Support Vector Machines for Multi-Class Pattern Recognition

J. Weston and C. Watkins

Department of Computer Science
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
{jasonw,chrisw}@dcs.rhbnc.ac.uk

Abstract. The solution of binary classification problems using support vector machines (SVMs) is well developed, but multi-class problems with more than two classes have typically been solved by combining independently produced binary classifiers. We propose a formulation of the SVM that enables a multi-class pattern recognition problem to be solved in a single optimisation. We also propose a similar generalization of linear programming machines. We report experiments using bench-mark datasets in which these two methods achieve a reduction in the number of support vectors and kernel calculations needed.

1. 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 $y_i \in \{1, \ldots, k\}$ represents the class of the sample. A natural loss function is the number of mistakes made.

2. Solving k-Class Problems with Binary SVMs

For the binary pattern recognition problem (case $k = 2$), the support vector approach has been well developed [3, 5].

The classical approach to solving k-class pattern recognition problems is to consider the problem as a collection of binary classification problems. In the one-versus-rest method one constructs k classifiers, one for each class. The n^{th} classifier constructs a hyperplane between class n and the $k-1$ other classes. A particular point is assigned to the class for which the distance from the margin, in the positive direction (i.e. in the direction in which class "one" lies rather than class "rest"), is maximal. This method has been used widely in

the support vector literature to solve multi-class pattern recognition problems, see for example [4] or [2].

Alternatively, ($\frac{k(k-1)}{2}$) hyperplanes can be constructed (the one-versus-one method), separating each class from each other class and a decision function constructed using some ad-hoc voting system.

3. k-Class Support Vector Machines

A more natural way to solve k-class problems is to construct a piecewise linear separation of the k classes in a single optimisation.

The binary SVM optimisation problem [5] is generalised to the following: minimise

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

subject to

$$(w_{y_i}\cdot x_i) + b_{y_i} \geq (w_m \cdot x_i) + b_m + 2 - \xi_i^m, \tag{3}$$

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

This gives the decision function:

$$f(x) = \arg\max_n \left[(w_n \cdot x) + b_n\right], \quad n = 1,\ldots,k. \tag{4}$$

Note first that for $k = 2$, this formulation of the optimisation problem reduces exactly to the binary SVM solution [5] if we take $w_1 = -w_2$, $b_1 = -b_2$, and $\xi_i = (1/2)\xi_i^1$ for pattern i in class 1 and $\xi_i = (1/2)\xi_i^2$ for pattern i in class 2.

Second, a decision function of the form (4) can be more powerful than a set of one-versus-rest binary classifiers, in the sense that it is possible to construct multi-class datasets that can be separated perfectly by a decision rule of type (4), but in which the training data cannot be classified without error by one-versus-rest. For example, consider classes which lie inside segments of a d dimensional sphere (see [1] for details).

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

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

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

with the dummy variables

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

and constraints

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

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

$$A_i = \sum_{m=1}^{k} \alpha_i^m \qquad \text{and} \qquad c_i^n = \begin{cases} 1 & \text{if } y_i = n \\ 0 & \text{if } y_i \neq n \end{cases}$$

and through various manipulations (omitted due to lack of space) we arrive at: maximise

$$W^*(\alpha) = 2 \sum_{i,m} \alpha_i^m + \sum_{i,j,m} \big[-\frac{1}{2} c_j^{y_i} A_i A_j + \alpha_i^m \alpha_j^{y_i} - \frac{1}{2} \alpha_i^m \alpha_j^m \big] (x_i \cdot x_j) \qquad (5)$$

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

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

and

$$0 \leq \alpha_i^m \leq C, \quad \alpha_i^{y_i} = 0, \qquad i = 1, \ldots, \ell \quad m \in \{1, \ldots, k\} \setminus y_i. \qquad (6)$$

One can find the values $b_n, \;\; n = 1, \ldots, k$ by solving a set of simultaneous equations from the Kuhn-Tucker optimality conditions or by obtaining them as the values of the dual variables when using an interior point optimizer (as noted by A. Smola and B. Schölkopf). One then obtains the decision function:

$$f(x) = \arg\max_n \big[\sum_{i=1}^{\ell} (c_i^n A_i - \alpha_i^n)(x_i \cdot x) + b_n \big]. \qquad (7)$$

As usual the inner products $(x_i \cdot x_j)$ in (5) and (7) can be replaced with a generalised inner product $K(x_i, x_j)$ (see [5]).

4. k-Class Linear Programming Machines

One can also consider the generalisation of the linear programming machine method [6]. 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

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

In this formulation there are only ℓ coefficients, independent of the number of classes, k, whereas in the other methods there are ℓk coefficients. Furthermore, the regularization directly attempts to reduce the number of non-zero coefficients.

