Department of Applied Econometrics Working Papers
Warsaw School of Economics
Al. Niepodleglosci 164
02-554 Warszawa, Poland

Working Paper No. 2-09

Support vector machines with two support vectors

Marcin Owczarczuk
Warsaw School of Economics

Paper submitted to *Statistica Sinica*

Support vector machines with two support vectors

Marcin Owczarczuk

Warsaw School of Economics

Abstract: In this article we present a new class of support vector machines for binary classification task. Our support vector machines are constructed using only two support vectors and have very low Vapnik-Chervonenkis dimension, so they generalize well. Geometrically, our approach is based on searching of a proper pair of observations from different classes of explained variable. Once this pair is found the discriminant hyperplane becomes orthogonal to the line connecting these observations. This method deals well with data sets with large number of features and small number of observations like gene expression data. We illustrate the performance of our classification method using gene expression data and show that it is superior to other classifiers especially to diagonal linear discriminant analysis and k-nearest neighbor which achieved the lowest error rate in the previous studies of tumor classification.

Key words and phrases: support vector machines, Vapnik-Chervonenkis dimension, microarray experiment, tumor classification.

1. Introduction

Support vector machines (svm) introduced by Vapnik (1989) are very popular classification tool. Usually their construction is derived using geometrical arguments - in case of linearly separable classes we want to find a hyperplane that separates classes and can be put into the maximally wide margin without points from the sample. In case of classes that are not linearly separable, we want to find a hyperplane that realizes the compromise between the width of the margin and the sum of the distances of missclassified observations from this margin.

However svm can be also derived using statistical decision theory. We are interested in the error rate of the classifier on the test sample. In this setting, the performance of the classifier depends on the error rate on the train sample and on the Vapnik-Chervonenkis (VC) dimension of the classifier. VC dimension of the classifier is defined as the maximal number of points it can shatter.

In order to generalize well and achieve low missclassification rate on the test sample, the classifier should have a low missclassification rate on the train sample and have low VC dimension. The following inequality with probability $1-\epsilon$ holds (Vapnik 1989)

$$e(d) \leq e_R(d) + c(h) \tag{1.1}$$

where

$$c(h) = \left(\frac{h(\ln \frac{2n}{h} + 1) - \log \frac{\epsilon}{4}}{h} \right)^{1/2} \tag{1.2}$$

Quantities $e(d)$ and $e_R(d)$ are the error rate for the test and train sample of the classifier d respectively, n is the number of observations in the train sample, h is the VC dimension of the classifier d. The function $c(h)$ is increasing.

Vapnik(1989) showed that hyperplanes with at least $2/A$ wide margins have the following VC dimension

$$h \leq \min(R^2A^2, q) + 1 \tag{1.3}$$

where q is the dimension of the feature space and R is the radius of the ball containing all the vectors of features from the train sample.

In order to minimize the error rate $e(d)$ when the classes are separable, svm is constructed in the following way. First we minimize $e_R(d)$. Classes are separable, so $e_R(d) = 0$. Next we minimize $c(h)$ using inequality (1.3). We minimize VC dimension by maximizing the width of margin.

In this article we adopt a different strategy and reverse this order. First we minimize (in the certain sense) the VC dimension of the classifier and then we minimize $e_R(d)$ - the error rate on the train sample.

This artice is organized as follows. In the next section we derive our class of support vector classifiers and describe its statistical properties. Then we discuss numerical issues concerning its computer implementation. In the fourth section we illustrate its performance using gene expression data. The last section concludes the article.

2. The classifier

We introduce the following class of classifiers

$$d(\mathbf{x}) = \text{sign}(y_i \mathbf{x}_i^T \mathbf{x} - y_j \mathbf{x}_j^T \mathbf{x} - b) \tag{2.1}$$

where $b = y_i \mathbf{x}_i^T \mathbf{x}_k - y_j \mathbf{x}_j^T \mathbf{x}_k$, $y_i y_j = -1$. Labels of the class y are $+1, -1$. Our support vector machines are expressed in terms of the dual form of the classical svm. In order to find the optimal decision rule d it is necessary to find i,j and k.

