Multi-class Support Vector Machines

J. Weston, C. Watkins.

Technical Report

CSD-TR-98-04

May 20, 1998

Royal Holloway
University of London

Department of Computer Science
Egham, Surrey TW20 0EX, England

1 Introduction

The solution of binary classification problems using the Support Vector (SV) method is well developed. Multi-class pattern recognition problems (where one has $k > 2$ classes) are typically solved using voting scheme methods based on combining many binary classification decision functions [5, 2]. We propose two extensions to the SV method of pattern recognition to solve k-class problems in one (formal) step. We describe two different methods of SV k-class classification which do not use a combination of binary classification rules. The first method is a direct generalisation of the binary classification SV method (for the case $k = 2$ they give exactly the same support vectors and hyperplane), and the other results in solving a linear program rather than a quadratic one.

2 k-class Pattern Recognition

The k-class pattern recognition problem is to construct a decision function given ℓ iid (independent and identically distributed) samples (points) of an unknown function, typically with noise:

$$(x_1, y_1), \ldots, (x_\ell, y_\ell) \tag{1}$$

where $x_i, \ \ i = 1, \ldots, \ell$ is a vector of length d and $y_i \in \{1, \ldots, k\}$ representing the class of the sample.

The decision function $f(x, \alpha)$ which classifies a point x, is chosen from a set of functions defined by the parameter α. The task is to choose the parameter α such that for any x the function provides a classification y which is as close to the function to be estimated as possible. It is assumed the set of functions which the learning machine implements is chosen a *priori.*

In order to choose the best parameter α one measures the loss $L(y, f(x, \alpha))$ between the real class y and the decision $f(x, \alpha)$ at a given point x. We consider the loss function which measures the number of incorrectly classified training patterns:

$$L(y, f(x, \alpha)) = \begin{cases} 0 & \text{if y=}f(x, \alpha), \\ 1 & \text{otherwise} \end{cases}$$

3 Binary Classification Support Vector Machines

For the binary pattern recognition problem (case $k = 2$), the Support Vector approach has been well developed [4, 6, 7, 8]. The main idea is the following: construct a hyperplane to separate the two classes (labelled $y \in \{-1, 1\}$) so that the margin (the distance between the hyperplane and the nearest point) is maximal. This gives the following optimisation problem: minimise

$$\phi(w, \xi) = \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{\ell} \xi_i \tag{2}$$

with constraints

$$y_i((w \cdot x_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l$$

$$\xi_i \geq 0, \quad i = 1, \ldots, l.$$

The dual solution to this problem is: maximise the quadratic form (3) under constraints (4) and (5).

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{3}$$

with constraints

$$0 \leq \alpha_i \leq C, i = 1, \ldots, \ell \tag{4}$$

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \tag{5}$$

giving the decision function:

$$f(x) = \text{sign}\Big[(\sum_{i=1}^{\ell} \alpha_i y_i K(x, x_i)) + b\Big].$$

4 One-against-the-Rest and One-against-One Classifiers

One approach to solving k-class pattern recognition problems by considering the problem as a collection of binary classification problems. k classifiers can be constructed, one for each class. The k^{th} classifier constructs a hyperplane between class n and the $k-1$ other classes. A majority vote across the classifiers or some other measure can then be applied to classify a new point. Alternatively, $(\frac{k(k-1)}{2})$ hyperplanes can be constructed, separating each class from each other and similarly some voting scheme applied. The one-against-all method has been used widely in the Support Vector literature to solve multi-class pattern recognition problems, see for example [5] or [2].

5 k-class Support Vector Machines

A more natural way to solve k-class problems is to construct a decision function by considering all classes at once.

One can generalise the optimisation problem (2) to the following: minimise

$$\phi(w, \xi) = \frac{1}{2} \sum_{m=1}^{k} (w_m \cdot w_m) + C \sum_{i=1}^{\ell} \sum_{m \neq y_i} \xi_i^m \tag{6}$$

with constraints

$$(w_{y_i} \cdot x_i) + b_{y_i} \geq (w_m \cdot x_i) + b_m + 2 - \xi_i^m, \tag{7}$$

$$\xi_i^m \geq 0, \quad i = 1, \ldots, l \quad m \in \{1, \ldots, k\} \setminus y_i.$$

This gives the decision function:

$$f(x) = \arg\max_k \big[(w_i \cdot x) + b_i\big], \quad i = 1, \ldots, k.$$

We can find the solution to this optimisation problem in dual variables by finding the saddle point of the Lagrangian:

