Bootstrapping the Support Vector Machine

Thomas Goerttler, Christian Koopmann, Patricia Craja

Department of Computer Science,
Machine Learning Group

Humboldt-University of Berlin, Germany
thomas.goerttler@gmail.com, c.k.e.koopmann@gmail.com,
Patricia.craja@gmx.de

Abstract

The goal of this project is the analysis of the variance of Support Vector Machines (SVMs) and the relationship between this variance and other important aspects of the SVM. We provide a characterization of the variance of SVMs using the minimal distance of prediction points to the decision boundary and apply the bootstrap method in order to estimate the uncertainty of the prediction rule. The implementation of the Bootstrap (in Python) is done in a parallelized manner. Finally it will be shown how this variance is correlated with other characteristics of the SVMs, such as the regularization parameter and the number of support vectors, and with the balance of the training data set. This impact is analyzed for both Linear and Gaussian Kernel SVMs. Doing this for both simulated as well as real data sets produced very similar results. In this paper we therefore concentrate on showing the results obtained for simulated data.

Introduction

Support Vector Machines are one of the most successful methods of Machine Learning ([**?**]). SVMs have many merits that distinguish them from other machine learning algorithms, including the speed of calculation, and the use of only two tuning parameters. Given a binary training data set, an SVM learning algorithm builds a model that predicts the unknown class for new input data, making it a non-probabilistic binary linear classifier ([**?**]). SVM is based on a minimization problem, which seeks for a hyperplane in the feature space that optimally separates data points of two clusters. By reducing non-linear complex decision problems to linear problems through application of the Kernel-Trick, they represent a computationally efficient way to tackle these problems. The dimension of the feature space is controlled by the choice of a specific Kernel function. Common Kernels are the Linear Kernel, the Gaussian Kernel (RBF) and the Polynomial Kernel.

SVMs based on certain Kernels (e.g. Gaussian RBF Kernel) are non parametric methods. Since the distribution of the underlying data is generally unknown, so is the finite sample distribution of these methods ([**?**]).

There has been considerable research on the asymptotic distribution of SVMs, which have been shown to be asymptotically normally distributed under certain conditions. An alternative idea to estimate these distributions is to use Efrons empirical bootstrap ([**?**]). The idea behind this method is to repeatedly draw samples with replacement from the full data according to the empirical distribution function of the data. Through the repeated calculation of the statistic of interest one can get an estimate of its distribution. For the SVM this estimate has been shown to be consistent under relatively mild conditions.

In the first section we will give an overview of the Problem to calculate the variance of the SVM. Then we will explain how we implemented the Bootstrap and the Data Simulation in order to estimate this variance. Once we estimated the variance we can finally analyze the influence of different parameters on this variance for both Linear and Gaussian Kernels.

Problem

In contrast to probabilistic classifiers which provide classification with a degree of certainty, SVMs only predict the most likely class that the sample should belong to.

Since no adequate distribution theory exists, we apply the bootstrapping method to the SVM algorithm, i.e. drawing random training samples with replacement from the full training data set to train different SVMs. Through the repeated calculation of the predictions of the test data set we can estimate the prediction variance, which is an indicator of the degree of certainty of the SVM.

Predictions however are only binary variables. Since they might be identical across all bootstrap samples, we use real valued substitutes such as the minimal distance of each prediction point to the decision boundary (Figure 1) or the estimated probability of each prediction point to belong to the predicted class. After testing both substitutes we decided to use the minimal distance to the decision boundary as it is easier to understand how these values are calculated.

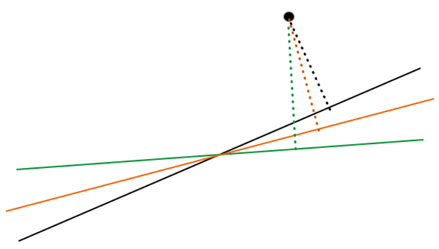

Fig. 1: The minimal distance of a prediction point to each hyperplance is considered.

Implementation

Bootstrapping

In order to calculate the variance of the predictions we use bootstrap samples. We start by training the SVM on the full training data and will call this SVM the *Full SVM*. Then we draw N random bootstrap samples with replacement ([**?**]) from the full training data and train the SVM N times on each bootstrap sample in a parallelized manner. That way we get N different SVMs that predict for each input of the test data of size n to which class it belongs. Figure 2 illustrates the different results obtained for the linear Kernel for perfectly separated data (a and c) and not perfectly separated data (b and d), as well as for a smaller size of the training data (a and b) and a bigger one (c and d). We observe that the variance is smaller if the data is perfectly separated and if the training data contains more observations.

