DIMACS Series in Discrete Mathematics
and Theoretical Computer Science

Support Vector Machines for Classification

Dmitriy Fradkin and Ilya Muchnik

1. Introduction

In many real-life situations we want to be able to assign an object to one of several categories based on some of its characteristics. For example, based on the results of several medical tests we want to be able to say whether a patient has a particular disease or should be recommended a specific treatment.

In computer science such situations are described as classification problems. A binary (two-class) classification problem can be described as follows: given a set of labeled points (x_i, y_i), $i = \overline{1, l}$, where $x_i \in \Re^d$ are vectors of features and $y_i \in \{-1, +1\}$ are class labels, construct a rule that correctly assigns a new point x to one of the classes.

The vectors x_i in this formulation correspond to objects, and the dimensions of the space are the features or characteristics of these objects. For example, a vector may represent a person, with individual features corresponding to the measurements given by some medical tests — blood pressure, cholesterol level, white cell count and so on.

Using labels $\{0, \ldots, K-1\}$ instead of $\{-1, +1\}$ we can describe a multiclass problem with K classes. A classification method or algorithm is a particular way of constructing a rule, also called a classifier, from the labeled data and applying it to the new data.

Support Vector Machines (SVM) recently became one of the most popular classification methods. They have been used in a wide variety of applications such as text classification [**8**], facial expression recognition [**9**], gene analysis[**6**] and many others.

Support Vector Machines can be thought of as a method for constructing a special kind of rule, called a linear classifier, in a way that produces classifiers with theoretical guarantees of good predictive performance (the quality of classification on unseen data). The theoretical foundation of this method is given by statistical learning theory [**15**].

2000 *Mathematics Subject Classification.* 62H30.
Key words and phrases. support vector machine, VC dimension, kernel method.
Dr. Muchnik's work was supported through NSF grant CCR 0325398.

While no general method for producing non-linear rules with such properties is known, the so-called "kernel trick" can be used to construct special kinds of non-linear rules using the SVM methodology.

2. Linear Classifiers

A classifier can frequently be represented as a function $f(x) : \Re^d \rightarrow \Re$. In a two-class case, a point is assigned to the positive class if $f(x) \geq 0$, and to the negative class otherwise.

A classifier function $f(x)$ is linear if it can be expressed as:

$$f(x; w, b) = < w, x > +b \tag{2.1}$$

where w, b are parameters of the function and $<, >$ denotes the inner product of two vectors.

A set of points (x_i, y_i), $i = \overline{1, l}$, where $y_i \in \{-1, +1\}$ are class labels, is called *linearly separable* if a linear classifier can be found so that $y_i f(x_i) > 0$, $\forall i = 1, \ldots, l$.

The use of linear classifiers in machine learning can be traced back to Rosenblatt's work on the perceptron [**14**], though they have been used before that in statistics. The perceptron algorithm works by taking one instance at a time and predicting its class. If the prediction is correct, no adjustments are made. If the prediction is wrong, the parameters, describing a hyperplane, are moved in the direction of the point where the mistake occurred. A scalar value, η, called the *learning rate* determines how the how far the parameters are moved. The choice of a learning rate can significantly affect the number of iterations until convergence on a linearly-separable set. The pseudocode for the perceptron learning, taken from [**4**], is given as Algorithm 1.

Algorithm 1 Online Perceptron Learning Algorithm [**4**]

```
Require: A linearly separable set $S$, learning rate $\eta \in \Re^+$
 1: $w_0 = 0; b_0 = 0; k = 0;$
 2: $R = \max_{1 \leq i \leq l} ||x_i||$
 3: while at least one mistake is made in the for loop do
 4:   for $i = 1, \ldots, l$ do
 5:     if $y_i(< w_k, x_i > +b_k) \leq 0$ then
 6:       $w_{k+1} = w_k + \eta y_i x_i$
 7:       $b_{k+1} = b_k + \eta y_i R^2$ (updating bias[1])
 8:       $k = k + 1$
 9:     end if
10:   end for
11: end while
12: Return $w_k, b_k$, where $k$ is the number of mistakes
```

[1]Usually this update is given as $b_{k+1} = b_k + \eta y_i$. However, such formulation makes the result of Novikov's theorem, given below, dependent on the exact value of the bias, b^*, of the separating hyperplane. This reflects the fact that R can be increased simply by moving all points away from the origin. The update used in the Algorithm 1 (suggested in [**4**]) compensates for that.

