Efficient Implementation of SVM for Large Class Problems

P. Ilayaraja, Neeba N. V. and C.V. Jawahar
Center for Visual Information Technology,
International Institute of Information Technology, Hyderabad, India - 500 032

Abstract

Multiclass classification is an important problem in pattern recognition. Hierarchical SVM classifiers such as DAG-SVM and BHC-SVM are popular in solving multiclass problems. However, a bottleneck with these approaches is the number of component classifiers, and the associated time and space requirements. In this paper, we describe a simple, yet effective method for efficiently storing support vectors that exploits the redundancies in them across the classifiers to obtain significant reduction in storage and computational requirements. We also present our extension to an algebraic exact simplification method for simplifying hierarchical classifier solutions.

1. Introduction

Multiclass pattern classifiers have significant applications in many real-life problems. Recent comparative studies have argued that Support Vector Machine (SVM) based classifiers provide the best results on a wide variety of data sets [1]. SVMs were originally designed for binary (two-class) classification. Direct extension of SVM to multiclass classification is not attractive due to the complexity of the associated optimization task [2]. Therefore, multiclass problems are usually solved using several independent binary classifiers [3, 4].

Figure 1 shows the basic architecture of DAG and BHC for 4-class and 8-class problems respectively. Each node represents a trained binary-SVM classifier designed for a specific pair (or set) of classes. Each of these classifiers take decision based on the associated support vectors (s_i) it has, using $f(x) = \sum_i \alpha_i y_i K(x, s_i) + b$. Here, x is the test sample, α_is are the lagrangians and y_is are the predictions corresponding to the s_i. A node contains an array A of scalar values $(\alpha_i \cdot y_i)$ (we refer this product as A_i) and another array V of support vectors (SVs). Support Vectors are of dimension D, where D is the feature dimension. Clearly, the storage and computational complexity of each node is proportional to the number of SVs it has.

There are well established methods that reduce the complexity of SVMs by reducing the number of SVs. Some of them are *exact* simplification methods while others aim at *approximate* reductions. Burges [5] introduced a method for determining a reduced set of vectors from the original SVs. The reduced set of vectors is computed from the original support vector set such that it provides best approximation to the original decision surface. However the method proved to be computationally expensive. Downs *et al.* [6] presented a method for $exactly$ simplifying SVM solutions. The method eliminates unnecessary SVs and modifies the Lagrangians of the remaining SVs so that the solution remains unchanged.

In this paper, we describe an effective multiclass reduction using an ideal data structure, exploiting the redundancies in hierarchical classifiers. We also propose an algorithm that extends the application of exact simplification method for further reduction in classification time.

2. Multiclass Data Structure

In this section, we describe first a Multiclass data approach for efficiently storing and accessing SVs ($Fig.\ 1.c$). It consists of two major components. First one is a set of nodes, each of which represents a *modified* independent pair-wise classifier node. Second one is a list of vectors L, containing *reduced* set of SVs for the multiclass problem. The effectiveness of this approach comes from the following change in the node structure. The first scalar array A in the node is retained as such, while the second array of vectors that stored the SVs in the original node are replaced with a scalar array $INDEX$. This second array now stores the index positions of the corresponding SVs, that are moved to list L.

A primitive implementation treats the pairwise clas-

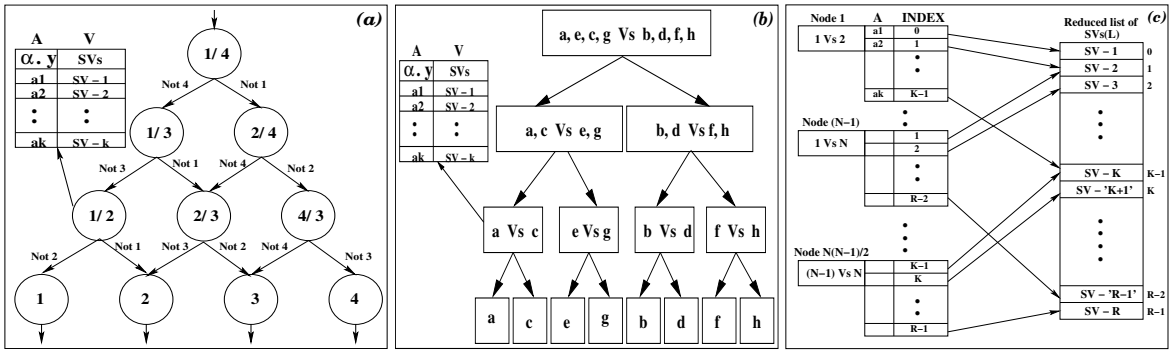

Figure 1. (a)DAG with independent binary classifiers. (b) BHC architecture (c) Storage schema for multiclass problem. Support vectors are stored in a single list (L) uniquely.

