Support Vector Machine based Hierarchical Classifiers for Large Class Problems

Tejo Krishna Chalasani, Anoop M. Namboodiri, C.V. Jawahar
Center for Visual Information Technology, International Institute of Information Technology, Hyderabad, India
E-mail: tejokrishna@students.iiit.ac.in, anoop@iiit.ac.in, jawahar@iiit.ac.in

One of the prime challenges in designing a classifier for large-class problems such as Indian language OCRs is the presence of a large similar looking character set. The nature of the character set introduces problems with accuracy and efficiency of the classifier. Hierarchical classifiers such as Binary Hierarchical Decision Trees (BHDTs) using SVMs as component classifiers have been effectively used to tackle such large-class classification problems. The accuracy and efficiency of a BHDT classifier will depend on: i) the accuracy of the component classifiers, ii) the separability of the clusters at each node in a hierarchical classifier, and iii) the balance of the BHDT. We propose methods to tackle each of the above problems in the case of binary character images. We present a new distance measure, which is intuitively suitable when Support Vector Machines are used as component classifiers. We also propose a novel method for balancing the BHDT to improve its efficiency, while maintaining the accuracy. Finally we propose a method to generate overlapping partitions to improve the accuracy of BHDTs. Comparison of the method with other forms of classifier combination techniques such as $1vs1$, $1vsRest$ and Decision Directed Acyclic Graphs shows that the proposed approach is highly efficient, while being comparable with the more expensive techniques in terms of accuracy. The experiments are focused on the problem of Indian language OCR, while the framework is usable for other problems as well.

Keywords: Binary Hierarchical Decision Tree, Support Vector Machine, Decision Directed Acyclic Graph.

1. Introduction

Efficient and accurate OCR engines play a highly critical role in making the information present in large quantities of document images, available for searching and indexing. The challenges in developing OCR systems for Indian languages are different from that of English due to a variety of reasons. Unlike English, most of the Indian languages have a large number of characters in their scripts, which makes the task of designing a classifier for them, much more difficult. The characters in Indian languages are formed as a composition of basic shapes and sometimes also a composition of basic characters. This composition not only leads to a large number of characters because of the numerous possible combinations, but also similar looking characters, which makes the designing of the classifier more difficult.

Support Vector Machines (SVMs) [1] that build large margin classifiers for binary class classification problem, they have proved to have high generalization performance both theoretically and empirically. The SVM formulation tries to find a hyper plane that divides a set of two classes with largest margin. Extending this formulation of SVM directly to more than two classes is generally avoided due to the complex optimization equation it leads to. Instead, the multi-class SVM problem is dealt with by using an ensemble of two-class SVMs. There are various strategies to achieve the combination, of which 1vs1, 1vsRest, and hierarchical classification are the popular methods. Consider a problem with N classes. In 1vRest strategy, N two-class classifiers are trained, where the ith classifier is trained considering the i-th class samples as the positive class and all the other samples as the negative class. When an unseen sample is given for testing, the distance from the separating plane is calculated for each classifier and the sample is assigned the label of that classifier for which it is farthest from the separating plane. In the 1vsRest strategy, a classifier is trained for each pair of classes, resulting in ${}^NC_2$ classifiers. When a sample is given for classification, all the ${}^NC_2$ classifiers are used and a vote is taken from each classifier. The sample is assigned the label of the class that has the maximum votes.

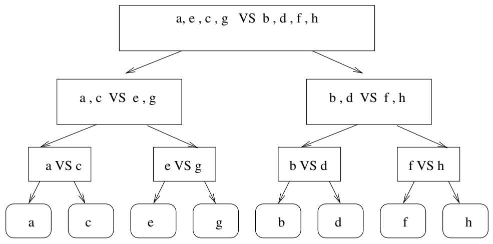

Fig. 1. A BHDT that classifies the first 8 alphabets of English.

Hierarchical decision classifiers divide a complex problem into simpler ones and tackle the sub-problems thus created. The results from these sub-problems are integrated to solve the main problem. Decision Directed Acyclic Graphs (DDAGs) and Binary Hierarchical Decision Trees (BHDTs) are two such popular hierarchical classification methods, which extend a binary classifier to multi-class classification using different ensemble strategies.