The n distances of each point in the test data set from the decision boundary can be seen as random variables. For the N SVMs we obtain N different values of these random variables and can calculate their variance. We take the average of these n variances as an indicator for the variance of the *Full SVM*. Therefore our Bootstrap Method uses as input the training data set, the test data set, SVM-Parameters, and the number N of bootstrap replications and it returns the *Full SVM*, its number of support vectors as well as its variance.

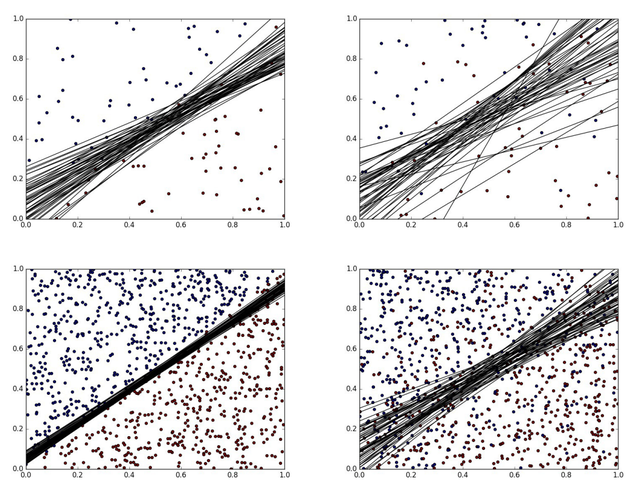

Fig. 2: In each picture 100 hyperplances are plotted based on samples. In (a) and (c) the data are perfectly separated whereas in (b) and (d) small errors occur. In (a) and (b) we used 100 observations in the training data and in (c) and (d) we used 1000.

Parallelization: In order to make the bootstrapping efficient, it is parallelized. A pool of processes is created to carry out tasks submitted to it using the Python `Pool` class. Each process trains a certain numbers of SVMs on bootstrap samples and calculates the distances for each test point. The results from each process are then collected in an array which is used to calculate the statistics mentioned above. Since the training of each SVM is independent of the others, the parallelization of this algorithm is relatively straight forward.

Data Simulation

For the Data Simulation we use two different approaches, the Hyperplane- and the Centroid- Approach. For both we randomly draw n observations for each of the input variables x from a normal distribution. The Hyperplane-Approach calculates the labels using the formula $y = sign(c + w^T x + error)$, given the Hyperplane-Parameter w, a constant c as well as the error distribution, whereas the Centroid-Approach calculates the labels using the formula $y = sign(c + a_1 * (d(x, z_1)^{-1} ... + error)$ given the Centroid- Locations z_i, the Centroid- Parameters a_i, a constant c and a distance function $d(a, b)$. To change the number of support vectors of a data set the variance of the error is modified, whereas the intercept controls the balance of the data set.

Results

In this section we will show how the prediction variance is correlated with other aspects of the SVM, such as the regularization parameter C, the balance of the training data set and the number of support vectors of the original *Full SVM*. Analyzing this impact for both Linear and Gaussian Kernel SVMs produced the following results:

