Adv. & Appl. in Stat. **6**(2) (2006), 207-216

SUPPORT VECTOR MACHINE WITH KERNEL METHODS AND SIMULATIONS

TAE-SOO KIM and JUNG-HO AHN

School of Liberal Arts
Seoul National University of Technology
Seoul, 139-743, South Korea
e-mail: tskim@snut.ac.kr

Computer Science Department
Yonsei University
Seoul, South Korea

Abstract

Support Vector Machines (SVMs) are powerful tools for data classification. SVMs attempt to separate two given sets in N-dimensional real (Euclidean) space $\Re^N$ by a nonlinear surface, often only implicitly defined by a kernel function. We examined the priority of given various kernel functions for given data sets which follow particular probability distributions.

1. Introduction

SVM is an implementation of Vapnik's Support Vector Machine [8] for the problem of pattern recognition, for the problem of regression, and for the problem of learning a ranking function. The optimization algorithms used in SVM are described in [3] and [4]. The algorithm has scalable memory requirements and can handle problems with many thousands of

2000 Mathematics Subject Classification: 62-04, 60-08, 62p30.

Keywords and phrases: support vector machine, kernel method, Lagrangian, risk function.

Received December 26, 2005

support vectors efficiently. Many researches have been devoted to the study of various learning algorithms which allow the extraction of these underlying regularities. If the learning has been successful, these intrinsic regularities will be captured in the values of some parameters of a learning machine; for a polynomial classifier, these parameters will be the weights and biases, and for a Gaussian classifier they will be weights and centers. In this paper, we study the Support Vector Algorithm with different kernel functions. We show that the algorithm allows us to construct different classifiers: polynomial classifiers, Gaussian classifiers.

2. Support Vector Classifier with Kernels

Let us start with a general notion of the learning problems that we consider in this paper. The task of classification is to find a rule, which, based on external observations, assigns an object to one of several classes. In the simplest case there are only two different classes. For the case of two-class pattern recognition, the task of learning from example can be formulated in the following way: suppose we are given empirical data

$$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \in X \times \{\pm 1\}. \tag{1}$$

Here, the *domain* X is some nonempty set that the patterns x_i are taken from; the y_i are called *labels* or *targets*. Unless stated otherwise, indices i and j will always be understood to run over the training set, i.e., $i, j = 1, \ldots, m.$ Note that we have not made any assumptions on the domain X other than it being a set. In order to study the problem of learning, we need additional structure and want to be able to generalize unseen data points. In the case of pattern recognition, this means that given some new pattern $x \in X,$ we want to predict the corresponding $y \in \{\pm 1\}.$ By this we mean, loosely speaking, that we choose y such that (x, y) is in some sense similar to the training examples. To this end, we need similarity measures in X. So, we require a similarity measure

$$k : X \times X \to \Re$$

$$(x, x') \mapsto k(x, x'),$$

i.e., a function that, given two examples x and x', returns a real number characterizing their similarity. For reasons that will become clear later, the function k is called a *kernel* [2]. In this domain, we have not made the assumption that the patterns live in a dot product space. In order to be able to use a dot product as a similarity measure, we use a map

$$\Phi : X \to F \subset \Re^N,$$

$$x \mapsto \vec{x}.$$

The space F is called a *feature space*. For the case of two-class pattern recognition, the task of learning from examples can be formulated in the following way: given a set of functions

$$\{f_\alpha : \alpha \in \Lambda\},\ f_\alpha : \Re^N \to \{-1, +1\}$$

and a set of examples

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_m, y_m),\quad \vec{x}_i \in \Re^N,\ y_i \in \{-1, +1\},$$

each one generated from an unknown probability distribution $P(\vec{x}, y)$, we want to find a function f_{a^*} which provides the smallest possible value for the risk

$$R(\alpha) = \int | f_\alpha(\vec{x}) - y \,| dP(\vec{x}, y).$$

The problem is that $R(\alpha)$ is unknown, since $P(\vec{x}, y)$ is unknown. Therefore an induction principle for risk minimization is necessary. The straightforward approach to minimize the empirical risk