sifiers as independent. Such an implementation treats the set of SVs that belong to a particular binary classifier to be *independent* of SVs that belong to other binary classifiers. Hence it stores the SVs of each binary classifier at the corresponding node($See\ 1(a),(b)$). This multiclass implementation breaks the independence assumption and maintains a single list (L) of all SVs, thereby allowing component binary classifiers to point a single instance of the *shared* SVs. Thus it brings a true and exact multiclass reduction, exploiting the redundancies in a multiclass scenario. This helps in scaling the solution for large classes as explained below.

Suppose we have a DAG with S support vectors and only R of them are unique. Though MDS adds an extra storage $(S \times i)$ for indexing, it is negligible considering the amount of reduction in the storage of SVs for large class problems. Our experiments show that for a 300-class problem R is only 1% of S. As N increases, the space reduction approaches $\frac{S-R}{S}$, since the space requirement of A and $INDEX$ are negligible compared to that of SVs. SVs are stored as a vector/array and not as a list of index and content.

Though the $\frac{N(N-1)}{2}$ pairwise classification problems are independent, we observe that many of these classifiers share SVs. This is because of the fact that SVs are the samples on the class boundaries. Therefore, even if the number of classes increase, the unique SVs in the solution increase only marginally. This is the primary reason for obtaining very high reduction in space requirement. Figure 2 shows that as many binary classifiers as there are in the decomposed solution, the dependency among them will also be high for any type of kernel. $i.e.$ the number of $shared$ SVs is more. Hence, as we combine the SVs of the binary classifiers into a unique reduced set, the reduced set converges only with SVs from a fraction of binary classifiers.

Algorithm 1 shows the computations performed by a

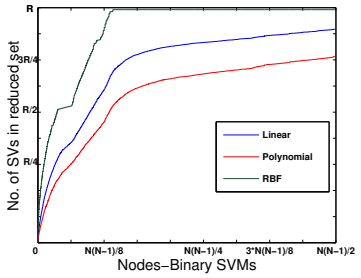

Figure 2. Dependency analysis

binary SVM for classifying a sample. It makes use of the MDS architecture. The costly kernel evaluations are saved for duplicate SVs and thus achieving a significant reduction in the evaluation cost.

Algorithm 1 SVM_CLASSIFY(Node, Sample)

```
for i = 1 to (Node→NumOfSVs) do
    index ← (Node → INDEX[i])
    if FLAG[index] = 0 then
        KERNEL[index] ← K(Sample, L[index])
        FLAG[index] ← 1
    end if
    Add KERNEL[index] × (Node → A[i]) to D
end for
Add (Node → b) to D
RETURN sign of D
END SVM_CLASSIFY
```

In all our experiments, SVM^{light} [7] implementation was used as binary classifier. There are multiclass SVM implementations available from libraries such as $LIBSVM$, $SVM^{multiclass}$, and SVM^{Torch}. LIBSVM stores the SVs uniquely, similar to the approach discussed above. However it needs to be refined for optimizing the kernel evaluations for a large class problem for minimizing the redundant kernel evaluations as

the 1. SVM^{light} computes a single linear weight vector $w = \sum_i \alpha_i y_i s_i$ for linearly separable problems. There exists a large number of real life problems that are linearly separable. So the binary classifier stores only w and need not require the SVs for classification. Therefore the multiclass solution stores only $(N(N-1)/2)$ linear weights and uses $(N-1)$ of them for classifying a sample in the case of a DAG. Though this linear weight method gives better space reduction for 10 and 50-class problems, MDS has better reduction for large classes, since $R < N(N-1)/2$ for $N >= 100$.However, $R > (N-1)$ for all $N = 10 \ldots\ldots 300$, hence linear weight method has faster classification rate. (The serious limitation of the linear weight method is that it is not applicable to non-linear kernels). While handling a large class problem, usually the model file(file containing classifier parameters and SVs) size will be in Gigabytes. Hence it is a advisable to write the model file in binary format to reduce the storage requirement.

