Multiclass Learning at One-class Complexity

Sandor Szedmak
ISIS Group
Electronics and Computer Scinece
University of Southampton
Southampton, U.K.
ss03v@ecs.soton.ac.uk

John Shawe-Taylor
ISIS Group
Electronics and Computer Scinece
University of Southampton
Southampton, U.K.
jst@ecs.soton.ac.uk

Abstract

We show in this paper the multiclass classification problem can be implemented in the maximum margin framework with the complexity of one binary Support Vector Machine. We show reducing the complexity does not involve diminishing performance but in some cases this approach can improve the classification accuracy. The multiclass classification is realized in the framework where the output labels are vector valued.

1 Introduction

Multiclass classification has been considered a more complex problem than the well-known implementations of binary classification. There are two main streams among the attempts to attack these kind of problems. In the first one decomposes the multiclass problem into a certain combination of binary problems, e.g. "one versus all", "one versus one" approaches built upon some kind of binary classification. The second derives a regression based solution framework [12], possibly also exploiting the multivariate capability of partial least squares analysis [11], [1] and [10].

Recently Evegniou et al. [4] and Micchelli et al. [7], [8] presented a synthesis of the kernel learning approach with a general form of regression. In this framework vector labelled output items are also learned by a machine that is an extension of the Support Vector learner. In [4] the complexity issue is mentioned as a weakness of the presented approach. We show there is an implementation of this kind of machine with computational complexity independent of the number of classes and that it requires no more computation than a single binary Support Vector Classifier. The multiclass learning is then expressed as an application of this technique.

First, we formulate the Support Vector Machine with vector output (voSVM) that cuts down the complexity and then the realization of the multiclass learning will be given with experimental results. Following this we present a vector perceptron learner realizing a potential online multiclass learning, derived from the vector valued SVM. We present a Novikoff type theorem for this algorithm as well as experimental results on the same data as that used to test the multiclass SVM.

The notations that we use are summarized in Table 1. Note that we assume every Hilbert space mentioned has finite dimension and is defined above the real numbers.

$\mathcal{H}_n$ is an arbitrary Hilbert space with dimension n,

$\mathcal{H}_x$ is a Hilbert space comprising the possible input vectors

$\mathcal{H}_{\phi(x)}$ is a Hilbert space comprising the feature vectors

$\mathcal{H}_y$ is a Hilbert space comprising the label vectors

$\mathbf{W}$ is a matrix representing a linear operator mapping from the feature space $\mathcal{H}_{\phi(x)}$ into the label space $\mathcal{H}_y$,

$\langle ., . \rangle$, $\|.\|$ denotes the inner product and the norm defined in the corresponding Hilbert space,

$\mathbf{tr}(\mathbf{W})$ denotes the trace of the matrix $\mathbf{W}$,

$\mathbf{dim}(\mathcal{H})$ gives the dimension of the space $\mathcal{H}$.

Table 1: Notation used in the paper

2 Formulation of the SVM with vector output

The idea of our implementation for the vector valued Support Vector Machine comes from a simple reinterpretation of the normal vector of the separating hyperplane. We say this vector is a projection operator of the feature vectors into a one-dimensional subspace. An extension of the range of this projection into multi-dimensional subspaces gives the solution for vector labelled learning.

Assume we have a sample S of pairs $\{(\mathbf{y}_i, \mathbf{x}_i) : \ \mathbf{y}_i \in \mathcal{H}_y, \ \mathbf{x}_i \in \mathcal{H}_x, \ i = 1, \ldots, m\}$ independently and identically generated by an unknown multivariate distribution P. The Support Vector Machine with vector output is realized on this sample by the following optimization problem