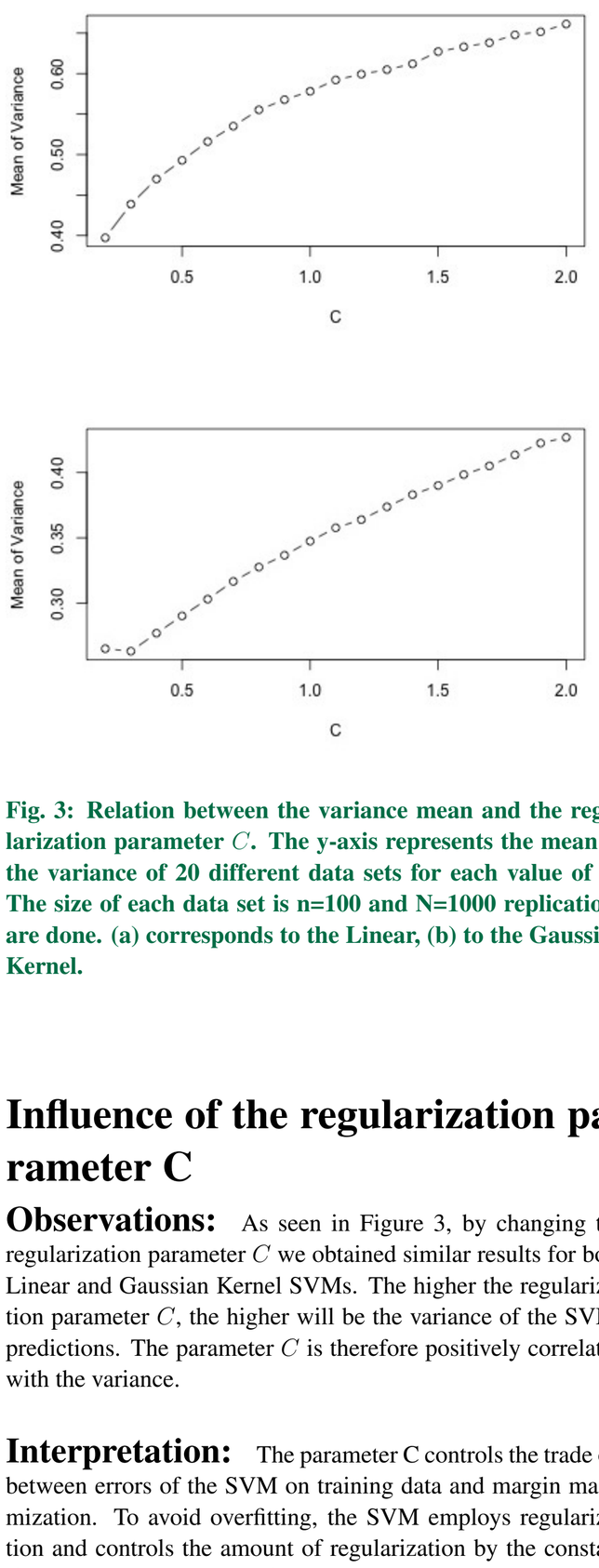

Fig. 3: Relation between the variance mean and the regularization parameter C. The y-axis represents the mean of the variance of 20 different data sets for each value of C. The size of each data set is n=100 and N=1000 replications are done. (a) corresponds to the Linear, (b) to the Gaussian Kernel.

Influence of the regularization parameter C

Observations: As seen in Figure 3, by changing the regularization parameter C we obtained similar results for both Linear and Gaussian Kernel SVMs. The higher the regularization parameter C, the higher will be the variance of the SVM-predictions. The parameter C is therefore positively correlated with the variance.

Interpretation: The parameter C controls the trade off between errors of the SVM on training data and margin maximization. To avoid overfitting, the SVM employs regularization and controls the amount of regularization by the constant C ([**?**]). A high value of C corresponds to low regularization,

i.e. lower size of slack variables and therefore higher overfitting, which means that the SVM fits the training data too well but does not generalize to new data and therefore the variance of the predictions of our test data rises.

Figure 3 shows this relation for values between 0 and 2 of the parameter C for Linear and Gaussian Kernels. Because each value of C corresponds to a different *Full SVM*, we calculated the average of 20 SVM variances for each value of C, i.e. 20 test data set variances as explained at the beginning (using N=1000 Bootstrap Replications, and n=100 data points).

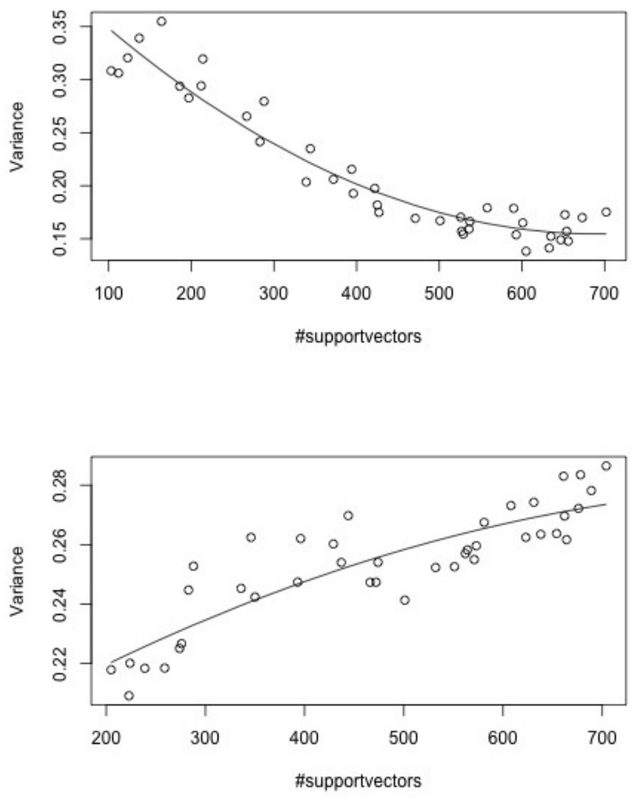

Fig. 4: Relation between the variance and the number of support vectors. The data set size is n=1000 and N=500 replications are done. (a) corresponds to the Linear, (b) to the Gaussian Kernel.

Influence of the number of support vectors

Observations: In Figure 4 it can be seen, that changing the number of support vectors for the *Full SVM* we obtained different results for Linear and Gaussian Kernels. For the Linear Kernel, the variance of the SVM decreases with an increase in support vectors, whereas for the Gaussian Kernel the variance increases with more support vectors. The number of support vectors is therefore negatively correlated with the variance of the Linear Kernel and positively correlated with the variance of the Gaussian Kernel.