$$R_{emp}(\alpha) = \frac{1}{m}\sum_{i=1}^{m} | f_\alpha(\vec{x}_i) - y_i \,|.$$

Turns out to guarantee a small actual risk (i.e., a small error on the training set does not imply a small error on a test set), if the number m of training examples is limited. To make the most out of a limited amount of data, novel statistical techniques have been developed during the last 35 years. The structural risk minimization principle is such a technique. It is based on the fact that for the above learning problem, for any $\alpha \in \Lambda$ with a probability of at least $1 - \eta$, the bound $R(\alpha) \leq R_{emp}(\alpha) + \phi\left(\frac{h}{m}, \frac{\log(\eta)}{m}\right)$ holds, ϕ being defined as

$$\phi\left(\frac{h}{m}, \frac{\log(\eta)}{m}\right) = \sqrt{\frac{h\left(\log\frac{2m}{h} + 1\right) - \log(\eta/4)}{m}}.$$

The parameter h is called the *VC-dimension* of a set of functions. It describes the capacity of a set of functions implemented by the learning machine. For binary classification, h is the maximal number of points k which can be separated into two classes in all possible 2^k ways by using functions of the learning machine, i.e., for each possible separation there exists a function which takes the value 1 on one class and -1 on the other class.

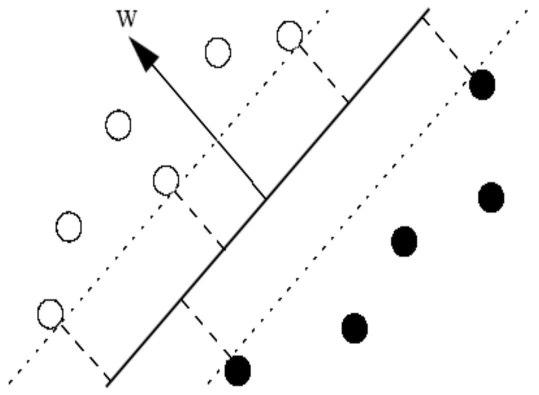

Figure 1. Linear classifier and margins: a linear classifier is defined by a hyperplane's normal vector h and an offset h, i.e., the decision boundary is h (thick line). Each of the two halfspaces defined by this hyperplane corresponds to one class, i.e., h. The margin of a linear classifier is the minimal distance of any training point to the hyperplane. In this case it is the distance between the dotted lines and the thick line.

Let us for moment assume that the training sample is separable by a hyperplane (see Fig. 1), i.e., we choose functions of the form $f(\vec{x}) = (\vec{w} \cdot \vec{x}) + b.$ It was shown that for the class of hyperplanes the VC-dimension itself can be bounded in terms of another quantity, the *margin* (also Fig. 1). The margin is defined as the minimal distance of a sample to the decision surface. The margin in turn can be measured by the length of the weight vector $\vec{w}$: as we assumed that the training sample is separable we can rescale $\vec{w}$ and b such that the points closest to the hyperplane satisfy $|(\vec{w} \cdot \vec{x}_i) + b| = 1$ (i.e., obtain the so-called canonical representation of the hyperplane). Now consider two samples $\vec{x}_1$ and $\vec{x}_2$ from different classes with $(\vec{w} \cdot \vec{x}_1) + b = 1$ and $(\vec{w} \cdot \vec{x}_2) + b = -1,$ respectively. Then the margin is given by the distance of these two points, measured perpendicular to the hyperplane, i.e., $\left(\frac{\vec{w}}{\| \vec{w} \|} \cdot (\vec{x}_1 - \vec{x}_2)\right) = \frac{2}{\| \vec{w} \|}.$ The result linking the VC-dimension of the class of separating hyperplanes to the margin or the length of the weight vector $\vec{w},$ respectively is given by the following inequalities: $h \le \Lambda^2 R^2 + 1$ and $\| \vec{w} \|_2 \le \Lambda,$ where R is the radius of the smallest ball around the data. Thus, if we bound the margin of a function class from below, say by $\frac{2}{\Lambda},$ we can control its VC-dimension. The choice of linear functions seems to be very limiting (i.e., instead of being likely to overfit we are now more likely to underfit). Fortunately there is a way to have both, linear models and a very rich set of nonlinear decision functions. The so-called curse of dimensionality from statistics says essentially that the difficulty of an estimation problem increases drastically with the dimension N of the space, since-in principle-as a function of N one needs exponentially many patterns to sample the space properly. This well-known statement induces some doubts about whether it is a good idea to go a high-dimensional feature space for learning. Fortunately, for certain feature spaces and corresponding mappings there is a highly effective trick for computing scalar products in feature spaces using *kernel functions* [1], [2], [6] and [8].