A Decision Directed Acyclic Graph using SVM has been used successfully for multi-class classification problems [3]. The disadvantage with this approach is that the evaluation time taken for an unseen sample is of $O(N)$ where N is the number of classes, since the number of classes is huge for IL-OCR it is advisable to look for a logarithmic order, which becomes the prime motivation for exploring Binary Hierarchical Decision Trees.

Hierarchical Classification has been used for character recognition to considerable success using DDAGs, BHDTs or a hybrid of both [4,5]. The emphasis of the existing work has been on designing of DDAGs [3] or on hybrid techniques [4,5], where at each node in the tree, a decision is made whether to choose a DDAG or a BHDT, depending on the complexity of the decision boundary. Another possible approach is to build regular decision trees for a particular problem and then replace the classifier at each node with a large-margin classifier [6] such as SVMs. However, many issues in designing BHDTs for applications such as character recognition have not been examined in sufficient detail.

A BHDT for N classes has $N-1$ binary classifiers arranged as a binary tree, with N leaf nodes, where each leaf node represents a class. The BHDT is built by combining the classes recursively, two at a time, and a classifier learnt between the two groups. Figure 1 gives an example of a BHDT that classifies the first 8 alphabets of English language.

The running time of algorithms are often described using the Big-Oh (O(.)) notation, which gives the time required to solve an instance of the problem in terms of its size, N [7]. The running times are usually reported for the worst case and the average case. The average running time for labeling an unseen sample will be $O(log(N))$, while the worst case time complexity is $O(N)$. However, computing the optimal partition for a BHDT is NP-Complete [8], that makes problem intractable as the number of classes increase. Hence, one needs to resort to approximate solutions in practice.

The primary issues that affect the accuracy and efficiency of a BHDT that uses SVMs as component classifier are: i) the use of an appropriate distance measure in computing the clusters in a BHDT, ii) maintaining the balance in cluster sizes to improve efficiency, and iii) dealing with error rates that cascade with the levels of the tree.

In this paper, we present a new distance measure (Section 2.1) that intuitively suits the SVM (the binary classifier to be used at each node of the binary decision tree) Section 2.2 presents a novel approach to balancing the tree. Section 2.3 proposes an approach for improving the accuracy of the BHDT using overlapping partitions at each node in the tree. Our conclusions and future directions of work are presented in Section 3.

All the experiments are performed on Telugu OCR data set with 329 classes. Each segmented character is scaled to a size of 25x25 pixels maintaining the aspect ratio and then binarized. Each such character is taken and the 25X25 matrix is converted to a single row my appending the rows consecutively resulting in a 625-dimensional feature vector. A linear kernel is used for the SVMs in all the experiments since the performance was comparable (or sometimes better) to any polynomial kernel. All the results are provided in Section 2.4. As a reference point for the results, we provide the accuracies of various classifiers such as K-Nearest Neighbor, Artificial Neural Networks, and various ensembles of SVMs in Table 1.

Table 1. Comparison of accuracies of various classifiers. The classifiers considered are K-Nearest Neighbors, Artificial Neural Networks, and 4 different ensemble strategies for SVMs

Training samples per class	15	20	25	30
KNN	80.29%	83.87%	87.34%	91.81%
ANN	85.78%	88.45%	90.38%	89.64%
1vsRest	20.31%	22.31%	25.31%	25.97%
1vs1	93.28%	97.91%	98.01%	98.74%
DDAG	91.97%	95.89%	97.52%	98.86%

The results for 1vRest ensemble strategy are particularly bad because of the imbalance in training samples of the +ve and -ve classes, for example consider the case of 20 samples per class, and we are designing a the $i-th$ classifier meaning it will be give 20 +ve sample and 6560 -ve samples, because of which +ve class is not represented properly. Though the 1v1 ensemble strategy gives a good experimental results, every training sample has to go through ${}^{N}C_{2}$ classifiers for the it to be labeled making it quadratic in time and impractical.