Table 1 shows the results against two UCI data sets. The reduction in SVs for linear and polynomial kernels are observed to be closer, while RBF kernel gives higher reduction rate. Because RBF solution picks more samples as SVs, that leads to more redundancy among the SVs from the binary solutions. Also, comparing the results from PenDigits and Letters data sets shows that the reduction rate increases with increase in number of classes, irrespective of the type of kernel used.

Experiments with large class character recognition data set, with 300 classes, were performed to test the scalability of our proposed implementation. A 10 class problem , a subset of the dataset, gives 66.02% reduction in SVs and 15.47% reduction in the classification time, while we obtain 98.5% of reduction in SVs and 60% reduction in classification time for linear and polynomial kernels on the 300-class data set. This huge reduction in storage is in comparison with a direct/native implementation assuming that each node is solving an independent problem. The time reduction is lesser than the reduction in SVs obtained, since classifying a sample involves only $N-1$ binary classifiers and not all the $N(N-1)/2$. The reduction in SVs gradually increases as N goes from 10 to 300. All our experiments are conducted on DAG, but similar kinds of reduction can be achieved in a BHC also.

3. Hierarchical Simplification of SVs

Downs *et al.* [6] introduced a method called Exact Simplification for recognizing and eliminating unnecessary SVs in SVMs. The method reduces a given set of SVs by finding those SVs that are linearly dependent of other SVs in the feature space and removing them from the solution. Suppose we have a support vector solution that has r SVs that determine the decision surface

Data set Name	Kernel Type	No. of SVs	
		IPI(S)	MDS(R)
PenDigits (10-class)	Linear	5788	2771
	Poly.	3528	1777
	RBF	67450	7494
Letters (26-class)	Linear	113249	15198
	Poly.	80553	12961
	RBF	482975	18666

Table 1. MDS Vs IPI on UCI data sets.

$$f(x) = \sum_{i=1}^{r} \alpha_i y_i K(x, s_i) + b. \quad (1)$$

Now suppose that SV x_k is linearly dependent on other SVs in the feature space, *i.e.*, $K(x, s_k) = \sum_{i=1, i \neq k}^{r} c_i K(x, s_i)$, where the c_i are scalar constants. Then the solution can be simplified as $f(x) = \sum_{i=1, i \neq k}^{r} \overline{\alpha_i} y_i K(x, s_i) + b$, where $\overline{\alpha_i}$ are the updated Lagrangians that keeps the solution otherwise unchanged.

A direct extension of this exact simplification to any hierarchical multiclass SVM solutions is to apply the reduction method on each of the component classifiers. This reduces each component classifier by removing a subset of SVs, leaving only the linearly independent SVs in their corresponding solutions. However, the obtained reduction from each component classifier need not be the best for the overall multiclass solution. At the same time, the method cannot further eliminate any SVs since all of them are now linearly independent within a component solution.

We can eliminate SVs further from the solutions of the component classifiers, if we break the linear independence. This is possible if we add new vectors to each component solutions. Suppose we have a component classifier that has a reduced set of SVs I that are linearly independent. We could add a new set of vectors to I to get an extended set of vectors E. If the extended set of vectors are no longer linearly independent, then we can further reduce the component classifier by eliminating many SVs. Note that while we are eliminating a subset of SVs from I, we have already added some new vectors to the solution to get the benefit. The addition of new vectors is justifiable and do not bring any extra cost when we add the SVs that belong to component classifiers that are up in the hierarchy in a decision path to the one at a lower level for reducing the later. Since the kernel computations $K(x, s_i)$ for those SVs once computed at any component classifier higher in the decision

path are reusable anywhere down the decision path.

The hierarchical simplification involves two steps as given in Algorithm 2. The algorithm also simplifies a multiclass solution $exactly$, by reducing the number of SVs as explained below.

