained by averaging over 100 repetitions, where 240 randomly generated training examples were utilized. A Gaussian kernel was used, and its parameter 'r' was decided by 5-fold-cross-validation. The C value was decided by using a cross validation scoring method from the library sklearn, a plot of it is depicted on Figure 5.
The data generated from the sinc function follows the procedure described on the previous section. The data generated from the decaying-sinusoidal function had 100 equally-spaced x-values in [-10,10] as the sinc function. A Gaussian noise with standard deviation 0.1 was added. The obtained results are shown on the table below:

Table 1: Benchmark comparison

			SVM		**RVM**	
	N	**d**	**vectors**	**error**	**vectors**	**error**
Sinc (Gaussian Noise)	100	1	32.54	0.164	6.61	0.0936
Decaying-sinusoidal (Gaussian noise)	100	1	39.25	0.053	6.270	0.094
Friedman 1	240	10	126.3	5.5	32.76	4.19

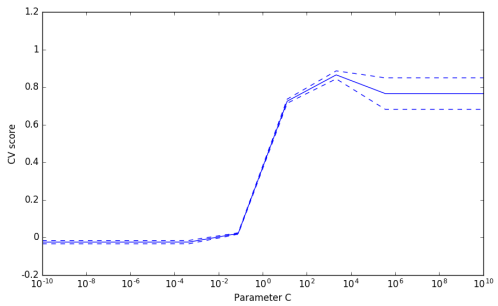

Figure 5: Cross validation score for the C parameter

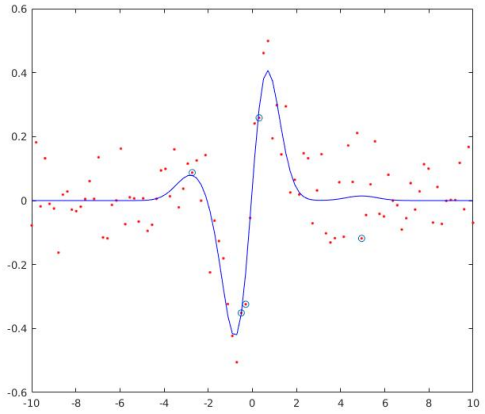

Figure 6: Decaying-sinusoidal plot with 6 relevance vectors

Discussion

Scaling

Tipping argues that the principal disadvantage of RVMs is that training involves optimizing a nonconvex and training time can be longer than for an SVM. If we have M basis functions, the RVM requires inverting a M x M matrix, which has a complexity of $O(M^3)$ while SVM training generally requires $O(M^2)$. In the case of classification case, inverting the Hessian can be computationally heavy. This is very problematic for for large data sets. In Table 2 we implement a scaling test for both SVM and RVM using the function $y(x) = sin(x)e^{-x^2}$. As can be seen in the table, the time for RVM increases considerably. At some point when we have large data sets it may not be feasible to use the RVM. The error remains the same for RVM regardless of data-set size. The number of relevance vectors increases but remains sparse. Figure 7 shows a plot of runtime vs number of data points in a logarithmic scale. It could be argued that when the data sets are large, the resulting sparsity is prioritized over the computation time. If that is the case, then this method is still recommended.

Data points, N	SVM time[sec]	RVM time[sec]	SVM vectors	RVM vectors	SVM error	RVM error
100	0.100	5.900	39.25	6.270	0.053	0.094
200	0.346	46.04	72.55	7.770	0.034	0.095
300	0.609	203.2	104.7	8.949	0.033	0.097
400	0.923	615.4	135.8	12.03	0.029	0.098
500	1.357	1477	167.5	12.85	0.027	0.099

Table 2: Table of runtime, support and relevance vectors and RMSE. With function $y(x) = sin(x)e^{-x^2}$.

Comparing the methods by looking at the results

One thing is absolutely clear from our results, every time we used the relevance vector machine method, the result was a sparse model. The SVM uses far more support vectors than the number of relevance vectors the RVM uses. By looking at the Benchmark Table (Table 1), one can see that RVM was significantly more sparse but with comparable errors. In some cases the errors were smaller and in others larger, but the errors were not of orders of magnitude different. Therefore we find Tipping's argument that RVM gives more compact models but with similar errors to be correct. So while the training phase for the RVM is longer for large data sets, the resulting

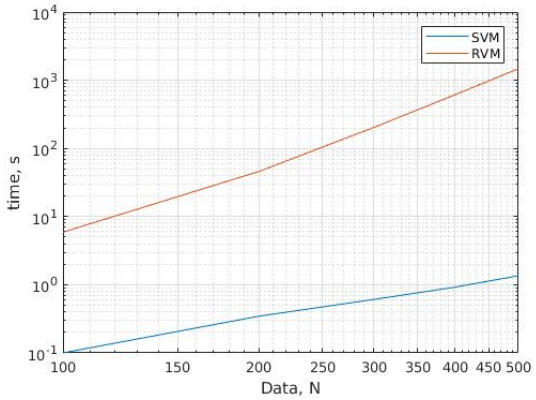

Figure 7: Log-space plot of runtime and number data points from Table 2.

sparsity means that they are easier to process than the SVM. We did not need to estimate the trade-off parameter 'C', this made things a lot easier. The fact that the predictions are probabilistic means that we can capture uncertainty as well as adapt our model to to varying class priors.

We were very uncertain about our cross-validation procedures, our results were not always the same as Tipping's. But the general conclusion remains. RVM is significantly more sparse than SVM. We did not get a chance to test wether Kernel's that do not satisfy Mercer's condition could be applied.