Configurable Hybrid Architectures for Character Recognition Applications

M. N. S. S. K. Pavan Kumar and C. V. Jawahar
Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad, INDIA
jawahar@iiit.ac.in

Abstract

Character recognition is a multiclass problem with typically large number of classes. Hierarchical classifiers are found to be suitable for such classification problems due to their low space and time complexities. However, most hierarchical classifiers employ similar classifiers at different levels of the hierarchy. This may not be ideal, as the complexity of classification is not same at all the stages. In this paper, we propose an algorithm to build a hybrid hierarchical classifier by choosing the classifiers of appropriate complexity at each level of the hierarchy. We demonstrate that hierarchical combination of complex classifiers improves the classification performance significantly.

1. Introduction

Optical Character Recognition (OCR) applications range from massive tasks like digitization of large document databases, to tiny applications in a PDA or a cell phone for text access while being mobile. Each of these applications have different requirements and constraints on parameters like the size, speed and accuracy. English has smaller set of characters which makes it easy to build OCR applications of different sizes and performances. However, this was not possible with Asian and African scripts owing to the large size of their alphabet.

In many of the Asian and African scripts, the characters are formed by conjunctions of basic shapes. Owing to the large number of possible combinations, these scripts have a very large number of characters. Moreover, the characters which share one or more basic shapes are highly similar to each other, making the classification task further difficult. Building a classifier to recognize all the characters independently is not possible as the resulting classifier would be prohibitively large. Many of the characters can be segmented into their basic shapes. However, this only reduces the number of characters to a lesser number of classes to recognize. Table 1 shows the details of few Asian and African scripts in the form of the number of basic alphabet, the number of characters that arise from their combinations, and the approximate number of classes that are used in a typical character recognizer.

Hierarchical classifiers [1, 8] are found to be suitable for large class classification, as they have low space and time complexity. They are highly modular in nature, allowing use of a variety of classifiers at each node. However most existing designs do not explore this modularity, and use similar classifiers at all the nodes [7, 8]. This results in classifiers which under-perform, as the classification problems at each node are quite different from each other. An effective way to build a classifier is to evolve the hierarchy by choosing appropriate classifiers at each node in the hierarchy. The appropriateness of a component classifier can be measured in terms of speed, size and accuracy, depending on the requirements of the system.

In this paper, we present a formulation for the problem of building classifiers with hybrid design. A brief discussion on existing multiclass classification methods and a comparison with hierarchical classifiers is presented in Section 2. A classifier combination design called Binary Hierarchical Combination of Decision Directed Acyclic Graphs (BHCD), and an algorithm to build this classifier is presented in Section 3. Complexity and error analysis of the algorithm is presented in Section 4. The results of this algorithm are shown on many datasets in Section 5.

2. Hierarchical Classifiers for Character Recognition

Multiple classifier combination approaches for multiclass classification have received increased attention in the recent past. For an N-class classification problem,

Language	Alphabet	Characters	Classes
Devanagari (India)	57	64000	165
Telugu (India)	56	64000	432
Hangul (Korea)	67	11772	2350
Hiragana (Japan)	46	5000	1945
Amharic (Ethiopia)	34	310	225

Table 1. Few Asian and African scripts, the number of basic alphabet, possible characters and classes in a typical OCR system are shown.

one-vs-rest approaches build N classifiers, each with samples from one class as positive, and from rest of the classes as negative. These classifiers are shown inferior to one-vs-one approaches in both accuracy, as well as training time. One-vs-one methods train $N(N-1)/2$ classifiers, one for each possible pair of classes. The combined classification system usually has high performance in terms of accuracy. However, they are large in size and take longer time for classification. This has been overcome by an approach called Decision Directed Acyclic Graph (DDAG) [9], which connects the pairwise classifiers as a directed acyclic graph (DAG). The sample is initially presented to the root node of the graph, and depending on the decision taken, either left or right sub-DAG is selected for the next evaluation. This ends when a leaf node is reached, where a decision is made about the class of the sample. Another popular approach is to use a binary tree based classifier, where each node in the tree acts as a binary classifier between two subsets of classes. A label is assigned to the sample after the two-class decision at the leaf nodes of the classifier.

