Using Hierarchical Classification to Exploit Context in Pattern Classification for Information Fusion

A. Bailey and C. J. Harris
ISIS Research Group, Department of Electronics and Computer Science
University of Southampton,
Southampton, SO17 1BJ, UK.
e-mail: ab96r@ecs.soton.ac.uk

Abstract *In data fusion applications it is important that only the minimum set of relevant features are combined at any one stage in the fusion process. A hierarchical classification methodology is described which handles features at different levels of abstraction to produce a more robust and interpretable classifier. This is achieved by dividing the classes into contextual subgroups, which are further divided to produce a tree structure defining relationships between classes.*

A novel approach is proposed for the class structure design which is formulated as a constrained search in the structure space. This can be performed via a forward search algorithm driven by a cost function dependent on the performance of the class structure and constraints on the required solution.

Keywords: Statistical Pattern Classification, Context, Hierarchical

1 Introduction

In data fusion applications it is important that only the minimum set of relevant features are combined at any one stage in the fusion process. A hierarchical classification methodology is described which handles features at different levels of abstraction to produce a more robust and interpretable classifier. Classification is important in many areas of data fusion such as automatic target recognition, situation assessment and ballistic missile defence [1].

In pattern classification tasks of many classes in a high-dimensional input space, e.g. document classification, remote sensing and automatic target recognition, traditional flat classifiers tend to suffer from the curse of dimensionality [2]. Even after feature selection, a large number of inputs is still needed to discriminate the large number of potential classes. A hierarchical classifier overcomes this problem by dividing the classes into contextual subgroups, which are then further divided to produce a tree structure defining relationships between classes.

It can be shown that by using arbitrary probabilistic classifiers to discriminate between subgroups at each node of the tree, posterior probabilities can be output equivalent to a standard flat classifier but the feature space for each classifier is significantly reduced [3]. In this case each classifier node can be more robustly estimated as they perform simpler discrimination tasks in lower dimensional spaces.

Previous methods have used various clustering schemes to build the most appropriate class structure in terms of interpretation and accuracy of classification [4]. Once the class structure has been elicited, standard feature selection and parameter estimation techniques can be used to specify each classifier node.

Rose *et al.* [5] present a constrained hierarchical clustering method using simulated annealing. A small number of coarse clusters are identified at high temperature. As the temperature is lowered coarse clusters split into more detailed clusters. This leads to a course to fine hierarchy of clusters extracted during the annealing schedule. Kim and Landgrebe [2] define a hierarchical classifier and a hybrid hierarchy design process combining both agglomerative and divisive clustering techniques.

A novel approach to identifying the class hierarchy by discrete optimization search is proposed. Constraints on the structure of the hierarchies in the search space are defined.

2 Context

Dealing with situations in their proper context is an everyday notion that allows people to select the right information in completing complex tasks. Researchers have previously identified the use of context as an advantage in problem solving.

Toussaint [7] suggests that to solve a problem instead of increasing the depth of analysis, one should widen the field of context in which the problem is viewed.

Whereas Antony [8] refers to context in tactical data fusion applications as including 'current friendly force disposition, existing weather conditions, natural domain features (terrain/elevation, surface materials, vegetation, rivers, drainage regions), and cultural features (roads, airfields, mobility barriers)'.

Context may be used in different ways to aid a classifier system. The above two references refer to widening the field of context by searching for extra information that may aid discrimination between classes. In terms of classification, this can be thought of as an intelligent method to generate more features for a classifier, given knowledge of the problem domain.

An alternative standpoint is that one can use the context of a situation to refine and abstract only the relevant information in the context of the classification problem at hand. This can be thought of as decomposing a problem and focusing attention only on those features of importance.

This paper is primarily concerned with the second view on context and a classification scheme is proposed that will fully exploit this notionl.

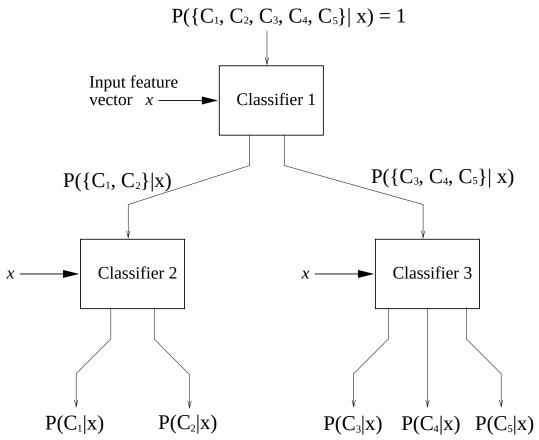

Figure 1: Hierarchical classifier structure

3 Hierarchical classifier

The framework described in this section is based on Schurmann [3]. Let Ω^{all} be the set of all the possible classes. This can be split, initially arbitrarily, into a number of subsets. The top level classifier provides posterior probabilities for each subset given the input vector, x. The second level classifiers then work on the individual subsets, breaking them down into smaller subsets, and incorporating the posteriors from the level above. This is performed in a tree-structured hierarchy until a posterior has been calculated for each individual class. This is shown in figure 1.