2. BHDT for IL-OCR using SVMs

In this section, we describe the specific approaches that we propose to improve the accuracy and efficiency of the BHDT classifier:

- The use of an appropriate distance measure,
- Improving balance of the BHD Tree
- Dealing with cascading error rates within the tree.

2.1. *Distance Metric for Binary Decision Tree*

A BHDT contains a decision tree is that partitions the classes into two sets at each node. The partitioning is often achieved by a hierarchical clustering algorithm [8], and the accuracy of the classifier depends on the clusters generated. Different clustering algorithms such as k-means, agglomerative clustering and graph-cut based partitioning can be used to cluster the classes, and they use a distance measure between samples to achieve the clustering. A good distance metric should be invariant to the type of features, and should be robust and easy to compute on small training sets. It should also be compatible with binary classifier used at each node. Commonly used distance metrics such as Euclidean distance does not work well with binary features for clustering [9]. Other metrics such as Mahalanobis distance and Kullback-Leibler distance [10] needs a large number of samples per class for robust estimates. However, the segmented characters used for recognition are binary; and having a large number of samples per class with already huge number of classes often makes the training process prohibitively expensive.

As the clustering algorithm tries to group character classes, we propose a new distance measure based on the separability of the classes. To compute the distance measure, an SVM classifier is trained between each pair of classes. The margin of the classifier for a class pair is used as the distance between the two classes. We use the single link clustering algorithm with the above distance metric to build the tree. This clustering algorithm considers all classes to be different clusters and merges the nearest classes and continues the process till two clusters are left [2].

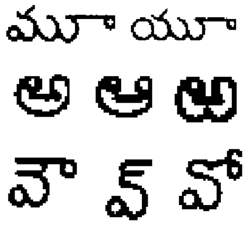

Fig. 2. Each row contains a character cluster that is placed near the leaves by the proposed clustering scheme. Note the similarity in shapes.

Since margin can be used as a measure of classifiability, the classes at the leaf nodes should be those that are most difficult to classify and this observation has been consistent with our experiments (See Fig 2). The results on Telugu OCR data set when the proposed metric is used as the distance measure in comparison with Euclidean distance are provided in Section 2.4.

2.2. *Balanced BHDTs (BBHDT)*

A balanced tree can bring down the bound on worst case time complexity from $O(N)$ to $O(logN)$. This can considerably increase the speed of classification in large class problems such as IL-OCR. We present an algorithm that balances the tree with only minimal reduction in accuracy. Let $P(\omega_i)$ be the *apriori* probability for the class $\omega_i \in \Omega$ and d_{xi} be the distance from the current node, x, its successor, corresponding to class ω_i. Each node x is given a weight $W_x = \sum_{i \in subtree(x)} d_{xi}.P(\omega_i)$. The weight of the node is the expected time to classify a sample that arrives at x. Let the imbalance of a node x be defined as $imB_x = |W_{x \rightarrow left\_child} - W_{x \rightarrow right\_child}|$. Let T be the root node of tree that is built using some clustering algorithm for the given classes. With this notation Algorithm 2.1 describes the balancing procedure, that balances the tree in a recursive fashion. The idea is to rectify the imbalance at the node, by pushing the classifier at x to a lower position in the heavier subtree.

Algorithm 2.1 Balance Tree(T, δ)

```
1:  if $T = leafnode$ then
2:      return $T$
3:  end if
4:  Balance Tree($T \rightarrow left\_child$, $\delta$)
5:  Balance Tree($T \rightarrow right\_child$, $\delta$)
6:  if $imB_T > \delta$ then
7:      $T' = push(T)$
8:      if $imB_{T'} < imB_T$ then
9:          $T = T'$
10:         updateWeigth($T$)
11:     end if
12: end if
```

Let C_{max} denote the child of node C that has the maximum weight and C_{min} child with the minimum weight. $push$ operation is defined in Algorithm 2.2 using this notation. Figure 3 shows an illustration of the push operation and the balance change that results.