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2} \sum_{m=1}^{k} (w_m \cdot w_m) + C \sum_{i=1}^{\ell} \sum_{m=1}^{k} \xi_i^m \tag{8}$$

$$-\sum_{i=1}^{\ell}\sum_{m=1}^{k}\alpha_i^m\left[((w_{y_i}-w_m)\cdot x_i)+b_{y_i}-b_m-2+\xi_i^m\right]-\sum_{i=1}^{\ell}\sum_{m=1}^{k}\beta_i^m\xi_i^m$$

with the dummy variables

$$\alpha_i^{y_i}=0,\;\; \xi_i^{y_i}=2,\;\; \beta_i^{y_i}=0, \quad i=1,\dots,\ell$$

and constraints

$$\alpha_i^m\geq 0,\;\; \beta_i^m\geq 0,\;\; \xi_i^m\geq 0, \quad i=1,\dots,\ell \quad m\in\{1,\dots,k\}\setminus y_i$$

which has to be maximised with respect to α and β and minimised with respect to w and ξ. Introducing the notation

$$c_i^n=\begin{cases} 1 & \text{if } y_i=n \\ 0 & \text{if } y_i\neq n \end{cases}$$

and

$$A_i=\sum_{m=1}^{k}\alpha_i^m$$

and differentiating in w_n,b_n and ξ_j^n we obtain:

$$\frac{\partial L}{\partial w_n}=w_n+\sum_{i=1}^{\ell}\alpha_i^n x_i-\sum_{i=1}^{\ell}A_i c_i^n x_i$$

$$\frac{\partial L}{\partial b_n}=-\sum_{i=1}^{\ell}A_i c_i^n+\sum_{i=1}^{\ell}\alpha_i^n$$

$$\frac{\partial L}{\partial \xi_j^n}=-\alpha_j^n+C-\beta_j^n.$$

In the saddle point the solution should satisfy the conditions:

$$\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial w_n}=0 \;\Rightarrow\; w_n=\sum_{i=1}^{\ell}(c_i^n A_i-\alpha_i^n)x_i \tag{9}$$

$$\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial b_n}=0 \;\Rightarrow\; \sum_{i=1}^{\ell}\alpha_i^n=\sum_{i=1}^{\ell}c_i^n A_i \tag{10}$$

$$\frac{\partial L(w,b,\xi,\alpha,\beta)}{\partial \xi_n}=0 \;\Rightarrow\; \beta_j^n+\alpha_j^n=C \quad\text{and}\quad 0\leq\alpha_j^n\leq C. \tag{11}$$

Substituting (9) back into (8) we obtain:

$$W(b,\xi,\alpha,\beta)=\frac{1}{2}\sum_{m=1}^{k}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(c_i^m A_i-\alpha_i^m)(c_j^m A_j-\alpha_j^m)(x_i\cdot x_j)$$

$$-\sum_{m=1}^{k}\sum_{i=1}^{\ell}\alpha_i^m\left[\sum_{j=1}^{\ell}(c_j^{y_i}A_j-\alpha_j^{y_i})(x_i\cdot x_j)-\sum_{j=1}^{\ell}(c_j^m A_j-\alpha_j^m)(x_i\cdot x_j)+b_{y_i}-b_m-2\right]$$

$$-\sum_{m=1}^{k}\sum_{i=1}^{\ell}\alpha_i^m\xi_i^m+C\sum_{m=1}^{k}\sum_{i=1}^{\ell}\xi_i^m-\sum_{i=1}^{\ell}\sum_{m=1}^{k}\beta_i^m\xi_i^m.$$

Adding the constraint (11) the terms in ξ cancel. Considering the two terms in b only:

$$B_1=\sum_{i,m}\alpha_i^m b_{y_i}=\sum_m b_m(\sum_i c_i^m A_i)$$

and

$$B_2=-\sum_{i,m}\alpha_i^m b_m=-\sum_m b_m(\sum_i \alpha_i^m).$$

But, from (10) we have

$$\sum_i \alpha_i^m=\sum_i c_i^m A_i$$

so $B_1=B_2$ and the two terms cancel, giving

$$W(\alpha)=2\sum_{i,m}\alpha_i^m+\sum_{i,j,m}(\frac{1}{2}c_i^m c_j^m A_iA_j-\frac{1}{2}c_i^m A_i\alpha_j^m-\frac{1}{2}c_j^m A_j\alpha_i^m+\frac{1}{2}\alpha_i^m\alpha_j^m-c_j^{y_i}A_j\alpha_i^m$$

$$+\alpha_i^m\alpha_j^{y_i}+c_j^m A_j\alpha_i^m-\alpha_i^m\alpha_j^m)(x_i\cdot x_j).$$

