Design of Hierarchical Classifier with Hybrid Architectures

M.N.S.S.K. Pavan Kumar and C.V. Jawahar

Centre for Visual Information Technology,
International Institute of Information Technology,
Hyderabad, INDIA

Abstract. Performance of hierarchical classifiers depends on two aspects – the performance of the individual classifiers, and the design of the architecture. In this paper, we present a scheme for designing hybrid hierarchical classifiers under user specified constraints on time and space.

1 Introduction

Classifiers for multiclass problems, built using combinations of two-class solutions are found to be superior to direct multiclass classifiers in terms of training time and performance [1]. Classifier combination can efficiently be done using hierarchical techniques. Hierarchical classifiers are usually built of simple classifiers. Only a subset of these classifiers evaluate a sample, resulting in low classification time. Hierarchical classifiers are being preferred to the two-stage approaches as demonstrated in BHC-SVM [2]. The topology using which the classifiers are combined is generally referred to as the architecture of the combined classifier. Most of the hierarchical algorithms follow a binary tree architecture. Algorithms for design of hierarchical classifiers focus on evolving the tree by learning the classifiers at each node.

The performance of the hierarchical classifiers depends on the performance of the individual classifiers and the overall combination architecture. In this paper we propose a scheme that incorporates both these factors in designing a hierarchical architecture. Generalization at individual classifiers is obtained with the help of Support Vector Machines(SVM) [1]. The proposed scheme allows selection of a suitable architecture at each node of the tree. A mechanism to specify the suitability of an architecture, as per the requirements like high-accuracy or low-classification time, is provided.

1.1 Background

There are two architectures popular for hierarchical classifiers — Binary Hierarchical Classifier (BHC) [2] and a Decision Directed Acyclic Graph (DDAG) [1]. A BHC is fast but of poorer in accuracy compared to a DDAG. For large class problems, a DDAG requires a prohibitively large number of classifiers.

S.K. Pal et al. (Eds.): PReMI 2005, LNCS 3776, pp. 276–279, 2005.

A BHC recursively partitions the dataset into two at each stage of the hierarchy building, and thereby learns a complete classifier. This results in a classification tree with a binary tree architecture. A sample is presented to the root, and based on the decision at the root node, left or right subtree is selected. This continues till the sample reaches a leaf node, where it gets its label assigned.

A DDAG is an efficient way to combine the pairwise classifiers in the form of a rooted binary directed acyclic graph. Although a DDAG uses $\binom{N}{2}$ classifiers, the number of classifier evaluations is of $O(N)$ for each sample. In [3], an algorithm for improving the performance of a DDAG has been discussed. In a specific case, where misclassification probabilities of all pairwise classifiers are equal, a DDAG design algorithm has been proposed. The algorithm results in an optimal DDAG. The design algorithm attempts to maximize a global objective function J, which is defined as $J = \sum_{i=1}^{N} P_i(1 - Q_i)$ where Q_i is misclassification probability of class ω_i by the DDAG. A greedy algorithm to maximize the global objective function is shown in Table 1.

Table 1. Algorithms for improving the DDAG([3]), and Building the BHC Tree([2])

Algorithm improveDDAG(Ω)
Intialize S = ();
while Ω not empty **do**
  select ω_i, ω_j with highest priors
  $S = (\omega_i, S, \omega_j)$
  $\Omega = \Omega - \omega_i, \omega_j$
end while
output S;

Algorithm BuildTree(Ω)
(Ω_l,Ω_r) = partition(Ω);
if $|\Omega_l| > 1$ **then**
  BuildTree(Ω_l);
end if
if $|\Omega_r| > 1$ **then**
  BuildTree(Ω_r);
end if

In a DDAG the classifier components are fixed and hence the design algorithm involves only arrangement of nodes. In a BHC, the node design decides the architecture of the tree. A BHC partitions the available set of classes into two subsets at each step. This is continued recursively until only a single class remains at each partition. The partitioning step is crucial, and decides the performance of the classifier. The algorithm is described in Table 1.

2 Hybrid Algorithm

In a BHC, at each step of tree building, the classification problems are different, which means a single classifier may not suit all the problems. The proposed hybrid architecture employs different classifiers for building the hybrid tree. We compare a binary partition and a DDAG and at each step, and use the better one among them. The algorithm therefore measures the classifiability of the datasets for these two architectures, and prefers one of them depending on the parameter λ.

Estimating Classifiability. A key step in the algorithm is to compute the advantage obtained by using various architectures at each step of tree building. Let L denote a measure which can estimate the classifiability of a dataset.