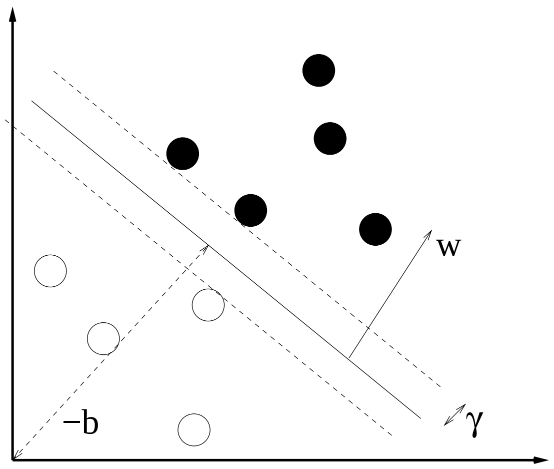

FIGURE 1. Here $(w, -b)$ define the separating hyperplane and γ is the size of the margin. The relation between w and γ is discussed in the text.

2.1. Margin and VC dimension. The idea of *margin* (Figure 1) has come to play an important role in the theory of statistical learning.

A hyperplane

$$< w^*, x > +b^* = 0, \; ||w^*|| = 1 \tag{2.2}$$

is called γ-margin separating hyperplane if

$$y_i(< w^*, x > +b^*) \geq \gamma \tag{2.3}$$

for all (x_i, y_i) in set S. Here γ (clearly $\gamma > 0$) is the margin.

Any separating hyperplane can be converted into this form. Suppose

$$y(< w, x > +b) \geq 1. \tag{2.4}$$

Then, by setting $w^* = \frac{w}{||w||}$ and $b^* = \frac{b}{||w||}$, we obtain a γ-margin separating hyperplane with $\gamma = \frac{1}{||w||}$.

The first result suggesting a relation between the margin and predictive ability of a classifier was Novikov's theorem.

THEOREM 2.1. *Novikov's Theorem* [**11**] *Let S, $|S| = l$ be a training set, i.e. a set of points with class labels, and let*

$$R = \max_{1 \leq i \leq l} ||x_i|| \tag{2.5}$$

Suppose that there exists a γ-margin separating hyperplane (w, b) such that $y_i(< w, x_i > +b) \geq \gamma$, $\forall\ 1 \leq i \leq l$. Then the number of mistakes made by the on-line perceptron algorithm on S is at most

$$\left(\frac{2R}{\gamma}\right)^2 \tag{2.6}$$

This theorem effectively proves that for a linearly separable set of points the perceptron algorithm finds a separating hyperplane after making a finite number of mistakes. The number of mistakes is directly proportional to the ratio of the volume of the data to the measure of separation of the classes, γ.

Note that Novikov's theorem shows convergence on a periodic training sequence. This result has been extended to an arbitrary infinite sequence of points (each belonging to one of two region that can be linearly separated) by Aizerman et. al. [**1**].

Novikov's theorem bounds the number of errors in the training stage. But in classification we are interested in the accuracy of a classifier on unseen data. Such a number clearly cannot be computed exactly, but it turns out it can be bounded.

Before proceeding let us define the notion of Vapnik-Chevronenkis dimension.

DEFINITION 2.2. The *Vapnik-Chervonenkis (VC) dimension* of a set of classifiers is the maximum number h of points that can be separated into all possible 2^h ways using classifiers in this set. If for any n there exists a set of n points that can be separated into two classes in all possible ways, the VC dimension of this set of functions is said to be infinite.

Intuitively, VC dimension measures the complexity of the classifiers in the set. If the classifiers are simple, they have small VC dimension. If they are complicated, the VC dimension is large. For example, the VC dimension of hyperplanes in R^d is known to be $d + 1$. The following two results bound the VC dimension of the set of γ-margin separating hyperplanes and the probability of misclassifying an unseen instance with such a hyperplane chosen on the training data.