Since $\sum_m c_i^m A_i\alpha_j^m=\sum_m c_j^m A_j\alpha_i^m$ we have

$$W(\alpha)=2\sum_{i,m}\alpha_i^m+\sum_{i,j,m}(\frac{1}{2}c_i^m c_j^m A_iA_j-c_j^{y_i}A_iA_j+\alpha_i^m\alpha_j^{y_i}-\frac{1}{2}\alpha_i^m\alpha_j^m)(x_i\cdot x_j)$$

but $\sum_m c_i^m c_j^m=c_i^{y_i}=c_j^{y_j}$ so

$$W(\alpha)=2\sum_{i,m}\alpha_i^m+\sum_{i,j,m}[-\frac{1}{2}c_j^{y_i}A_iA_j+\alpha_i^m\alpha_j^{y_i}-\frac{1}{2}\alpha_i^m\alpha_j^m](x_i\cdot x_j) \tag{12}$$

which is a quadratic function in terms of alpha with linear constraints

$$\sum_{i=1}^{\ell}\alpha_i^n=\sum_{i=1}^{\ell}c_i^n A_i,\quad n=1,\dots,k$$

and

$$0\le\alpha_i^m\le C,\quad \alpha_i^{y_i}=0$$

$$i=1,\dots,\ell\quad m\in\{1,\dots,k\}\setminus y_i.$$

This gives the decision function:

$$f(x,\alpha)=\arg\max_n\,[\sum_{i=1}^{\ell}(c_i^n A_i-\alpha_i^n)(x_i\cdot x)+b_n] \tag{13}$$

or equivalently

$$f(x,\alpha)=\arg\max_n\,[\sum_{i:y_i=n}A_i(x_i\cdot x)-\sum_{i:y_i\neq n}\alpha_i^n(x_i\cdot x)+b_n].$$

As usual the inner products $(x_i\cdot x_j)$ in (12) and (13) can be replaced with the convolution of the inner product $K(x_i,x_j)$.

Let us consider the solution to the optimisation problem (6) for different values of k.

1. Simplest case : k=1

   There are no constraints, and the decision is given by default to class 1.

2. Binary classification : k=2

   There is only one constraint for each training point (if we label $y \in \{0, 1\}$):

   $$(w_{y_i} \cdot x_i) + b_{y_i} \geq (w_{1-y_i} \cdot x_i) + b_{1-y_i} + 1 - \xi_i^{1-y_i}.$$

   The separation of the two classes by the maximum of two linear functions is equivalent to separating by a single hyperplane, re-arranging:

   $$((w_{y_i} - w_{1-y_i}) \cdot x_i) + b_{y_i} - b_{1-y_i} \geq 1 - \xi_i^{1-y_i}.$$

   It can be shown that this hyperplane is identical to the one obtained from the binary pattern recognition support vector machine (SVM) and will have the same support vector coefficients (the optimisation problems are equivalent).

6 k-Class Linear Programming Machines

One can also consider the generalisation of optimisation problem (3). The decision function

$$f(x) = \text{sign}((\sum_{i=1}^{\ell} \alpha_i y_i K(x, x_i)) + b).$$

used in Binary Classification SVM is equivalent to

$$f(x) = \arg\max_k (\sum_{i:y_i=k} \alpha_i K(x, x_i) + b_k)$$

when $k = 2$. In this formulation there are only ever ℓ coefficients, independent of the number of classes, k. Instead of considering the decision function in case $k = 2$ as a separating hyperplane (from which the problem was derived) it is equally valid to view each class having its own decision function, which is defined by only the training points belonging to the class. The decision rule is then the largest of the decision functions at any point x.

One can minimise the following linear program:

$$\sum_{i=1}^{\ell} \alpha_i + C \sum_{i=1}^{\ell} \sum_{j \neq y_i} \xi_i^j$$

subject to

$$\sum_{m:y_m=y_i} \alpha_m K(x_i, x_m) + b_{y_i} \geq \sum_{n:y_n=y_j} \alpha_n K(x_i, x_n) + b_{y_j} + 2 - \xi_i^j$$

$$\alpha_i \geq 0, \quad \xi_i^j \geq 0, \quad i = 1, \ldots, \ell, \quad j \in \{1, \ldots, k\} \setminus y_i$$

and use the decision rule

$$f(x) = \arg\max_n (\sum_{i:y_i=n} \alpha_i K(x, x_i) + b_n).$$

One can also consider the problem of finding a separating (hard margin) decision function only by removing the slack variables, ξ_i^k.