Interpretation: The number of support vectors controls the slack variables of the soft margin SVM. A high number of support vectors implies larger margins and higher slack variables. Training the Linear SVM on the bootstrap samples is therefore more robust and the variance of the decision boundary as well as the distance from each point of the test data set to the decision boundary has smaller variation. For Gaussian Kernels, a higher number of support vectors might imply overfitting on the training data set and therefore bad generalization to new data which could be the reason why the variance of the predictions of our test data rises.

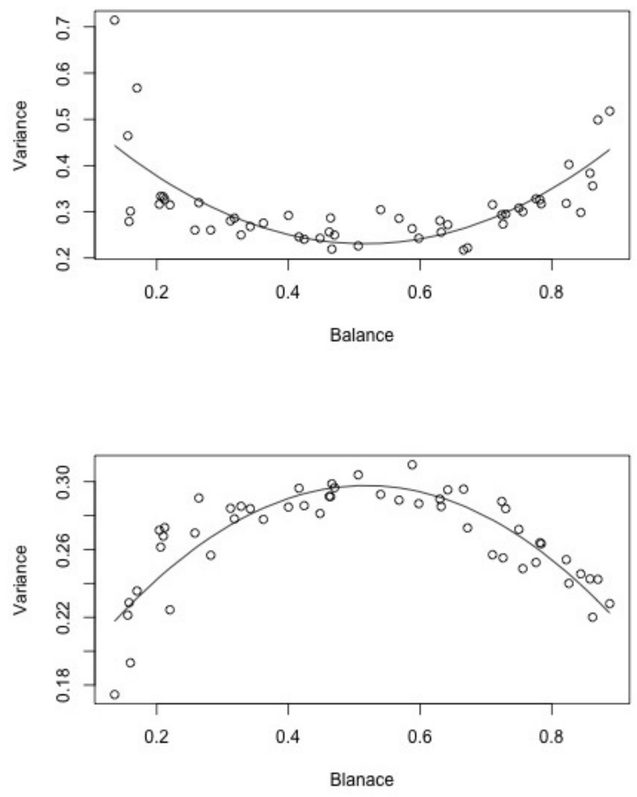

Fig. 5: Relation between variance and balance of the data. The data set size is n=500 and N=1000 replications are done. (a) corresponds to the Linear, (b) to the Gaussian Kernel.

Influence of the balance of the training data set

Observations: Changing the balance of the training data set, from which the samples are taken, we obtain different results for the Linear and Gaussian Kernels, as seen in Figure 5. As expected the resulting graph is symmetric around 0.5 for both kernels. For the Linear Kernel, the variance of the SVM rises when the data is unbalanced and it is minimal for perfectly balanced data. However, for the Gaussian Kernel the maximal variance is attained for perfectly balanced data, and it drops for unbalanced data.

Interpretation: In the linear case the variance is especially high, when one class is very small compared to the other class. The location of the linear hyperplane is on the few points from the smaller class. Depending on which points are sampled from this class, the decision boundary and therefore the distance of test points to this boundary might vary dramatically. The Gaussian Kernel shows exactly the opposite results. This occurs because the Gaussian Kernel tends to overfit the data on the training data sets ([**?**]). We have taken 1000 samples from the data set. If the data is perfectly balanced, more different samples of the smaller group are possible. Therefore the hyperplanes differ more. If the smaller class is very small, the sample from the smaller class can not differ that much from each other. Nevertheless it can be obtained, that the effect is extremely high in the tails and rather small between a balance of 0.3 to 0.7 .

Conclusions

- Calculating the variances by using bootstrap samples seems to be a quite good idea to calculate a variance.
- The variance of distances of test points seems to be a good indicator for the robustness of the SVM as shown in Figure 2 .
- The results of changing the C-parameter are not surprising but confirm the strength of the variance estimator.
- Especially interesting and to some degree counterintuitive are the different effects of support vectors and data balance on the Kernels.
- Overall the bootstrapping method has once again shown its value for situations where classical methods of statistical inference are not available.

Forthcoming Research

Further interesting analysis would be to look for the influence of the dimensionality on variances. The influence of other tuning parameters such as the gamma parameter of the RBF-Kernel or the degree of Polynomial Kernels could be interesting areas of inquiry. Also in this context we could only give rough explanations for the findings. More accurate explanations might be found when analysing one of the aspects in more detail. Also despite the parallelization of the algorithm the procedures are still computationally very intensive. Using more sample computing resources (and thereby increasing both the number of observations as well as bootstrap samples) would probably lead to even more reliable results.