THEOREM 2.3. **Theorem 5.1 in** [**15**] *Let $x \in X$ belong to sphere of radius R. The the set of γ-margin separating hyperplanes has VC dimension h bounded by:*

$$h \leq min\left(\left(\frac{R}{\gamma}\right)^2, d\right) + 1 \tag{2.7}$$

THEOREM 2.4. **Corollary to Theorem 5.1 in** [**15**] *With probability $1 - \eta$ the probability of a test example not being separated correctly by a γ-margin hyperplane has the bound*

$$P_{error} \leq \frac{m}{l} + \frac{E}{2}\left(1 + \sqrt{1 + \frac{4m}{lE}}\right) \tag{2.8}$$

where

$$E = 4\frac{h(\ln\frac{2l}{h} + 1) - \ln\frac{\eta}{4}}{l}, \tag{2.9}$$

m is the number of training examples not separated correctly by γ-margin hyperplane, and h is the bound of the VC dimension given in theorem 2.4.

The bound on the probability of making a mistake on unseen data is proportional to the VC dimension of the set of classifiers. In other words, everything else being equal, a classifier with a lower VC dimension is likely to be a better predictor. Notice that this result does not depend on any assumptions about the distribution of the test data — it is a "distribution-free" bound.

Since the upper bound on VC dimension is inversely proportional to the margin, the strategy for building a good classifier is to have as large a margin as possible while keeping the number of errors on the training set low.

This is somewhat similar to the idea regularization but is motivated from the perspective of the statistical learning theory. It can also be seen as a version of Occam's razor: we want to use the simplest classifier that makes no (or fewest) mistakes on the training set.

2.2. The Maximal Margin in Separable Case. We have seen in the previous section that the probability of making a mistake is inversely proportional to the size of the margin. Thus we would like to find a classifier with the largest margin that still correctly separates the training points.

The maximal-margin separating hyperplane can be found by solving the following optimization problem:

$$\text{minimize}_{w,b} < w, w > \tag{2.10}$$

subject to:

$$y_i(< w, x_i > +b) \geq 1,\ \forall i = 1, \dots, l \tag{2.11}$$

One method for solving optimization problems involves introducing Lagrange multipliers,α_i, for the constraints [**2**]. In this case, the so-called Lagrangian function is given by:

$$L(w, b, \alpha) = \frac{1}{2} < w, w > - \sum_{i=1}^{l} \alpha_i [y_i(< w, x_i > +b) - 1] \tag{2.12}$$

Taking derivatives with respect to w and b gives:

$$w = \sum_{i=1}^{l} y_i \alpha_i x_i \tag{2.13}$$

and

$$0 = \sum_{i=1}^{l} y_i \alpha_i \tag{2.14}$$

Note that w is given by a linear combination of the training points. We will return to this observation later on.

Re-substituting these into the primal problem (2.12) gives a dual formulation:

$$\text{maximize } W(\alpha) = \sum_{i=1}^{l} \alpha_i - \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i, x_j > \tag{2.15}$$

subject to:

$$\sum_{i=1}^{l} y_i \alpha_i = 0,\ \alpha_i \geq 0,\ \forall i = 1, \dots, l \tag{2.16}$$

Let α^* be a vector of parameters optimizing $W(\alpha)$. The the weight vector $w^* = \sum_{i=1}^{l} y_i \alpha_i^* x_i$ is a maximal margin hyperplane, with margin

$$\gamma = \frac{1}{||w^*||_2}. \tag{2.17}$$

The parameter b is not present in the dual problem and has to be computed from the primal constraints (2.11).

Notice that, because of (2.13), the classifier $f(x)$ can be expressed as:

$$f(x) = < w, x > +b = \sum_{i=1}^{l} y_i \alpha_i < x_i, x > +b \tag{2.18}$$

If $\alpha_i = 0$, then x_i is not used in decision rule and can be discarded. Points x_i such that $\alpha_i \neq 0$ lie on the margin and are called support vectors. They determine the decision boundary.