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

$f = \phi$
$(\Omega_l^{(i)}, \Omega_r^{(i)})$ = Cluster($\mathcal{X}_i^{(i)}, 2$); // Make two clusters
Ł$_{binary} = ComputeClassifiabilityEstimate(\mathcal{X}_i^{(i)}, l, r)$
Ł$_{ddag} = ComputeClassifiabilityEstimate(\mathcal{X}_i^{(i)}, \Omega_i)$
if (Ł$_{ddag} <$ Ł$_{binary}$) **then**
  $f = f \oplus \theta_i$
  θ_i = 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
  θ_i = trainDDAG($\mathcal{X}^{(i)}$)
  $f = f \oplus \theta_i$
end if

One such measure is proposed in [4]. We extend this measure for applicability to pairwise classifiers, L_{pw}, by taking the average pairwise classifiabilties of individual datasets. This is used in obtaining the estimate for a DDAG. Let L_{lm} denote the classifiability computed for the classes ω_l, ω_m, then $L_{pw} = \frac{2}{N(N-1)} \sum_{l,m \in \Omega^{(i)}, l \neq m} L_{lm}$.

A recursive procedure is used to build the BHCD. At each node i, $\Omega^{(i)}$ denotes the set of labels of the classes that reach that node. The root node of the hierarchy handles all the classes $\Omega^{(0)} = \Omega$. The dataset is split into two subsets using any data clustering algorithm. We use K-Means clustering, with $K = 2$. The classifiability of the data with this split(partition) is L_{split}. The pairwise classifiability, L_{pw} is computed for all the classes at the current node. Using the user specified parameter λ, if $L_{pw} < \lambda L_{split}$, then the DDAG is chosen as architecture for that set of classes. Otherwise, of classes are split into $\Omega_l^{(i)}, \Omega_r^{(i)}$, the left and right subsets of the classes. The algorithm recursively repeats these steps until each node is left with only two classes to train at the leaf level.

When $\lambda = 1$, the algorithm is equivalent to searching for the best design at each node. In this case, the resulting hybrid tree is at least as good as BHC or a DDAG. When $\lambda \neq 1$, then user preferences are taken into account. Considering accuracy, the algorithm is better than a BHC, and could be better than than both BHC and DDAG. More importantly the constraints on size are taken into account in the design.

3 Results

We demonstrate the performance of the BHCD algorithm on the Letter dataset of UCI [5] repository, and an Indian Language Character dataset. The letter dataset contains 26 classes, and 20000 samples in total. In the experiments, 60% of the dataset is used for training and the remaining for testing. At each

node of the classifier, a linear SVM is used. When λ is set to a low value, the algorithm results in a DDAG classifier. The DDAG has 325 classifiers, and give a performance of about 75% on the test dataset. For a high value of λ, a pure BHC with 24 nodes is output with an accuracy of 71%. For a medium value of λ, a classifier with 105 nodes is obtained with an accuracy of 73.23%.

We tested the algorithm on Malayalam character recognition dataset with 116 classes and 100 samples per class. 100 samples per class. In this experiment, 60% of the training set is used in training, and the rest for testing. Different classifiers were built on this dataset with varying λ. For a low λ, a pure DDAG is obtained, and has 6555 nodes in total. The length of the evaluation path of the DDAG is 115 nodes. In each case, if each node has parameters which require 3.2KB of storage space, the size of the whole classifier is 21Mb. The accuracy of this classifier is found to be 98.6%. Using a pure BHC tree, an accuracy of 98.2% is obtained. The number of classifiers required were 115, and the storage space is 368KB. Using a medium value for λ, an accuracy of 98.6 is obtained, for a classifier with 990 nodes. This classifier took 3.1MB of storage space. The average length of the evaluation path is 24 nodes, as compared to 8 for a BHC and 115 for a DDAG.

The results show that it is possible to obtain different performances by varying the value of λ. It is upto the user to pick the classifier with the required performance, with respect to size, accuracy and evaluation time. It is shown that a compact classifier without much loss in the performance can be built using this algorithm.

4 Conclusions

We describe a flexible scheme for building hybrid hierarchical classifiers, which can give the best possible classifier meeting the constraints on parameters like classification time and size of the classifier, by capturing and combining different benefits offered by different architectures.

References

1. Platt, J.C., Cristianini, N., Shawe-Taylor, J.: Large margin DAGs for multi-class classification. In: Advances in NIPS-12. (2000) 547–553
2. Kumar, S., Ghosh, J., Crawford, M.: A hierarchical multiclassifier system for hyperspectral data analysis. Lecture Notes in Computer Science **1857** (2000) 270–278
3. M N S S K Pavan Kumar, Jawahar, C.V.: On improving design of multiclass clasifiers. In: Proceedings of the 5th International Conference on Advances in Pattern Recognition. (2003) 109–112
4. Dong, M., Li, Y., Kothari, R.: Theoretical results on a measure of classification complexity. In: Proceedings of 5th International Conference on Advances in Pattern Recognition. (2003) 85–88
5. Hettich, S., Blake, C., Merz, C. In: UCI repository of machine learning databases, *http://www.ics.uci.edu/∼mlearn/MLRepository.html*. (1998)