To formalise this hierarchical process consider a single classifier node as depicted in figure 2. The classifier need only be concerned with the subset of classes that it has been designated from the level above, this is called the input set, Ω^{in}. The input set is split into S output subsets, $\Omega_1^{\mathrm{out}}, \Omega_2^{\mathrm{out}}, ..., \Omega_S^{\mathrm{out}}$. Each output subset is unique and all elements in the input subset are assigned to only one output subset. Formally,

$$\Omega_i^{\mathrm{out}} \subset \Omega^{\mathrm{in}} \;\forall i \text{ and } \Omega_{\mathrm{i}}^{\mathrm{out}} \cap \Omega_{\mathrm{j}}^{\mathrm{out}} = \emptyset \;\forall \mathrm{i} \neq \mathrm{j}, \quad (1)$$

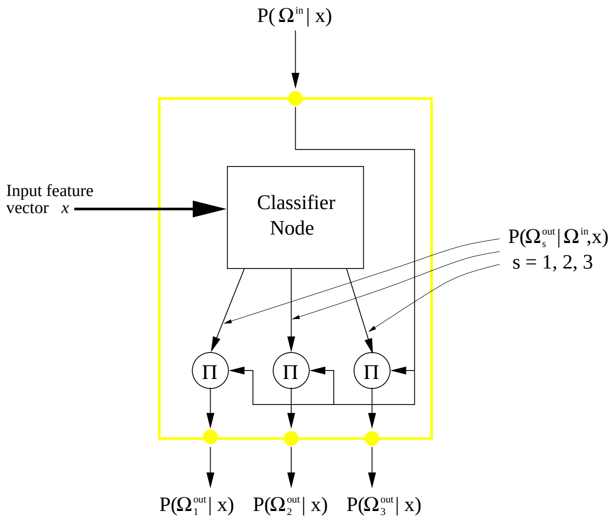

Figure 2: Individual classifier node

$$\bigcup_{s=1}^{S} \Omega_s^{\text{out}} = \Omega^{\text{in}}. \tag{2}$$

The goal of the complete classification process is to calculate posterior probabilities for each class given the input vector. However, each classifier in the hierarchical structure can only generate posteriors that are valid given the input set Ω^{in}:

$$P(\Omega_s^{\text{out}}|\Omega^{\text{in}}, x) \;\; s = 1, ..., S. \tag{3}$$

Here we can use the law of conditional probability to determine the relation between the input and output sets of the individual classifiers:

$$P(\Omega_s^{\text{out}}|\Omega^{\text{in}}, x) = \frac{P(\Omega_s^{\text{out}}, \Omega^{\text{in}}, x)}{P(\Omega^{\text{in}}, x)} \tag{4}$$

$$\begin{aligned} P(\Omega_s^{\text{out}}|\Omega^{\text{in}}, x) &= \frac{P(\Omega_s^{\text{out}}, x)}{P(\Omega^{\text{in}}, x)} \\ &= \frac{P(\Omega_s^{\text{out}}|x)p(x)}{P(\Omega^{\text{in}}|x)p(x)} \\ &= \frac{P(\Omega_s^{\text{out}}|x)}{P(\Omega^{\text{in}}|x)} \end{aligned} \tag{5}$$

Using this relationship it is easy to demonstrate that to calculate the required class posteriors, all is required is to multiply the posteriors given at each level while following the route down the tree to the required class.

3.1 Comments

No assumptions have been made on the type of classifiers used, save that they return a posterior probability for each output. No assumptions have been made at all concerning the structure of the class subsets and the elements of the input vector, x. Schurmann [3] states that, given optimal discrimination at each node, optimal discrimination will be achieved for each final class regardless of the choice of tree structure. Evidently optimal discrimination is not always possible and a sensible choice of the class subsets that defines the tree structure will maximise discrimination as each node and therefore maximise global discrimination.

3.2 Difference from Decision Tree Classifiers

The princicple differences between hierarchical classifiers and decision tree classifier (such as Quinlan's C4.5 [9]) are the use of soft decision making at each node, allowing posteriors to be output for all classes, the use of arbitrary classifiers at each node, and the manipulation of the set of class hierarchies.