Hierarchical classifiers have been used for large-class OCRs in [1, 3, 8]. The hierarchy in these systems was determined manually based on perceptual similarity, or using simple cues like the positional information of characters (using zonal information [3]). In manual hierarchy building, a high correlation has to be maintained between the perceptual similarity and the feature representation in order to obtain hierarchies which give good performance.

Limitations of these manual or rule-based hierarchies were overcome recently, by automatically learning the class hierarchies [7, 10]. Binary Hierarchical Classifier (BHC) proposed in [4, 7] learns the hierarchy by dividing the samples available at each node into two clusters, and building a classifier for those samples at that node. DB2-SVM [10] was another approach which attempted to build a hierarchical SVM using basic clustering algorithms at each node. These approaches aim at the simplicity of the classifier, sometimes even at the cost of accuracy.

We demonstrate that use of a hierarchical combination of complex classifiers preserves both the simplicity of the system, as well as its high performance. A configurable design for a hybrid tree, which combines multiple DDAGs using a tree is described below.

3. Hybrid Hierarchical Classifier

A binary hierarchical classifier connects several two class classifiers in a hierarchy such that a sample is classified in a series of steps. The classification starts with broad and simple decisions and narrows down in stages to a precise decision at the leaf. Each node divides the set of available classes into two groups. However, all the classification tasks at the component classifiers are not equally complex. Many a time, binarization can not be done effectively for all possible subsets of classes, where a more complex multiclass classifier is required.

The hybrid classifier is a hierarchical arrangement of dissimilar classifiers, each chosen according to the complexity of the classification task. The current algorithm builds a hybrid classifier using two kinds of component classifiers, a linear binary classifier for simple classifications and a DDAG for complex multiclass groups. A DDAG is a high-performance classifier, but the number of component classifiers increase rapidly with increase in number of classes. The BHCD algorithm hence uses DDAG only on smaller subsets of classes which are difficult to classify within each other. A BHCD for a set of Devanagari characters is shown in Figure 1. The ovals are the binary classifiers between groups of classes, which are easy to classify. DDAGs are used to perform the within class classification of these groups, and are shown as triangles in the Figure 1.

3.1. BHCD Algorithm

A recursive algorithm is proposed to learn the hierarchical classifier. At each step of creating a node in the hierarchy, a selection is made between a binary classifier and a DDAG. For the binary classifier, a clustering algorithm is used to split the dataset into two subsets. The difficulty of classification of this binary classification is estimated. Of these two, the classifier with greater expected accuracy is chosen. To select the appropriate classifier, a "goodness" measure resembling its error estimate is needed. Section 3.2 describes the measure used in the algorithm.

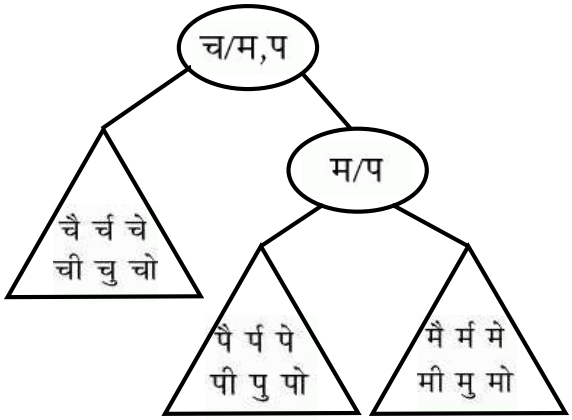

Figure 1. The BHCD classifier. The ovals are the binary classifiers for broad classification into groups of classes. The triangles are the DDAGS built to classify high-resemblance classes.

A DDAG is a multiple classifier method, which performs well due to the redundancy provided by the pairwise classifiers. Error estimates are always low for a DDAG because of the simplicity of pairwise classifications when compared to arbitrary binary classifiers. If a DDAG is always preferred, the resulting classifier suffers from large size and classification time. To discourage the selection of DDAG always, a parameter λ is used to weigh the advantage obtained while selecting the component classifiers at each node. The user provides the parameter λ to specify the relative importance of accuracy to the size and classification time of the hierarchical classifier.