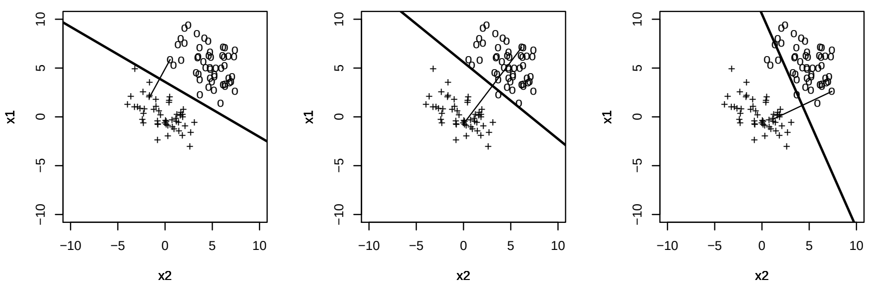

Figure 2.1: Randomly chosen three pairs of points and orthogonal hyperplane they may generate.

The geometrical interpretation is as follows: we consider only separating hyperplanes that are orthogonal to lines connecting pairs of points from the different classes in the train sample. Of course for a given pair of points there is a infinite number of hyperplanes that are orthogonal to the line connecting this pair. So we consider only hyperplanes that include at least one point from the train sample. As far as the parametric representation $\mathbf{a}^T\mathbf{x} = b$ of the hyperplane is concerned, its normal vector is given by $\mathbf{a} = y_i\mathbf{x}_i - y_j\mathbf{x}_j$ and the intercept by $b = \mathbf{a}^T\mathbf{x}_k$. The described idea is shown on Figure 1.

Vapnik (1989), in order to keep the VC dimension of the separating hyperplane at low level, introduced the restriction on the width of the margin. Wider margins imply lower VC dimension. We introduce the restriction on the orthogonality of the hyperplanes to the lines connecting pairs of points. Now we show that this class of hyperplanes has the VC dimension 9 or lower.

Theorem 1. *The VC dimension of the class of hyperplanes described by (2.1) is equal at most 9.*

Proof of Theorem 1

The VC dimension of the class of functions is equal to the maximal number of points it can shatter. A class of functions shatters a set of points if it can divide this set into two subsets in all possible ways. The set of h points can be divided into two subsets in 2^h ways. So the necessary condition for the class of functions to have the VC dimension h is that it can generate 2^h partitions of h points. So let us calculate the maximal number of partitions the class described by (2.1) can generate. We have $h(h-1)$ ordered pairs of points. Each pair may generate h partitions, because of the restriction that the hyperplane must include at least one point. So there are $h^2(h-1)$ partitions, not necessary unique. But if $h > 9$ then $2^h > h^2(h-1)$. So if there are more than 9 points, our method cannot shatter them, because it cannot generate sufficient number of partitions. So the VC dimension is equal 9 or less. *Q.E.D.*

Note that this bound on the VC dimension is very restrictive. Generally, the unrestricted family of hyperplanes can shatter $q+1$ points where q is the dimension of the feature vector, so the VC dimension of the hyperplanes in R^q is equal $q+1$. The bound (1.3) introduced by Vapnik for his class of hyperplanes may be reached and equal to the VC dimesion of the unrestricted hyperplanes. Our bound is equal 9 and it does not grow when the dimension of the feature space grows. This is very important in applications where the vector of features is very high for example in case of gene expression data where there are thousands of explanatory variables.

After introducing the family of hyperplanes with relatively low VC dimension we may go to the second step of building the optimal hyperplane: minimizing the error rate for the train dataset. We do it by exhaustive search: we consider all possible pairs of points from the class $y=-1$ and $y=+1$ (there are $n_1 n_{-1}$ pairs) and for a given pair of points we consider all possible hyperplanes that are orthogonal to this pair and include at least one point (there are n points). So there are $n_1 n_{-1} n$ hyperplanes to consider, where n_1 is the number of observations from class $y=1$ and n_{-1} is the number of observations from class $y=-1$. Finally we choose a hyperplane that achieved the lowest error rate. This is our optimal hyperplane.

So the problem of finding the optimal hyperplane is reduced to finding i, j and k in (2.1). Our classifier needs no extra tuning and has no additional parameters to set. This is the contrast to the svm with soft margin where the C constant must be set. This constant is usually set using crossvalidation which extends the computation time.