7 Further analysis

For binary SVMs the expectation of the probability of commiting an error on a test example is bounded by the ratio of the expectation of the number of training points that are support vectors to the number of examples in the training set [7]:

$$E[P(\text{error})] = \frac{E[\text{number of training points that are support vectors}]}{(\text{number of training vectors}) - 1} \tag{14}$$

This bound also holds in the multi-class case for the voting scheme methods (one-against-all and one-against-one) and for our multi-class support vector method (but not for linear programming machines). This can be seen by noting that any training point that is not a support vector is still classified correctly when it is left out of the training set. Note that this means we are interested in the size of the union of the set of all support vectors that define each hyperplane in the classifier, which is equivalent to the number of kernel calculations required, rather than the sum of the sizes of the sets.

Secondly, it is worth noting that the solution of the one-against-all method is a feasible solution of our multi-class sv method, but not necessarily the optimal one. This can be seen by considering the one-against-all method as one formal step. In the hard margin case ($C = \infty$) one is required to minimize

$$\sum_{m=1}^{k} (w_m \cdot w_m) \tag{15}$$

with constraints

$$w_{y_i} \cdot x_i + b_{y_i} \geq 1 \;\text{ and }\; w_j \cdot x_i + b_j \leq -1$$

$$i = 1, \ldots, \ell, \; j \in \{1, \ldots, k\} \setminus y_i.$$

One can see that if these constraints are satisfied so are constraints (7). This means that our multi-class method will have the same or lower value of (15) , where a small value of $(w \cdot w)$ in the binary case corresponds to low VC dimension and large margin (which mean good generalization).

8 Experiments

We present two types of experiments demonstrating the performance of the two new algorithms:

◇ experiments in the plane with artificial data which can be visualised, and

◇ experiments with benchmark data sets.

8.1 Examples in the Plane

To demonstrate the multi-class SV technique we first give two artifical examples (Fig 1).

Three classes of vectors (right) and four classes (left) are represented in the pictures as circles, boxes, pluses and crosses. The decision rule was constructed using an inner product of polynomial type with d=2. In the left hand side three classes are separated with maximal margin by the quadratic multi-class SVM (top) and the linear multi-class SVM (bottom). On the right hand side four classes cannot be separated without errors; the errors are indicated by large crosses and the support vectors by rings.

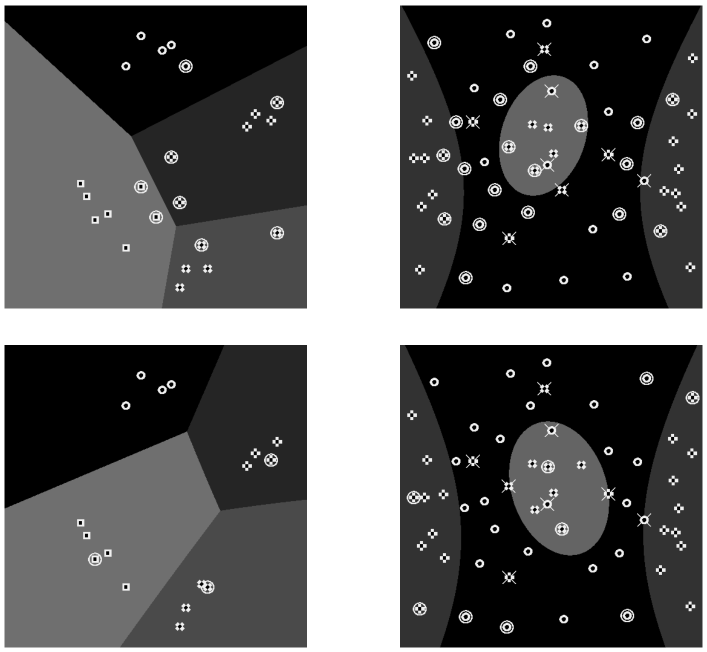

Figure 1 Three classes of vectors (right) and four classes (left) are represented in the pictures as circles, boxes, pluses and crosses. The decision rule was constructed using an inner product of polynomial type with d=2. In the left hand side three classes are separated with maximal margin by the quadratic multi-class SVM (top) and the linear multi-class SVM (bottom). On the right hand side four classes cannot be separated without errors; the errors are indicated by large crosses and the support vectors by rings.