Let f denote the the hybrid classifier with K nodes, and $\theta_i, i = 1 \cdots K$ be the parameters of the classifier at the ith node. At each node i, $\Omega^{(i)}$ denotes the set of labels of the classes that reach that node, Let $\Omega_l^{(i)}, \Omega_r^{(i)}$ represent the set of classes that correspond to the left subtree and right subtree respectively, $\mathcal{X}^{(i)}$ represent the samples reaching the node i and the operator $\oplus$ be used to denote the addition of a set of parameters at the current node to the existing tree. The functions *trainBinary* and *trainDDAG* are assumed to be the training algorithms for binary classification and a DDAG respectively. Using this notation, Algorithm 1 describes the procedure formally.

Algorithm 1 BHCD($\mathcal{X}, \lambda$)

$f = \phi$
$(\Omega_l^{(i)}, \Omega_r^{(i)}) = \text{Cluster}(\mathcal{X}_i^{(i)}, 2)$; // Make two clusters
$\text{L}_{binary} = ComputeClassifiabilityEstimate(\mathcal{X}_i^{(i)}, l, r)$
$\text{L}_{ddag} = ComputeClassifiabilityEstimate(\mathcal{X}_i^{(i)}, \Omega_i)$
if $(\text{L}_{ddag} < \lambda \text{L}_{binary})$ **then**
  $f = f \oplus \theta_i$
  $\theta_i = \text{trainBinary}(\mathcal{X}_l^{(i)}, \mathcal{X}_r^{(i)})$
  if $(n(\Omega^{(i)}) > 2)$ **then**
    BHCD($\mathcal{X}_l, \lambda$)
    BHCD($\mathcal{X}_r, \lambda$)
  end if
else
  $\theta_i = \text{trainDDAG}(\mathcal{X}^{(i)})$
  $f = f \oplus \theta_i$
end if

3.2. Classifier Evaluation

A key step in the algorithm is to compute the "goodness" of different classifiers at each step of tree building, and select the suitable classifier. We use an adapted version of the classifiability measure proposed in [5] for computing the possible error estimate on the datasets. The measure is closely related to the Bayesian error, and is easily computable. To start with, define an appropriate neighborhood size r. For each pattern $x^{(i)}$ which belongs to class $\omega_l \in \Omega$, obtain a co-occurrence matrix $W(x^{(i)})$ of size $N \times N$, whose element $w(x_{jk}^{(i)})$ is defined as $w(x_{jk}^{(i)}) \equiv \sum_{m=1}^{N_{\omega_k}} f(x^{(i)}, x^{(m)})$, where $x^{(m)}$ is a sample of class ω_k. $f(.)$ is 1 if $||x^{(l)} - x^{(m)}|| \leq r$ and $j = l$, 0 otherwise.

Compute the overall coocurrence matrix [5] as the sum of coocurrence matrices of all the samples, i.e $A = \sum_{m=1}^{M} W(x^{(m)})$. Normalize the elements a_{ij} of A such that sum of all the elements is 1. Then the classifiability L can be computed as,

$$L = \sum_{i=1}^{c} a_{ii} - \sum_{j \neq k}^{c} a_{jk}.$$

However, this measure computes the classifiability of all the classes together. The classifiability of datasets is different when pairwise subsets are considered. Here, we adapt this measure to provide an error estimate of the pairwise classifier, by taking average of pairwise classifiabilties. Let L_{lm} be defined as the classifiability between classes ω_l, ω_m. The pairwise classifiability L_{pw} is defined as

$$L_{pw} = \sum_{l,m \in \Omega^{(i)}, l \neq m} L_{lm}.$$

If $L_{ddag} - \lambda L_{binary}$ is negative, the binary classifier is used at the node. Otherwise, a DDAG is built between all the classes at that node.

4. Analysis

Generalization ability and computational complexity are important parameters to measure the effectiveness of a classifier. An analysis of the BHCD algorithm from these perspectives is presented in this section.