3. Implementation

The algorithm for finding the optimal hyperplane can be outlined in the following steps

```
1. for $i = 1, \ldots, n_1$ do

2.     for $j = 1, \ldots, n_{-1}$ do
```

```
3.        $\mathbf{a} = \mathbf{x}_{1i} - \mathbf{x}_{-1j}$
4.        $b_k = \mathbf{a}^T\mathbf{x}_k,\ k = 1, \ldots, n$
5.            consider classification rules "if $\mathbf{a}^T\mathbf{x} \geq b_k$ then $y = 1$ else $y = -1$", $k = 1, \ldots, n$
6.    **end**
7. **end**
8. choose rule that achieved lowest value of the error rate
```

We show how to modify this algorithm to diminish computational burden corresponding to the number of features.

Note that for $i = 1, ..., n_1,\ \ j = 1, ..., n_{-1},\ \ k = 1, ..., n$ we have

$$\mathbf{a}^T\mathbf{x}_k = (\mathbf{x}_{1i} - \mathbf{x}_{-1j})^T\mathbf{x}_k = \mathbf{x}_{1i}^T\mathbf{x}_k - \mathbf{x}_{-1i}^T\mathbf{x}_k, \tag{3.1}$$

Let us consider $\mathbf{X}_{n\times k}$ matrix which i-th row consists of the vector of features for i-th observation, $\mathbf{X}_{i\cdot} = \mathbf{x}_i^T$. Then

$$\mathbf{A} = \mathbf{X}\mathbf{X}^T = [a_{ij}] \tag{3.2}$$

$$a_{ij} = \mathbf{x}_i^T\mathbf{x}_j \tag{3.3}$$

so

$$\mathbf{a}^T\mathbf{x}_k = \mathbf{A}[1i, k] - \mathbf{A}[-1j, k] \tag{3.4}$$

So in order to calculate scores $\mathbf{a}^T\mathbf{x}$ it is sufficient to calculate the matrix $\mathbf{A}$ at the beginning of the computation and use its elements in the inner loop.

Then note that classifiers in the inner loop differ only in the intercept so in order to find proper cut-off value b that minimizes the fraction of incorrectly classified observations it is sufficient to sort observations by $\mathbf{a}^T\mathbf{x}$ and calculate number of observations from class $y = 1$ and $y = -1$ in subsets $\{(\mathbf{x}, y) : \mathbf{a}^T\mathbf{x} \geq b_k\}$ and $\{(\mathbf{x}, y) : \mathbf{a}^T\mathbf{x} < b_k\}$, where $b_k = \mathbf{a}^T\mathbf{x}_k$, $k = 1, \ldots, n$.

The overall average complexity of the algorithm is equal $o(n^3 \ln n) + o(n^2k)$. We have $o(n^2)$ pairs of observations and we need time $o(n \ln n)$ to sort observations in order to choose the cut-off value. In order to calculate the matrix $\mathbf{A}$ we need time $o(n^2k)$ using the naive algorithm for matrix multiplication. Note that the matrix $\mathbf{A}$ is

Table 4.1: Data sets statistics

data set	No. of vars	No. of instances	Size of Classes
Colon Tumor	2000	62	40/22
ALL-AML Leukemia	7129	72	47/25
Central Nervous System	7129	60	21/39
Lung Cancer	12533	181	31/150
Prostate Cancer	12600	136	77/59
Ovarian Cancer	15154	253	91/162
Breast Cancer	24481	97	46/51

calculated only once at the beginning of the algorithm and this time is negligibly small even for large data sets. So the overall complexity practically does not depend on the number of features k, but strongly depends on the number of observations. For example the algorithm needs about 0.5 second for data set with 100 observations and about 1 minute for data set with 1000 observations (for "spam" data set from UC Irvine Machine Learning Repository and randomly chosen 100 and 1000 observations respectively. The code was written in C and we used Pentium 2.59GHz machine). Calculating $\mathbf{X}\mathbf{X}^T$ for 97×24482 matrix takes about 1 second (this are dimensions of the largest data set in our study).