$$
\begin{aligned}
\min \quad & \frac{1}{2}\mathbf{tr}(\mathbf{W}^T\mathbf{W}) + C\mathbf{1}^T\boldsymbol{\xi} \qquad (1)\\
\text{subject to} \quad & \{\mathbf{W}|\mathbf{W} : \mathcal{H}_{\phi(x)} \to \mathcal{H}_y, \mathbf{W} \text{ linear operator}\},\\
& \{\mathbf{b}|\mathbf{b} \in \mathcal{H}_y\}, \text{bias vector}\\
& \{\boldsymbol{\xi}|\boldsymbol{\xi} \in \mathcal{H}_n\} \text{slack or error vector}\\
& \langle \mathbf{y}_i, (\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_i) + \mathbf{b})\rangle \geq q_i - p_i\xi_i, \ i = 1, \ldots, m,\\
& \boldsymbol{\xi} \geq \mathbf{0}.
\end{aligned}
$$

where $\mathbf{0}$ and $\mathbf{1}$ denote the vectors with components 0 and 1 respectively. The real values q_i and p_i denote normalization constraints that can be chosen from the set of values $\{1, \|\mathbf{y}_i\|, \|\boldsymbol{\phi}(\mathbf{x}_i)\|, \|\mathbf{y}_i\|\|\boldsymbol{\phi}(\mathbf{x}_i)\|\}$ depending on the particular task. This kind of normalization allows us to tune the training error.

A particular example will illustrate the point. Let every q_i be $\|\boldsymbol{\phi}(\mathbf{x}_i)\|$ and every p_i be 1, then the magnitude of the error measured by the slack variables ξ_i will be the same independently of the norm of the feature vectors, therefore the effect caused by outliers is controlled. Obviously the appropriate selection of this type of normalization depends on the problem being solved.

The norm of the feature vectors are of course given by the square root of diagonal entries of the corresponding kernel matrix.

Introducing dual variables $\{\alpha_i | i = 1, \ldots m\}$ to the margin constraints and based on the Karush-Kuhn-Tucker theory we can express the linear operator $\mathbf{W}$ by using the tensor

products of the output and the feature vectors, that is

$$\mathbf{W} = \sum_{i=1}^{m} \alpha_i \mathbf{y}_i \boldsymbol{\phi}(\mathbf{x}_i)^T. \tag{2}$$

The dual gives

$$\begin{aligned} \min \quad & \sum_{i,j=1}^{m} \alpha_i \alpha_j \overbrace{\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle}^{\kappa_{ij}^{\phi}} \overbrace{\langle \mathbf{y}_i, \mathbf{y}_j) \rangle}^{\kappa_{ij}^{y}} - \sum_{i=1}^{m} q_i \alpha_i, \\ \text{subject to} \quad & \{\alpha_i | \alpha_i \in \mathbb{R}\}, \\ & \sum_{i=1}^{m} (\mathbf{y}_i)_t \alpha_i = 0, \ t = 1, \ldots, \dim(\mathcal{H}_y), \\ & \frac{C}{p_i} \geq \alpha_i \geq 0, \ i = 1, \ldots, m, \end{aligned} \tag{3}$$

where we can write the output of inner products in the objective as kernel items $\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle \langle \mathbf{y}_i, \mathbf{y}_j) \rangle = \kappa_{ij}^{\phi} \kappa_{ij}^{y}$, where κ_{ij}^{ϕ} and κ_{ij}^{y} stand for the elements of the kernel matrices for the feature vectors and for the label vectors respectively. Hence, the vector labels are kernelized as well. The synthesised kernel is the element-wise product of the input and the output kernels, an operation that preserves positive semi-definiteness.

The complexity of the dual moderately increases relative to the base SVM because the structure of the objective remains the same and we have constraints with the same content but the number of them is increased to the dimension of the output space. However, using a special optimization technique known as the Augmented Lagrangian approach, see [2], this additional complexity can be eliminated. For most practical cases the bias $\mathbf{b}$ can be ignored to give a simpler formulation.