The algorithm first balances the left and right children at each node and then calculates the imbalance. If the imbalance is greater than δ, a parameter set by the user, the operation $push$ takes place. $push$ is committed

Algorithm 2.2 push (T)

```
1: $T' = T$
2: $T = T_{max}$
3: $tmp = T_{max_{min}}$
4: $T_{max_{min}} = T'$
5: $T'_{max} = tmp$
```

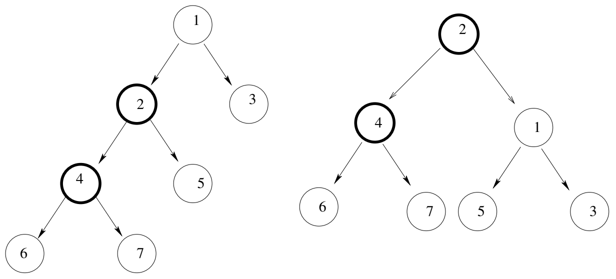

Fig. 3. Push operation: The darker nodes represent the roots of heavier subtrees. Note that the overall imbalance is reduced after the push.

only if the imbalance that is created after the push operation is lesser than that of the imbalance present before. $push$ operation also ensures that the natural clustering is not disturbed if the value of imbalance is large. The accuracies before and after balancing the decision tree using the above algorithm are reported in Section 2.4

2.3. *Overlapping BHDT (OBHDT)*

One of the reasons for the lower accuracy of BHDT when compared to DDAG is the presence of overlapping clusters at certain nodes in the BHDT. We now present an algorithm that takes this overlap into consideration, and modifies a BHDT to improve its accuracy. When classifiers are designed between two large clusters of classes, the accuracy often suffers as some classes will overlap the cluster decision boundaries. To get around this problem, we introduce the overlapping subset of classes into both the clusters. This will postpone the classification of such classes until the clusters are smaller and more manageable. The algorithm builds on the already existing BHDT. We use an evaluation set to identify the nodes at which misclassifications occur. Algorithm 2.3 shows how to build the OBHDT.

The parameter τ controls the overlap at which we decide to add a class to both the clusters at a node. The results of OBHDT algorithm in comparison to the results of using a DDAG and BHDT on Telugu OCR data set are presented Section 2.4.

Algorithm 2.3 Build OBHDT(T)

```
1: for each class $\omega_i$ do
2:     Let $t'$ be the first node at which $\omega_i$ gets misclassified
3:     pushclass($t'$,$\omega_i$)
4: end for
```

Algorithm 2.4 pushclass(T, ω_i)

```
1: if $T$ is leafnode with class $\omega_x$ then
2:     build classifier $\omega_i \ v \ \omega_x$ at $T$
3:     return
4: end if
5: if above $\tau\%$ of samples of $\omega_i$ fall to $x\_child$ then
6:     pushclass($T \rightarrow x\_child, \omega_i$)
7: else
8:     pushclass($T \rightarrow left\_child, \omega_i$)
9:     pushclass($T \rightarrow right\_child, \omega_i$)
10: end if
```

2.4. *Experimental Results*

In this section we present the results of various algorithms and experiments done on the Telugu OCR data set. Telugu language script has 329 character classes, which makes it challenging for us to classify and correctly, label the unseen data.

Table 2 shows the improvement of classification when the margin as specified in section 2.1 between the classes is used as a distance measure to partition the data set as opposed to Euclidean distance. Mahalanobis distance based metric was not able to give sufficient separation between clusters for the SVM classifier to converge.

Table 2. Comparison of accuracies with different distance metrics used for clustering.

Training samples per class	15	20	25	30
Euclidean distance	80.04%	83.27%	84.82%	86.95%
Margin	87.70%	89.27%	92.78%	96.30%

Table 3 gives the accuracies and time taken, which is measured as the weight of the root node before and after applying the Balance Tree Algorithm 2.1. Since we don't have any apriori information about the data we considered all are equally probable, so the value of $P(w_i)$ is made $1/n \forall i$ to calculate the weight at each node as defined in Section 2.2. The column, $time$, specified in the table 3 is the weight (calculated according to the formula in Section 2.2)of the root node of the BHDT, which is the expected number of classifications each sample might take.

Table 3. Accuracy(acc) and time taken before and after applying the balance tree algorithm.