4. Empirical example

We used data sets from Kent Ridge Bio-medical Data Set Repository (http://datam.i2r.a-star.edu.sg/datasets/krbd/).
We used only data sets with the binary response variable and without or with small number of missing values. We used 7 data sets. A short description of these sets is given in Table 4.1. We used the following benchmarking classifiers: k nearest neighbor (knn) with $k = 1$ and diagonal linear discriminant analysis (dlda). These are methods that achieved lowest error rate in the study of Dudoit, Fridlyand, Speed (2002).

Each data set was split randomly it in proportion 7:3 into the train and the test subset. Then the each classifier was learned on the train part and its correctness rate was calculated on the test part. We repeated this procedure 300 times and calculated the mean of the correctness rate using these 300 repetitions. These values are reported in tables. We used R software, especially *knn* function from *class* package and *dlda* from *supclust*. We implemented our classifier in C and in R but since the computational time was satisfactionary for R, we used this implementation in all experiments. We also tried to use different kernels for our algorithm but there was no significant improvement

Table 4.2: Correctness rate

	all variables			10 best variables		
	pair	knn	dlda	pair	knn	dlda
Prostate Cancer	**80.39**	78.32	54.93	60.41	**60.54**	55.24
Ovarian Cancer	92.30	**92.36**	71.39	**75.01**	70.92	65.24
Lung Cancer	97.56	93.49	**98.37**	**85.18**	83.58	80.86
Colon Tumor	**73.47**	73.12	62.53	**64.37**	63.47	57.05
Central Nervous System	**60.65**	57.93	56.69	**55.70**	54.59	50.28
Breast Cancer	**60.14**	58.79	52.68	55.47	52.80	**56.76**
ALL-AML Leukemia	87.15	85.88	**87.26**	67.89	66.21	**69.92**

Table 4.3: Correctness rate

	50 best variables			100 best variables		
	pair	knn	dlda	pair	knn	dlda
Prostate Cancer	63.63	**65.14**	55.07	66.42	**67.07**	54.76
Ovarian Cancer	**80.82**	78.73	67.84	**82.60**	81.54	69.19
Lung Cancer	89.58	89.91	**93.48**	92.03	91.70	**95.93**
Colon Tumor	**69.74**	68.75	60.32	**69.96**	69.81	59.12
Central Nervous System	**57.74**	55.54	52.57	**57.94**	55.02	53.17
Breast Cancer	**58.31**	56.23	57.47	**59.66**	57.82	55.54
ALL-AML Leukemia	72.32	75.50	**80.18**	74.86	78.00	**83.08**

over the linear kernel. We have chosen $k = 1$ for knn and used Euclidean metrics. All variables are standardized - we subtracted the mean and divided by the standard deviation.

We also repeated the experiment in several versions: with all the variables, with 10, 50 and 100 best variables according to the ratio of the between-group to the withingroup sums of squares, similarly to Dudoit, Fridlyand, Speed (2002). The correctness rate for all experiments are shown in Table 2 and 3. In all tables column "pair" refers to our method. Best results are marked bold.

We may observe that our algorithm achieved the highest correctness rate for four data sets out of seven in all the variants - when considering all variables or when using only a subset of features. We may also note that reducing the number of features diminished the correctness rate.

5. Conclusions

In this artice we presented a new class of support vector machines using only two support vectors. Our approach has a very low VC dimension and is a good classification tool for the gene expression data. It has no additional parameters and needs no tuning. It can also be easily implemented. Our implementation is very simple, but it can be easily parallelized which may be crucial for large datasets.

As far as the future work is concerned, it would be interesting to compare our approach with other classifiers using different than the error rate criterion. In some applications, for example in credit scoring or direct marketing, the performance of the classifier is measured by the area under the ROC curve or by lift charts. In the second step of our algorithm, when we choose the proper hyperplane we may replace the error rate criterion by any other and achieve different classifier that is well suited for different applications.

References

Dudoit S., Fridlyand J., Speed T. P. (2002). Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. *Journal of the American Statistical Association.* **97**, 77-87.

Vapnik V. N. (1989). *Statistical Learning Theory.* Wiley-Interscience, New York

Institute of Econometrics, Warsaw School of Economics
Al. Niepodleglosci 164, 02-554 Warsaw, Poland
E-mail: mo23628@sgh.waw.pl