3 Multiclass classification

The multiclass classification can be implemented within the framework of the vector valued SVM. Let us assume the label vectors are chosen out of a finite set $\{\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_T\}$ in the learning task. The decision function predicting one of these labels can be expressed by using the predicted vector output

$$\begin{aligned} d(x) &= \arg \max_{t=1,\ldots,T} \langle \hat{\mathbf{y}}_t, \mathbf{W}\boldsymbol{\phi}(\mathbf{x}) + \mathbf{b} \rangle \\ &= \arg \max_{t=1,\ldots,T} \sum_{i=1}^{m} \alpha_i \kappa^y(\hat{\mathbf{y}}_t, \mathbf{y}_i) \kappa^{\phi}(\mathbf{x}_i, \mathbf{x}) + \langle \hat{\mathbf{y}}_t, \mathbf{b} \rangle, \end{aligned} \tag{4}$$

where the bias vector $\mathbf{b}$ is the corresponding Lagrangian of the constraint $\sum_{i=1}^{m} (\mathbf{y}_i)_t \alpha_i = 0, \ t = 1, \ldots, \dim(\mathcal{H}_y)$ in the dual.

Now we are able to set up a multiclass classification. Let the label vectors be chosen as indicator vectors of the classes following the rule

$$(\mathbf{y}_i)_t = \begin{cases} 1 & \text{if item } i \text{ belongs to category } t, t = 1, \ldots, T, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Obviously there are some other choices of the vector labels. One, which minimizes the correlation between all pairs of the labels, gives the vertices of hyper-tetrahedron with

values

$$(\mathbf{y}_i)_t = \begin{cases} \sqrt{\frac{T-1}{T}} & \text{if item } i \text{ belongs to category } t, t = 1, \ldots, T, \\ -\frac{1}{\sqrt{T(T-1)}} & \text{otherwise,} \end{cases} \tag{6}$$

where the length of these vectors are normalized to 1.

If the label vectors are chosen as the indicator vectors given in (5), then we have T special maximum margin machines realizing a set of one class SVMs for each of the classes. This statement follows from the fact multiplying the projection matrix $\mathbf{W}$ from the left with the transpose of a label vector with only one non-zero component selects one row of $\mathbf{W}$ and this row can be considered as a normal vector of the hyperplane cutting the feature space into two parts such that one part contains the corresponding class with the smallest error.

If the indicator type label vectors from (5) are applied then the bias has to be excluded from the model, since the dual constraint contains only non-negative components of $\{\mathbf{y}_i\}$, thus, the only feasible solution for α is $\mathbf{0}$. In this case the separating hyperplanes are linear subspaces of the feature space.

Surprisingly this simple learner can work and gives as good result as the "one versus all" and "one versus one" approaches can but with much less computational effort. This result shows that the problem of multiclass learning still has not been fully understood.

4 Perceptron algorithm for multiclass learning

The formulation of the SVM for vector output also suggests an implementation of a perceptron type algorithm for multiclass classification. Consider the optimization problem where for the sake of simplicity we drop the normalization constants occurring in (1)

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} h(\lambda - \langle \mathbf{y}_i, \mathbf{W}\phi(x_i)\rangle) \\ \text{subject to} \quad & \{\mathbf{W} | \mathbf{W} : \mathcal{H}_x \to \mathcal{H}_y, \mathbf{W} \text{ a linear operator}\}, \end{aligned} \tag{7}$$

where λ is a prescribed margin assumed to be equal to 1 in the sequel, and the function $h(u)$ denotes the Hinge loss, that is

$$h(u) = \begin{cases} u & \text{if } u > 0, \\ 0 & \text{otherwise.} \end{cases} . \tag{8}$$

The error function that we are going to minimize has subgradient with respect to $\mathbf{W}$ and this can be computed independently in an incremental way for each term occurring in the summation (7). The reader can consult [2] and [6] for details of incremental subgradient methods. The term-wise subgradient is equal to

$$\partial h(\lambda - \langle \mathbf{y}_i, \mathbf{W}\phi(x_i)\rangle)|_{\mathbf{W}} = \begin{cases} -\mathbf{y}_i \phi(x_i)^T & \text{if } \lambda - \langle \mathbf{y}_i, \mathbf{W}\phi(x_i)\rangle > 0 \\ 0 & \text{otherwise.} \end{cases} . \tag{9}$$

We can define the learning speed with a step size, denoted by s, and we have the perceptron-like algorithm given in Figure 1.