Algorithm 2 Hierarchical Exact Simplification(HC: Hierarchical Classifier)

Step 1: Individual component reduction.
for each component classifier (node) $'X'$ in HC **do**
  Reduce $'X'$ using Exact Simplification method.
end for
Step 2: Reduction across component classifiers.
for each decision path $'P'$ in HC **do**
  for each internal node $'X'$ in $'P'$ **do**
    Extend the set of SVs in $'X'$.
    Identify the linearly dependent SVs $\in X$ and eliminate them.
  end for
end for

Suppose a component classifier X has its solution of the form Equation 1. Let V denote the set of SVs in the solution. $Step\ 1$ of the algorithm reduces V to a subset I of SVs and the solution becomes

$$f(x) = \sum_{i=1, s_i \in I}^{r} \overline{\alpha_i} y_i K(x, s_i) + b \qquad (2)$$

$Step\ 2$ of the algorithm now adds a set of SVs A that are picked up from any of the component classifiers above X in the decision path. Let $A = a_1, a_2, ..., a_k$ be the SVs with corresponding Lagrangians $\alpha_{a_1}, \alpha_{a_2}, ..., \alpha_{a_k}$ that are added to I. The exact simplification method is now applied on the extended set $E = I \cup A$. Let a set of SVs $R \subseteq I$ are recognized as linearly dependent on other SVs $\in E$ in the feature space, $i.e.$ for each $s_k \in R$, $K(x, s_k) = \sum_{s_i \in (I-R)} c_i K(x, s_i) + \sum_{a_j \in A} c_j K(x, a_j)$ where the c_i and c_j are scalar constants.This allows the algorithm to eliminate the SVs $\in R$ from E resulting in a reduced set $\overline{V} = (I - R) \cup A$. Hence the final solution for X is of the form

$$f(x) = \sum_{\substack{i=1 \\ s_i \in (I-R)}}^{r} \overline{\overline{\alpha_i}} y_i K(x, s_i) + \sum_{j=1}^{k} \overline{\alpha_{a_j}} y_{a_j} K(x, a_j) + b \qquad (3)$$

Note that the lagrangians of the SVs $\in A$ are now updated as per the exact simplification method. However this updated values are only used in computing the solution for X, leaving the original lagrangians at the corresponding component classifiers intact.

Table 2 shows the computational advantage obtained by applying Algorithm 2 on standard datasets. The extended set in $Step\ 2$ was obtained by adding all the SVs from nodes that are higher in a decision path to a particular node to reduce the later and the procedure was repeated iteratively. Note that the computational advantage comes with an additional storage requirement of modified lagrangian values. The reductions obtained are problem dependent in both the steps as also observed by Downs *et al.* [6] in their experiments.

Dataset (# Class)	#Dim.	Reduction(%)		
		$Step\ 1$	$Step\ 2$	Overall
PenDigits (10)	16	85.42	71.49	95.84
Letters (26)	16	94.87	17.78	95.60
OptDigits(10)	64	59.25	54.92	81.63
Vowel(11)	10	76.89	68.90	92.81

Table 2. Reduction in classification time (using linear kernel).

4. Conclusion

In this paper we have presented an efficient method for implementing multiclass solutions. We had shown that our method reduces both space and time complexity significantly on multiclass problems and the reduction becomes enormous on large class problems. The hierarchical exact simplification method reduce the number of necessary support vectors. However as the number of classes increases, the increase in number of decision paths becomes a challenge for scalability that needs to be addressed in future.

References

[1] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *ICML*, 2006.

[2] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. 2000.

[3] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *ANIPS*, pp. 547–553, 2000.

[4] T. K. Chalasani, A. M. Namboodiri, and C. V. Jawahar, "Support vector machine based hierachical classifieds for large class problems," in *ICAPR*, 2007.

[5] C. J. C. Burges, "Simplified support vector decision rules," in *ICML*, vol. 13, 1996.

[6] T. Downs, K.E.Gates, and A. Masters, "Exact simplification of support vector solutions," in *Journal of Machine Learning Research*, vol. 2, pp. 293–297, 2001.

[7] T. Joachims, "Making large-scale svm learning practical," in *Advances in Kernel Methods - Support Vector Learning,MIT-Press*, 1999.