Train. samples	20		25		30	
	acc	time	acc	time	acc	time
BHDT	89.27%	30.32	92.78%	32.25	96.30%	32.01
BBHDT	90.03%	23.37	91.31%	25.21	95.71%	23.04

Table 4 shows the improvement in accuracy of the overlapped version of BHDT over the regular BHDT, while maintaining the efficiency. The value of τ in Algorithm 2.3is set to 75% for the experiment and the time taken in the case of OBHDT was generated using a test set with 10 samples per class, and noting down the average number of classifications.

Table 4. Comparison of accuracies and time taken for the various ensemble of SVM classifiers.

Training samples per class	15	20	25	30	time
DDAG	91.97%	95.89%	97.52%	98.86%	328
BHDT	87.70%	89.27%	92.78%	96.30%	32
OBHDT	91.34%	96.11%	98.05%	98.91%	34.5

Table 5 shows the performance of the BHDT classifier on various datasets chosen from The UCI machine learning dataset repository. The datasets that were chosen had larger number of classes, to demonstrate the ability of the proposed classifier design algorithm. Note that the classifier under comparison is the non-overlapping BHDT, and the accuracies can be further improved by considering the overlapping version, at the cost of a slight increase in classification time.

Table 5. Performance comparison with popular classifiers on various datasets.

Classifier Dataset	ANN	KNN	DDAG	BHDT
optdigits	89.76%	93.43%	95.46%	95.53%
pendigits	90.34%	97.74%	96.32%	97.01%
glass	55.65%	76.47%	80.41%	81.31%
yeast	73.35%	48.36%	76.01%	75.89%

3. Conclusions and Future Work

We note that a hierarchical classifier system performs better when separability(margin) is used as the distance metric for partitioning the data sets. An overlapping partitioning scheme is proposed that increases the accuracy of a BHDT, with only minor loss of efficiency. The resulting classifier performs better than DDAGs, while being an order of magnitude smaller in both memory footprint as well as time taken for classification. Further increase in the efficiency of the classifier is obtained using a balancing algorithm. This design suits the large class classification problems such as OCRs, and it can be applied to other problems as well.

One could combine balancing with the clustering algorithm to directly generate more balanced clusters, that could possibly result in efficient classifiers of higher accuracy.

References

1. J.C.Burges , A Tutorial on Support Vector Machines for Pattern Recognition , *Datamining and Knowledge Discovery* **vol 2**, 121–167 (1998)
2. R.Sibson , SLINK: An optimally efficient algorithm for the single-link clustering method , *The Computer Journal* **vol 16**, 30–34 (1973)
3. John.C.Platt,Nello Cristainini , Large Margin DAGs for multi class classification *Advances in Neural Information Processing Systems* **vol 12**, 547–553 (2000).
4. M.N.S.S.K Pavan Kumar, C.V. Jawahar , Design of Hierarchical Classifier with Hybrid Architectures *PReMI* , 276–279 (2005)
5. M.N.S.S.K. Pavan Kumar, C.V. Jawahar, Configurable Hybrid Architectures for Character Recognition Applications *International Conference on Document Analysis and Recognition* **vol 1**, 1199–1203 (2005)
6. D. Wu, K.P. Bennet, N. Christianini, and J.S. Taylor, Enlarging the Margins in Perceptron Decision Trees *Machine Learning* **vol 41**, 295–313 (2000)
7. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C.Stein, Introduction to Algorithms, *MIT Press and McGraw-Hill* **2nd ed.**, (2001)
8. L.Hyafil and R.L.Rivest , Constructing optimal binary tree is NP-Complete , *Information Processing Letters* **vol 5**, 15–17 (1976).
9. Volkan Vural, Jennifer G.Dy, A Hierarchical Method for Multi-Class Support Vector Machines *International Conference on Machine Learning* **vol 69**, 105 (2004)
10. Yangchi Chen, Melba M.Crawford, Joydeep Ghosh , Integrating Support Vector Machines in a Hierarchical Output Space Decomposition Framework *IEEE International Geoscience and Remote Sensing Symposium* **vol-2**, 949-952 (2004)