The departure from the original perceptron algorithm is very slight. Here we need to learn a matrix realizing the projection of the input vectors into the output space. The incremental subgradient based update employs the tensor product of the corresponding output and input vectors to update the projection matrix. After choosing indicator vectors or vectors expressing interactions between the classes the multiclass learning can be trained against a sequence of input and output vectors.

An analogue of the standard Novikoff theorem is able to provide a bound on the number of updates.

Output of the learner: $\mathbf{W} \in \mathbb{R}^{dim(\mathcal{H}_y) \times dim(\mathcal{H}_x)}$
Initialization: $\mathbf{W} = \mathbf{0}$; $i = 1$;
Repeat for $i = 1, 2, \cdots$:
  read input: $x_i \in \mathbb{R}^n$;
  if $\langle \mathbf{y}_i, \mathbf{W}\phi(x_i) \rangle < \lambda$ **then**
    $\mathbf{W} = \mathbf{W} + s\mathbf{y}_i\phi(x_i)^T$

(10)

Figure 1: Vector perceptron algorithm

Theorem 1. *Suppose we have a training set S of m vector input/output pairs*

$$S = \{(\mathbf{y}_1, \phi(x_1)), \ldots, (\mathbf{y}_m, \phi(x_m))\}$$

with $\|\phi(x_i)\| = 1 = \|\mathbf{y}_i\|$ for all i and further that there exists a weight matrix $\mathbf{W}^\star$ such that

$$\mathbf{y}_i \mathbf{W}^\star \phi(x_i) \geq 1$$

then Algorithm 1 with $s = 1$ will halt after t steps where

$$t \leq 3\|\mathbf{W}^\star\|_F^2.$$

Proof. Following the Novikoff pattern we first upper bound the norm of the matrix $\mathbf{W}_t$ obtained after t updates:

$$\begin{aligned} \|\mathbf{W}_t\|_F^2 &= \|\mathbf{W}_{t-1}\|_F^2 + 2\mathbf{y}_i\mathbf{W}_{t-1}\phi(x_i) + \|\mathbf{y}_i\phi(x_i)^T\|_F^2 \\ &\leq \|\mathbf{W}_{t-1}\|_F^2 + 2 + \|\mathbf{y}_i\|^2\|\phi(x_i)\|^2 \\ &\leq \|\mathbf{W}_{t-1}\|_F^2 + 3 \\ &\leq 3t. \end{aligned}$$

We now provide a reverse inequality for the inner product with $\mathbf{W}^\star$:

$$\begin{aligned} \langle \mathbf{W}_t, \mathbf{W}^\star \rangle_F &= \langle \mathbf{W}_{t-1}, \mathbf{W}^\star \rangle_F + \langle \mathbf{y}_i\phi(x_i)^T, \mathbf{W}^\star \rangle_F \\ &= \langle \mathbf{W}_{t-1}, \mathbf{W}^\star \rangle_F + \langle \mathbf{y}_i, \mathbf{W}^\star\phi(x_i) \rangle \\ &\geq \langle \mathbf{W}_{t-1}, \mathbf{W}^\star \rangle_F + 1 \\ &\geq t. \end{aligned}$$

Now we can create the squeezing inequality:

$$3t\|\mathbf{W}^\star\|_F^2 \geq \|\mathbf{W}_t\|_F^2\|\mathbf{W}^\star\|_F^2 \geq \langle \mathbf{W}_t, \mathbf{W}^\star \rangle_F^2 \geq t^2.$$

implying the result. □

Sparsity bounds [5] can also be used to translate this bound on the number of updates into a corresponding bound on the generalisation of the resulting classifier.

5 Experiments with the multiclass SVM

In the test procedure of the vector output SVM we used multiclass classification problems from the UCI Repository of machine learning datasets [3]. The data sets chosen mostly correspond to those used by Rifkin et al. [9] to give a well-defined benchmark for comparison. Table 2 shows these sets and their descriptors.

We used similar configurations to those described in [9], the kernel type for the feature spaces was Gaussian.