	#	#	#	1-a-a		1-a-1		qp-mc-sv		lp-mc-sv	
name	pts	att	class	%err	svs	%Err	svs	%err	svs	%err	svs
iris	150	3	4	1.33	75	1.33	54	1.33	31	2.0	13
wine	178	13	3	5.6	398	5.6	268	3.6	135	10.8	110
glass	214	9	7	35.2	308	36.4	368	35.6	113	37.2	72
soy	289	208	17	2.43	406	2.43	1669	2.43	316	5.41	102
vowel	528	10	11	39.8	2170	38.7	3069	34.8	1249	39.6	258

Table 1 Performance of the two new algorithms, quadratic multi-class SVM (qp-mv-sv) and linear multi-class SVM (lp-mc-sv), is compared to one-against-all (1-a-a) and one-against-one (1-a-1) binary pattern recognition SVM. The raw error (% err) and number of non-zero coefficients (svs) of each algorithm is given for a variety of data sets.

8.2 Benchmark Data Set Experiments

We tested our two methods on a collection of five benchmark problems from the UCI machine learning repository[1]. Where no test set was provided the data were each split randomly ten times with a tenth of the data being used as a test set. The performance of the two multi-class support vector methods (quadratic and

[1]URL:http://www.ics.uci.edu/ mlearn/MLRepository.html

linear) were compared with one-against-all and one-against-one binary classification SV methods. To enable comparison, for each algorithm $C = \infty$ was chosen (the training data must be classified without error) , and the radial basis kernel $K(x, y) = \exp(-\sigma|x-y|^2)$ was used, with the same value of the parameter σ for each algorithm. The results are summarised in Table 1 [2].

The data sets chosen were small because currently we have not yet implemented a decomposition algorithm to train with larger data sets. The performance on these small data sets, however, is good. qp-mc-sv performance is comparable on the data sets to the 1-a-a method and was better on the "glass" data set, however, it always had fewer non-zero coefficients [3]. lp-mc-sv also achieved reasonable results, although worse than other methods, but with a significantly reduced number of support vectors compared to all other methods. A smaller number of support vectors means faster classification speed, which has been a problem with the SV method compared to other techniques, see for example [3]. 1-a-1 performed worse than qp-mc-sv and usually had a very large number of support vectors.

9 Limitations of the Methods

The optimisation problem we are required to solve for qp-mc-sv is quadratic in $((k-1)*l)$ variables, which can be large. Although decomposition techniques can be used, as in the usual SV case, it is still likely be slower to train than 1-a-a, although this is not yet clear.

lp-mc-sv is also a complex optimisation problem, with $(k*l)$ (or ℓ for a hard margin) variables and $(k*l)$ constraints, even though it is only linear.

10 Conclusions and Further Research

In conclusion, we have described two new methods of solving multi-class pattern recognition problems with support vector machines. Results obtained on the benchmark datasets suggest that although the new methods do not outperform the one-against-all and one-against-one methods (but maybe a slight improvement in the case of qp-mc-sv and slighlty worse in the case of lp-mc-sv) they do reduce the number of support vectors needed to describe the decision function.

Further research involves testing the methods on larger datasets by using decomposition techniques.

11 Acknowledgements

Thanks to all those who contributed to the data sets used in this paper. Thanks also to M. Stitson for writing the code for one-against-one and one-against-all SV classification. We also thank Kai Vogtlaender for useful comments.

In communication with V. Vapnik and V. Blanz we discovered they independently derived the quadratic multi-class support vector method described in this report.

[2] The "soy" data set is a modified version of the soy-bean data set found at the UCI repository, due to missing data. This data set was in fact obtained from the repository at "http://pages.prodigy.com/upso/datasets.htm".

[3] Actually, we are interested in the number of coefficients and the number of kernel calculations in all these methods (as some of the kernel values can be re-used). However, results are not reported here.

References

[1] K. Bennett and O.L. Mangasaria. Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3:27 – 39, 1993.

[2] V. Blanz, B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, pages 251 – 256, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.

[3] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.

[4] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

[5] B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings, First International Conference on Knowledge Discovery & Data Mining.* AAAI Press, Menlo Park, CA, 1995.

[6] V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian].* Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

[7] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[8] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998. forthcoming.