Although refence is made to the data in the input space via the distance metrics in the objective function, the space that is being searched in the construction of the hierarchy is not the set of possible splits in input space, but the combination of class subsets in the set of all possible class hierarchies.

A given class is always considered complete with all points belonging to that class irrespective of any class overlap in input space. Under the assumption of of normally distributed classes the collection of points in al class may be represented by their mean and covariance.

3.3 Existing methods for class hierarchy design

Standard methods exist for generating class hierarchies. These tend to be motivated from hierarchical clustering techniques which are unsupervised methods. Two distinct approaches are agglomerative and divisive methods, depending on whether the hierarchy is grown from the top down, or from the bottom up.

3.3.1 Agglomerative clustering

When designing a class structure from the bottom up, the goal is to group the classes together according to a similarity measure. The two most similiar classes are joined to form an intermediate class. Each new intermediate class replaces its members and the process is repeated until the last two classes are joined.

The similiary measures used can be chosen depending on the assumptions made on the class distributions. Suitable distance measures are the Mahalanobis distance, or the Battacharyya distance, both of which assume normally distributed classes with arbitrary covariance matrices.

3.3.2 Divisive clustering

The top-down approach to structure design works in an opposite fashion by considering the dataset as a whole initially and then splitting it into smaller groups according to a performance criteria. Each group can then be considered for splitting itself and a hierarchy is built in a recursive manner.

The technique of splitting data into homogeneous groups is well researched in the literature on clustering.

4 Contextual features

In most pattern recognition problems the dataset is defined by a set of points in an n-dimensional space. Each point represents an entity to be classified and each dimension represents a measureable feature of that object. If a feature is not present for all points in the dataset then methods are available to fill in these points with estimates of the most likely missing values [10]. Most classification algorithms require that all features are present for every point before training can take place. However, for the set of decision tree classifiers, including C4.5-type algorithms and the hierarchical classifier described here, certain features may not have to be present for all classes. A feature may be used to discriminate between only a subset of the available classes, and in this case its value for points outside that subset are not important, and even need not be defined.

In fact, the presence of such features, that may occur naturally in many real-world datasets such as in target recognition or medical diagnosis applications, can aid the design process of a hierarchical classifier. Regardless of any discriminative information that may be contained within such features their presence or absence can suggest a suitable class hierarchy. For example, studying the dataset in figure 3, one can determine the class hierarchy given alongside.

Although it may be possible to generate a class hierarchy simply from the pattern of such contextual features, it is again a difficult search problem. It is much easier to verify that a given class hierarchy matches a feature pattern by checking for consistency of contextual features across class subsets. A feature may only be entirely present or absent for a given class subset. If this constraint is broken then the hierarchy can be assumed to be an unlikely solution for that dataset. This constraint can easily be incorporated into the search procedure proposed below.

5 Solution requirements

It is proposed that the class hierarchy can be found by a search in solution space (i.e. the set of all possible class hierarchies). The search strategy must allow for four specific requirements on the eventual solution sought. That is:

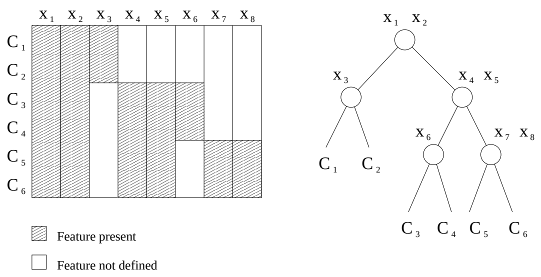

Figure 3: Dataset with contextual features and matching hierarchy

- Accuracy - Evidently the final classifier should produce accurate classification results.
- Degrees of freedom - The number of parameters for the final model should be controlled to prevent overfitting and the curse of dimensionality.
- Smoothness - The models must adhere to predefined constraints on the structure to prevent unbalanced trees.
- Prior Knowledge - If a prior model is given, the information represented in that prior must be respected.

5.1 Accuracy

To measure the accuracy of a classifier the most direct measure is the misclassification rate, as this truly reflects the desired performance. However this can be costly to compute, as it involves the complete training of each classifier node in the hierarchy. A more efficient and reasonable approximation is a sum of Mahalanobis distances between class subsets:

$$J = \sum_{n \in S} (\mu_{l_n} - \mu_{r_n})^T \Sigma^{-1} (\mu_{l_n} - \mu_{r_n}) \qquad (6)$$

where S is the set of all nodes in the hierarchy, μ_{l_n} and μ_{r_n} are the means of the points in the left and right class subsets of node n respectively and Σ is the covariance matrix for the points in both class subsets.