Name	Number of Training Items	Test Items	Classes	Numerical/ Nominal attr.
abalone	3133	1044	29	8/1
glass	214	*	7	9/0
optdigits	3823	1797	10	64/0
page-blocks	5473	*	5	10/0
satimage	4435	2000	6	36/0
spectrometer	531	*	48	101/0
yeast	1484	*	10	8/0

Table 2: Parameters of the data sets used in the experiment. * denotes the datasets with no dedicated training and test subsets.

Name	Number of subSVMs all vrs. all	one vrs. all	vector output
abalone	406	29	1
glass	15	6	1
optdigits	45	10	1
page-blocks	10	5	1
satimage	15	6	1
spectrometer	1128	48	1
yeast	45	10	1

Table 3: Number of binary classifiers computed in one multiclass classification problem

Table 3 shows the number of optimisation that need to be solved for the different problems. This is only an indication of the computational effort since they will not all be of the same size. Here we should mention that the elementary binary classifiers in the "one versus one" are generally much simpler than they are in the other methods but their number can be enormous.

In Table 4 the values for the methods "one versus all" and "one versus one" are borrowed from [9] as well. we should emphasize that if the computational complexity of a learner is small then a systematic scanning of the parameter space for an optimal configuration remains sufficiently cheap, so that using a validation set better (and sometimes much better) accuracies can be achieved. For example, the presented test results for the Glass and the Spectrometer use this approach.

6 Conclusions

In this paper we have shown that multiclass learning is expressible in a simple optimization framework and this sort of simplicity not only preserves the accuracy but may improve it. Furthermore it suggests a vector version of the perceptron algorithm with margin (or τ-perceptron) for which the number of updates can be bounded in terms of the optimal margin obtainable [5].

In further research we plan to make a similar reduction of the complexity for structural learning. The simplicity and transparency of the learning methods in this formulation can give strong support to the generalization theory as well by removing unnecessary technical complications.

An interesting and fruitful extension of our approach is to use objects in infinite dimen-

Table 4: Test error rates (%). If the data set has dedicated training and test subsets, marked with *, then the table shows the accuracy computed on the given test subset otherwise the presented accuracies are averages computed via 10-fold cross-validation.

	Test error rate (%)			
			vector output	
Name	all vrs. all	one vrs. all	Perceptron	voSVM
abalone *	**72.3**	79.7	78.6	75.4
glass	30.4	30.8	44.6	**24.3**
optdigits *	3.8	2.7	10.0	**1.5**
page-blocks	3.4	3.4	5.5	**3.2**
satimage *	8.2	**7.8**	17.3	8.5
spectrometer	42.8	53.7	61.3	**35.8**
yeast	41.0	**40.3**	46.8	**40.3**

sional Hilbert spaces, that is to learn when the input and the output are real valued functions exploiting the simplicity and finiteness of the dual problem.

References

[1] M. Barker and W. Rayens. Partial least squares for discrimination. *Journal of Chemometrics*, 17:166–173, 2003.

[2] D.P. Bertsekas. *Nonlinear Programming*. Athena Scienctific, second edition edition, 1999.

[3] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[4] T. Evgeniou, C.A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research, 6(Apr)*, pages 615–637, 2005.

[5] T. Graepel, R. Herbrich, and J. Shawe-Taylor. Generalisation error bounds for sparse linear classifiers. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 298–303. Morgan Kaufmann Publishers Inc., 2000.

[6] K.C. Kiwiel. Convergence of approximate and incremental subgradient methods for convex optimization. *Journal of Optimization*, 14, 3:807–840, 2004.

[7] C.A. Micchelli and M. Pontil. Kernels for multi-task learning. In *Proc. of the 18-th Conf. on Neural Information Processing Systems (NIPS'04)*. 2004.

[8] C.A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005.

[9] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[10] R. Rosipal, L. J. Trejo, and B. Matthews. Kernel pls-svc for linear and nonlinear classification. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003) Washington DC*. 2003.

[11] R. Rosipal and L.J. Trejo. Kernel partial least squares regression in reproducing kernel hilbert space, 2001.

[12] 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, Egham, TW20 0EX, UK, 1998.