Now suppose we are given a set of examples $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_m, y_m)$, $\vec{x}_i \in \Re^N$, $y_i \in \{-1, +1\}$, and we want to find a decision function of examples $f_{\vec{w},b}$ with the property $f_{\vec{w},b}(\vec{x}_i) = y_i$, $i = 1, \ldots, m$.

It is based on two facts. First, among all hyperplanes separating the data, there exists a unique one yielding the maximum margin of separation between the classes,

$$\max_{\vec{w},b} \min\{\| \vec{x} - \vec{x}_i \| : \vec{x} \in R^N, (\vec{w} \cdot \vec{x}) + b = 9, i = 1, \ldots, m\}.$$

Second, the capacity decreases with increasing margin. To construct this optimal hyperplane, one solves the following optimization problem:

$$\text{Minimize } \tau(\vec{w}) = \frac{1}{2} \| \vec{w} \|^2$$

$$\text{Subject to } y_i((\vec{w} \cdot \vec{x}_i) + b) \geq 1, \quad i = 1, \ldots, m.$$

This constrained optimization problem is dealt with by introducing Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \| \vec{w} \|^2 - \sum_{i=1}^{m} \alpha_i (y_i((\vec{x}_i \cdot \vec{w}) + b) - 1).$$

The Lagrangian L has to be minimized with respect to the primal variables $\vec{w}$ and b has to be maximized with respect to the dual variables α_i. The solution vector $\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i$ with $\sum_{i=1}^{m} \alpha_i y_i = 0$ thus has an expansion in terms of a subset of the training patterns, namely those patterns whose α_i is non-zero, called *Support Vectors* [3]. By substituting the solution into L with kernel function, one eliminates the primal variables and arrives at the Wolfe dual of the optimization problem: find multipliers α_i which

$$\text{Maximize } W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$\text{Subject to } \alpha_i \geq 0,\ i = 1, \ldots, m \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0.$$

Then the hyperplane decision function can be written as

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i \cdot (\Phi(x) \cdot \Phi(x_i)) + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b\right).$$

3. Experimental Results

Case 1. Random variables are normally distributed.

Let X_P and X_N be the random variables for positive and negative data, respectively and normally distributed by

$$X_P \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right) \text{ and } X_N \sim N\left(\begin{pmatrix} C_N \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right).$$

In real-world application, we usually normalize each feature of data to have unit variance so our assumption is not too artificial. Table 1 shows the simulation results according to the distance C_N between the means of positive and negative training data and correlation ρ of data features. We also compared the performances of the homogeneous polynomial and Gaussian kernels. In this example SVM with polynomial of degree 3 empirically outperforms that of other degree.

The more correlated the features and the farther the distance between means, the better SVM classifies. SVM with Gaussian kernel usually outperforms that of polynomial kernel but their performances are not significantly different. SVM is known as one of best classifiers but the distance between means can be not long enough its classification performance will be poor. Hence, it would be better for researchers to select good features to make sure the mean of clusters' centers are far enough in training phase. The statistical T test shows that the variability in ρ and C_N are both significant with level 0.01.

Table 1. The SVM performance according to the distance C_N between positive and negative means and the correlation ρ of data features. P and G indicate the SVM with the polynomial kernel of degree 3 and that with Gaussian kernel of parameter 1, respectively

C_N \ ρ		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.5	P	0.5537	0.5420	0.5470	0.5447	0.5570	0.5543	0.5583	0.5780	0.5887	0.6270
	G	0.5397	0.5250	0.5627	0.5627	0.5607	0.5787	0.5700	0.5753	0.5990	0.6643
1	P	0.6467	0.6443	0.6313	0.6440	0.6620	0.6617	0.6723	0.7030	0.7327	0.7937
	G	0.6430	0.6637	0.6583	0.6663	0.6677	0.6780	0.6900	0.7210	0.7517	0.8350
1.5	P	0.7330	0.7227	0.7237	0.7403	0.7457	0.7737	0.7867	0.8163	0.8633	0.9127
	G	0.7477	0.7453	0.7393	0.7513	0.7540	0.7870	0.7953	0.8250	0.8733	0.9287
2	P	0.8110	0.8100	0.8023	0.8297	0.8330	0.8487	0.8757	0.8920	0.9247	0.9740
	G	0.8183	0.8310	0.8243	0.8280	0.8390	0.8530	0.8727	0.8923	0.9297	0.9763
2.5	P	0.8573	0.8743	0.8740	0.8757	0.8913	0.8937	0.9187	0.9377	0.9647	0.9813
	G	0.8690	0.8757	0.8827	0.8747	0.9010	0.9070	0.9220	0.9367	0.9717	0.9890
3	P	0.9150	0.9097	0.9207	0.9190	0.9227	0.9340	0.9543	0.9620	0.9757	0.9907
	G	0.9187	0.9170	0.9220	0.9290	0.9297	0.9447	0.9607	0.9700	0.9830	0.9957