5. Further analysis

For binary SVMs the expectation of the probability of commiting an error on a test example is bounded by the ratio of of the expectation of the number of training points that are support vectors to the number of examples in the training set [5]:

$$E[P(\text{error})] = \frac{E[\text{number of training points that are support vectors}]}{(\text{number of training vectors}) - 1} \qquad (8)$$

This bound also holds in the multi-class case for the voting scheme methods (one-against-rest and one-against-one) and for our multi-class support vector method. 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-rest method is a feasible solution of our multi-class method, but not necessarily the optimal one. This can be seen by considering the one-against-rest method as one formal step. In the hard margin case ($C = \infty$) one is required to minimize $\sum_{m=1}^{k} (w_m \cdot w_m)$ with constraints $w_{y_i} \cdot x_i + b_{y_i} \geq 1$ and $w_j \cdot x_i + b_j \leq -1$ for $i = 1, \ldots, \ell$ and $j \in \{1, \ldots, k\} \setminus y_i$. One can see that if these constraints are satisfied so are constraints (3). This means that our multi-class method will have the same or lower value of $\sum_{m=1}^{k} (w_m \cdot w_m)$, where a small value of $(w \cdot w)$ in the binary case corresponds to low VC dimension and large margin (which mean good generalization).

Table 1: Comparison of Error Rates

name	**#pts**	**#atts**	**#class**	**1-v-r**	**1-v-1**	**mc-sv**	**mc-lp**
iris	150	3	4	1.33	1.33	1.33	2.0
wine	178	13	4	5.6	3.6	3.6	10.8
glass	214	9	7	35.2	36.4	35.6	37.2
soy	289	208	17	2.43	2.43	2.43	5.41
vowel	528	10	11	39.8	38.7	34.8	39.6
postal	500	256	10	9.36	8.37	9.67	13.9

Table 2: Comparison of the Number of Support Vectors

	1-v-r		**1-v-1**		**mc-sv**		**mc-lp**	
name	**svs**	**k-calcs**	**svs**	**k-calcs**	**svs**	**k-calcs**	**svs**	**k-calcs**
iris	75	40	54	40	46	31	13	13
wine	398	136	268	135	135	110	110	110
glass	308	131	368	127	203	113	72	72
soy	406	197	1669	229	316	160	102	102
vowel	2170	439	3069	774	1249	348	238	238
postal	651	294	1226	285	499	249	114	114

6. Simulations

We tested our method 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 multi-class support vector method (**mc-sv**) and multi-class linear programming method (**mc-lp**) were compared with one-versus-rest (**1-v-r**) and one-versus-one (**1-v-1**) binary classification SV methods. To enable comparison decision functions were constructed with a hard margin (parameter $C = \infty$) and the same radial basis function kernel for each algorithm. The results are summarised in Tables 1 and 2.

mc-sv performed comparably with the voting scheme methods, but had a smaller number of non-zero coefficients (**svs**) and kernel calculations (**k-calcs**)[2]. Note this means lower values of the upper bound on generalization error (8). **mc-lp** had a significantly reduced number of **svs** and **k-calcs** (they are equivalent because each inner product has only one associated multiplier)

[1]URL:http://www.ics.uci.edu/mlearn/MLRepository.html. The training set "postal" is the first 500 examples of the U.S Postal service database (LeCun et *al.*, 1989), using the whole testing set.

[2]The number of kernel calculations will usually be less than the number of support vectors in all the algorithms as they can be cached if two support vectors use the same kernel calculation.

but did not achieve good generalization ability. We have since realized this could be due to the difference in the decision rules between the methods: **mc-lp** has a different number of basis functions in its decision rule (ℓ instead of ℓk) and may perform well with very different choices of kernel to the other three methods, but we kept the choice fixed.

7. Conclusions

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 the new methods can reduce the number of support vectors and kernel computations. One can construct examples where the new methods will separate data but voting scheme methods cannot. However, this has not been reflected in error rates on the datasets used.

Acknowledgements

In communication with V. Vapnik and V. Blanz we discovered they independently derived the multi-class support vector method described in this article. We thank ESPRC for providing financial support through grant GR/L35812 ("Support Vector and Bayesian Learning Algorithms"). We also thank Mark O. Stitson for co-writing the support vector implementation used in our experiments.

References

[1] K. Bennett and O.L. Mangasarian. 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 *Artificial Neural Networks — ICANN'96*, pages 251 – 256, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.

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

[4] 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.

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

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