2.3. Extension for the non-separable case. Until now we have discussed only the linearly-separable case. However, similar machinery can be used to handle the non-separable case. The main idea is that those points that lie on the wrong side of the hyperplane are explicitly penalized by introducing *slack* variables, ξ, that control how far on the wrong side of a hyperplane a point lies.

The optimization problem becomes:

$$\text{minimize}_{w,b} \frac{1}{2} < w, w > +C \sum_{i=1}^{l} \xi_i \tag{2.19}$$

subject to:

$$y_i(< w, x_i > +b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \forall i = 1, \ldots, l \tag{2.20}$$

where the parameter C, controlling the trade-off between the size of the margin and the training errors, is chosen by the user.

The dual then becomes:

$$\text{maximize} \sum_{i=1}^{l} \alpha_i - \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i, x_j > \tag{2.21}$$

subject to:

$$\sum_{i=1}^{l} y_i \alpha_i = 0,\ C \geq \alpha_i \geq 0,\ \forall i = 1, \ldots, l \tag{2.22}$$

Once again the solution is given by a linear combination of inner products with support vectors: $w = \sum_{i=1}^{l} y_i \alpha_i x_i$. There are no general methods for choosing parameters b in a non-separable case - it is usually set to optimize some performance measure on a training or a validation set. The same approach is often taken for choosing a value of C.

2.4. Multi-class classification. Many methods exist for building a multi-class classifier system from binary classifiers (one-vs-all, one-vs-one, error-correcting output codes (ECOC) [**5**], Directed Acyclic Graph (DAG) [**13**]). In all of these approaches multiple binary classifiers are trained separately and their predictions are then combined. For example, in one-vs-all classification, with K classes, K classifiers are constructed. Each recognizes points of one of the classes as positive and those of all others as negative. A new point is assigned to the class h if the corresponding classifier gives this point the highest score among all K classifiers. In one-vs-one, a classifier is trained for each pair of classes. Classification usually proceeds as follows: each of $\frac{K(K-1)}{2}$ classifiers makes a prediction and the number of votes for each class is counted. The point is assigned to the class that has received most votes. DAG and ECOC are more sophisticated in how classifiers are trained and how their predictions are combined.

Special multiclass methods have been developed for SVM. They usually involve solving a single optimization problem. In [**16**] the optimization problem to be solved (over K separating hyperplanes and lK slack variables) is:

$$\text{minimize}_{w^j, b^j, \xi^j} \frac{1}{2} \sum_{j=1}^{K} < w^j, w^j > + C \sum_{i=1}^{l} \sum_{j \neq y_i} \xi_i^j \tag{2.23}$$

subject to:

$$(< w^{y_i}, x_i > + b^{y_i}) \geq (< w^j, x_i > + b^j + 2 - \xi_i^j),\ \forall j \neq y_i \tag{2.24}$$
$$\xi_i^j \geq 0,\ \forall i = 1, \ldots, l$$

A new point x is classified as follows:

$$f(x) = \text{argmax}_{j=1,\ldots,K} (< w^j, x > + b^j) \tag{2.25}$$

Cramer and Singer [**3**] suggested a somewhat different formulation, requiring only l slack variables:

$$\text{minimize}_{w^j, \xi} \frac{1}{2} \sum_{j=1}^{K} < w^j, w^j > + C \sum_{i=1}^{l} \xi_i, \tag{2.26}$$

subject to:

$$(< w^{y_i}, x_i > - < w^j, x_i >) \geq 1 - \xi_i \tag{2.27}$$
$$\forall i = \overline{1, l},\ j = \overline{1, K} : \xi_i \geq 0,\ j \neq y_i.$$

The resulting decision function is:

$$f(x) = \text{argmax}_{j=1,\ldots,K} < w^j, x > \tag{2.28}$$

2.5. Computational Issues. We have shown that the problem of finding a maximal margin hyperplane can be formulated as a particular quadratic optimization problem. Many numerical methods for solving general quadratic optimization problems are known. For the special kind of optimization needed for SVM there exist particularly efficient algorithms such as Platt's sequential minimal optimization (SMO) [**12**] capable of handling thousands of vectors in thousand-dimensional spaces.