Case 2. Random variables are distributed by Gaussian Mixture Model (GMM).

In face recognition, it has been observed that the variability in an image due to pose and illumination is often greater than that due to a change in the person's identity [5].

Therefore, researchers on face recognition usually assume that it is more likely that face images can be clustered in some Euclidean space according to the variations of pose and illumination. This experiment reflects such situation.

Let X_P and X_N be positive and negative random variables and be distributed by

$$X_P \sim p_1 N(m_1^P, C_1^P) + p_2 N(m_2^P, C_2^P)$$

and

$$X_N \sim p_1 N(m_1^N, C_1^N) + p_2 N(m_2^N, C_2^N),$$

where $p_1 = p_2 = 0.5$. m_1^P and m_1^N are generated in a ball with center $(0, 0)$ and radius R. m_2^P and m_2^N are generated in a ball with center $(C, 0)$ and radius R. The covariance matrices are $C_1^P = C_2^P = C_1^N = C_2^N = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$, where $\rho = 10$. The larger C, the longer the distance between two clusters. As R is smaller, the clusters are smaller, i.e., some GMM components from different classes are closer. Table 2 shows the SVM performance according to the parameters of R and C. The statistical T test shows that SVM performance is affected by the radius of cluster, R, significantly with level 0.01, but not significantly by the variability in the distance C between two clusters.

Table 2. The SVM performance according to the radius of cluster, R, and distance C between two clusters. P and G indicate the SVM with the polynomial of degree 3 and that with Gaussian kernels of parameter 1, respectively

C \ R		1	1.5	2	2.5	3
1	P	0.5767	0.6447	0.6760	0.7303	0.7787
	G	0.6023	0.6800	0.7027	0.7753	0.8173
1.5	P	0.5887	0.6413	0.6897	0.7253	0.7573
	G	0.6110	0.6630	0.7283	0.7590	0.8043
2	P	0.5837	0.6273	0.7067	0.7393	0.7887
	G	0.6127	0.6613	0.7437	0.7870	0.8207
2.5	P	0.5937	0.6677	0.6877	0.7260	0.7817
	G	0.6173	0.6957	0.7547	0.7590	0.8230
3	P	0.5877	0.6713	0.7010	0.7247	0.7817
	G	0.6100	0.7013	0.7367	0.7830	0.8283

References

[1] M. Aizerman, E. Braverman and L. Rozonoer, Theoretical foundations of the potential function method in pattern recognition learning, Autom. Remote Control 25 (1964), 821-837.

[2] B. E. Boser, I. M. Guyon and V. N. Vapnik, A training algorithm for optimal margin classifiers, Proceedings of the 5th Annul ACM Workshop on Computational Learning Theory, D. Haussler, ed., 1992, pp. 144-152.

[3] T. Joachims, Making large-scale SVM learning practical, Advances in Kernel Methods - Support Vector Learning, B. Schölkopf, C. Burges and A. Smola, eds., MIT Press, 1999.

[4] Thorsten Joachims, Learning to classify text using support vector machines, Dissertation, Kluwer, 2002.

[5] Y. Moses, Y. Adini and S. Ullman, Face recognition: The problem of compensating for changes in illumination direction, European Conference on Computer Vision, 1994, pp. 286-296.

[6] S. Saitoh, Theory of Reproducing Kernels and its Applications, Harlow, Longman, U. K., 1988.

[7] Bernhard Scholkopf, Statistical Learning and Kernel Methods, Microsoft Research Limited, February, 2000,

http://research.microsoft.com/~bsc

[8] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.

■