4.1. Generalization

For a classifier to perform well on unseen samples, the learning algorithm has to reduce the generalization error of the classifier. Many of the Asian and African scripts are characterized by groups of classes spread around in the feature space. One group of classes is distinctly separate from the others, but the classes within a group resemble each other and are difficult to classify. If a simple classifier is used for all the classes together, the classifier under-performs for the within group classification. If a very complex classifier is used, it performs well within the groups, but over-fits and takes large space and classification time. Since, BHCD algorithm evolves the hierarchy by selecting the classifier of required complexity at each level of the hierarchy, it generalizes well.

The path taken by the sample from the root node to the leaf node of a hierarchy while being classified is called the *evaluation path*. In [2], it was shown that the generalization error of a tree classifier depends on the length of the evaluation path T and the margins of nodes on the path. According to [10], the best case performance of a tree based classifier where $T = 1$, is better than a DDAG, where $T = N - 1$ and both classifiers are equivalent in the worst case. When there are large number of classes, a BHC has a lower generalization error bound than a DDAG, owing to the difference in the lengths of evaluation paths [10]. At the deeper nodes of a hierarchical classifier, the difference between the lengths of evaluation paths between a DDAG and a BHC is less as there are fewer number of classes. However, at this stage, the pairwise margins obtained in a DDAG are higher than the margins obtained by an arbitrary binary classifier in BHC. This shows that the BHCD algorithm selects classifiers to reduce the overall generalization error of the hierarchical classifier.

4.2. Complexity

Table 2 presents the orders of space and testing time complexity in classifiers for various popular classifier combination architectures. The testing time complexity, expressed in terms of number of classifiers evaluated for each decision to be made is denoted using C. As an example, consider an OCR for an example Indian language say, Telugu. It has approximately 430 classes to be recognized. Using a one-vs-one training with a DDAG or a majority vote combiner, and stores around 92,235 classifier parameters, which is prohibitive. Using a majority vote based approach here would result in evaluating all the classifiers atleast once per sample. Whereas, in a hierarchical classifier, there are only 429 classifiers built, of which on average 7 classifiers get evaluated. This gain in space and time are immense, even at the cost of a little reduced accuracy, depending on the requirements of the application.

Architecture	Space	C_{best}	C_{avg}	C_{worst}
One vs Rest	$O(N)$	$O(N)$	$O(N)$	$O(N)$
One vs One	$O(N^2)$	$O(N^2)$	$O(N^2)$	$O(N^2)$
DDAG	$O(N^2)$	$O(N)$	$O(N)$	$O(N)$
Hierarchical	$O(N)$	$O(1)$	$O(N)$	$O(log(N))$

Table 2. Table comparing the order of space and time in best (T_{best}), average(T_{avg}), and worst (T_{worst}) cases in terms of number of classes for different architectures

5. Results and Discussion

Experiments are conducted on character recognition problems. Four datasets were used in the experiments, two Indian language scripts Tamil, Malayalam, an African script – Amharic and the Letter dataset from the UCI machine learning repository. Tamil dataset has 120 classes, Malayalam has 116 classes, and Amharic dataset has 225 classes. Each of them have 100 samples per class. Letter dataset has 26 classes, and 20000 samples. In each experiment we used 60% of the data for training and rest for testing. We used a linear SVM as the classifier at all nodes of the hybrid classifier built including the pairwise classifiers in the DDAG. Results show that BHCD performs better than BHC always, and sometimes better than the DDAG, taking significantly lesser space than a DDAG, on par with a BHC.

UCI Letter Dataset: UCI Letter dataset [6] has 26 classes. In this case, a DDAG requires 325 classifiers to be built and a BHC requires 25 classifiers. The number of classifier evaluations made when using a DDAG is 25, and using a BHC is around 5. Using linear SVMs at each node, the accuracy obtained with a DDAG is 75.16% and with a BHC is 71.20%. The BHCD algorithm is found to improve the accuracy of the classifier over BHC, and reduces the storage space by 3.1 times and classification time by 4.1 times compared to a DDAG.