3. The "Kernel Trick"

The main idea behind the "kernel trick" is to map the data into a different space, called *feature space*, and to construct a linear classifier in this space. It can also be seen as a way to construct non-linear classifiers in the original space. Below we explain what the kernel trick is and how these two views are reconciled.

Notice that in the the dual problem (2.15) the training points are included only via their inner products. Also, as can be seen in (2.18), the classifier function $f(x)$ can be expressed as a sum of inner products with support vectors.

An important result, called Mercer's theorem, states that any symmetric positive semi-definite function $K(x, z)$ is an inner product in some space (and vice-versa). In other words, any such function $K(x, z)$ implicitly defines a mapping into so-called feature space $\phi : x \rightarrow \phi(x)$ such that $K(x, z) =< \phi(x), \phi(z) >$. Such functions K are called kernels.

The feature space can be high-dimensional or even have infinite dimension. However we don't need to know the actual mapping since we can use kernel function to compute similarity in the feature space.

Some examples of kernels include polynomial kernels

$$K(x,z) = (< x,z > +1)^p \tag{3.1}$$

and gaussian kernels

$$K(x,z) = e^{-\frac{||x-z||^2}{2\sigma^2}}. \tag{3.2}$$

Many kernels have been developed for special applications such as sequence matching in bioinformatics [**7**]. General properties of kernels are described in many publications, including [**4**].

By replacing the inner product in the formulation of SVM by a kernel and solving for Lagrange multipliers α_i, we can obtain via (2.18) a maximal margin separating hyperplane in the feature space defined by this kernel. Thus choosing non-linear kernels allows us to construct classifiers that are linear in the feature space, even though they are non-linear in the original space.

The dual problem in the kernel form is:

$$\text{maximize } W(\alpha) = \sum_{i=1}^{l} \alpha_i - \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{3.3}$$

$$\text{subject to } 0 = \sum_{i=1}^{l} y_i \alpha_i,\ \alpha_i \geq 0,\ \forall i = 1, \ldots, l \tag{3.4}$$

and the classifier is given by:

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i K(x_i, x). \tag{3.5}$$

The idea of viewing kernels as implicit maps into a feature space was first suggested in [**1**]. However this approach was not widely used until the emergence of SVM. Many algorithms other than SVM have now been "kernelized" — reformulated in terms of kernels rather than inner products. A survey of kernel-based methods is given by [**10**].

4. Conclusion

In this note we attempted to highlight the main ideas underlying the SVM method. For a detailed explanation of these, and many other ideas in statistical learning and kernel methods an interested reader is referred to [**4, 15**]. While [**4**] provides a clear and broad introduction to the area, [**15**] goes into much greater depth. Both of these books provide many references to relevant literature. Another valuable resource is kernel-machines website (**http://www.kernel-machines.org/**) which contains information on books, articles and software related to SVM and other kernel-based methods.

References

[1] M. Aizerman, E. M. Braverman, and L. I. Rozenoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25(6):821–837, June 1964.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to support vector machines and other kernelbased learning methods*. Cambridge University Press, 2000.

[5] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[7] T. Jaakkola, M. Diekhaus, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. pages 149–158, 1999.

[8] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*. Springer, 1998.

[9] P. Michel and R. E. Kaliouby. Real time facial expresion recognition in video using support vector machines. In *Proceedings of ICMI'03*, pages 258–264, 2003.

[10] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–202, March 2001.

[11] A. Novikov. On convergence proofs for perceptrons. In *Proceedings of the Symposium of the Mathematical Theory of Automata*, volume 12, pages 615–622, 1962.

[12] J. Platt. Fast training of support vector machines using sequential minimal optimization. In C. B. B. Schölkopf and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208. MIT Press, 1999.

[13] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, 12:547–553, 2000.

[14] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization of the brain. *Psychological Review*, 65:386–408, 1959.

[15] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2nd edition, 1998.

[16] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.

Department of Computer Science, Rutgers, The State University of New Jersey, Piscataway, New Jersey 08854, USA

E-mail address: `dfradkin@cs.rutgers.edu`

DIMACS, Rutgers, The State University of New Jersey, Piscataway, New Jersey 08854, USA

E-mail address: `muchnik@dimacs.rutgers.edu`