	$\lambda = High$ (BHC)			$\lambda = Medium$ (Hybrid)			$\lambda = Low$ (DDAG)		
	Accuracy	Storage	T_{eval}	Accuracy	Storage	T_{eval}	Accuracy	Storage	T_{eval}
Tamil	91.50	119	8	93.6	1029	26	93.4	7140	119
Malayalam	98.2	115	8	98.62	990	24	98.60	6555	115
Amharic	84.72	224	9	86.82	2250	31	87.20	92235	224
Letter	71.20	25	6	73.23	105	8	75.16	325	25

Table 3. Accuracies of classifiers built with varying λ. The size of the classifier (storage) is shown as the total number of nodes in the classifier. The evaluation time of the classifier (T_{eval}), measured as average depth of the tree.

Malayalam Character Recognition: Malayalam, a South Indian language is considered for this experiment. The number of classifiers required for a DDAG is 6670, and makes 115 classifier evaluations to make a decision. The accuracy obtained using a DDAG is 98.60%. The number of nodes in a BHC is 115, and around 7 classifier evaluations are necessary to classify a given sample. The accuracy obtained by a BHC is 98.20%. BHCD yields an accuracy of 98.62% which is better than both the approaches and reduces the storage space by 6.5 times, and classification time by 4.8 times with respect to a DDAG.

Tuning the Classifier: Experiments on large class datasets are conducted to show the effect of the control parameter λ on the design of the classifier. Hybrid classifiers were generated with varying values of λ and the accuracies and size values of the classifiers are computed. We grouped the λ values into three groups – low, medium and high. For each dataset, the accuracy, storage and the evaluation time are presented in Table 3. When λ is high, the resultant classifier is a binary hierarchical classifier. When λ is low, there is no constraint on the size, and hence a classifier DDAG. When λ is medium, a classifier of size larger than the tree, but much smaller than a DDAG is obtained, with performance higher than the tree in all the cases, and also higher than DDAG in some cases. A large increase in the classifier size as well as the depth of the tree in the classifier was observed as the value of λ decreases. In most of the cases, the drop in the accuracy observed was a minimal compromise for the large reduction obtained in size and testing time.

6. Conclusions

An algorithm for building hybrid classifiers combining the advantages of two algorithms BHC and DDAG is presented. It is shown that using a hierarchical combination of complex classifiers, suits the problem of large class character recognition. This results in classifiers with better generalization and with less space and time complexities.

References

[1] H. Baird and C. Mallows. Bounded-error preclassification trees. In D. Dori and A. Bruckstein, editors, *Shape, Structure and Pattern Recognition*, pages 100–110. World Scientific Publishing Co., 1995.

[2] K. P. Bennett, N. Cristianini, J. Shawe-Taylor, and D. Wu. Enlarging the margins in perceptron decision trees. *Machine Learning*, 41(3):295–313, 2000.

[3] B. B. Chaudhuri and U. Pal. An OCR system to read two Indian language scripts: Bangla and Devanagari (Hindi). *Proc. ICDAR*, pages 1011–1015, 1997.

[4] Y. Chen, M. M. Crawford, and J. Ghosh. Integrating support vector machines in a hierarchical output space decomposition framework. In *IEEE International Geoscience and Remote Sensing Symposium, Alaska AK*, volume 2, pages 949 – 952, September 2004.

[5] M. Dong and R. Kothari. Feature subset selection using a new definition of classifiability. *Pattern Recognition Letters*, 24(9-10):1215–1225, 2003.

[6] S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases. 1998, (*http://www.ics.uci.edu/∼mlearn/mlrepository.html*).

[7] S. Kumar, J. Ghosh, and M. Crawford. A hierarchical multiclassifier system for hyperspectral data analysis. *Lecture Notes in Computer Science*, 1857:270–278, 2000.

[8] S. S. Marwah, S. K. Mullick, and R. M. K. Sinha. Recognition of Devanagari characters using a hierarchial binary decision tree classifier. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 414–420, October 1994.

[9] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multi-class classification. In *Advances in NIPS-12*, pages 547–553, 2000.

[10] V. Vural and J. G. Dy. A hierarchical method for multi-class support vector machines. In *Proceedings of the 21st ICML*, pages 831–838, July 2004.