This distance measure reflects the separation between the left and right class subgroups chosen at each level. This value is to be minimised, since the classes at the leaves of the tree are required to be close together so that they may sensibly be contained by the class superset at the parent node. At levels higher in the tree the requirement of proximity becomes less important and this is satisfied naturally since there will always be more nodes at each descending level of the tree. The summation therefore gives greater significance to the leaf classes due to their number.

5.2 Degrees of freedom

The dangers of overfitting and the curse of dimensionality have been known for some time now [10]. Controlling the effective number of parameters in a model is important, especially in terms of small datasets where the parameters need to be specified by a few data points in a manner that is statistically significant.

The number of nodes in the class hierarchy is fixed, due to the binary nature of the tree and the requirement that all classes should be present. The complexity of the model can be controlled by using the optimum minimal feature set at each classifier node.

However, distance measures such as in equation (6) can only be used to compare features sets of the same dimensionality. This presents a problem since performing feature selection for each state to be evaluated in solution space will result in models of different dimensionality across nodes, rendering the summation meaningless. Again a solution to this would be to use the final classification accuracy as a performance measure but this is too computationally expensive. As an initial investigation, the search is performed on models containing all features, allowing equation (6) to be used.

Feature selection and training can be performed on the model given by the final class hierarchy.

5.3 Smoothness

For a class hierarchy to be meaningful and interpretable, it is likely that is should be well balanced (i.e. no particular branch should be significantly deeper than the others). A constraint on the maximum allowable depth of a single branch can be imposed on the required solution.

5.4 Prior Knowledge

The data is not usually the only available source of information on the desired model. Prior knowledge should be used wherever possible to aid the construction of a good model. If a class hierarchy is suggested through domain knowledge then it may be desirable to find a solution that does not differ significantly from the given hierarchy.

Tree comparison algorithms are available, but can be computationally expensive. In some cases the distance between two trees is described by the number of transformations required to transform one tree to the other. This distance is effectively the search depth if the search is initialised with a prior hierarchy. This can be easily and efficiently incorporated as a constraint on the search.

6 Hierarchy design by combinatorial optimization

The search strategies presented in this section strive to find a class hierarchy that gives a mimimum value of the objective function. It is expected that there will be many local minima due to the discrete nature of the search space (the set of possible class hierarchies). There are several combinatorial optimisation techniques that have become standard in the literature, some of which are designed to overcome such local minima. As ever, there is no one algorithm which can guarantee the global minimum is found, but techniques can be used that increase the chance of finding a good result.

The discrete optimisation problem can be defined as finding a solution from the set of all possible states such that minimises a given objective function within the given constraints. The constraints may be incorporated as hard constraints on the states allowed by the search procedure, the search rejects any states that do not adhere to the constrainst, or they may be incorporated as soft constraints by introducing extra terms in the objective function. The technique in this paper uses hard constraints, effectively reducing the search space. Soft constraints require free parameters to be estimated as coefficients for each term in the objective function which can be costly for large search spaces. The set of all possible states (state space) may be viewed as a graph, given operators that transform one state to another. Such graphs tend to grow exponentially with the size of a problem and optimal search techniques are NP-compete, that is the solution time increases exponentially with problem size for all algorithms. However, heuristic algorithms exist that can find suboptimal solutions in polynomial time. Examples are directed depth-first search (DFS), cost-bounded DFS (IDA*), depth-first branch-and-bound (DFBB) and best-first search (BFS). Many of these algorithms may be implemented on a parallel processor architecture [11].

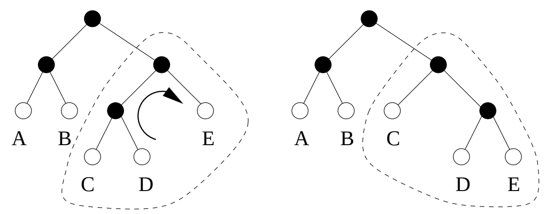

Figure 4: Example of branch shift operator

A single operator can be defined on class hierarchies to generate a search tree. From a given node, a branch of a child node may be shifted to be the sibling of the opposite child node. This is illustrated in figure 4. Any valid hierarchy may generated from any other valid hierarchy using combinations of this operator. For simplicity only binary hierarchies are used.

7 Results

Initial results have shown that anything other than a directed depth-first search without backtracking produces a search tree that is far too large to evaluate. (Simply backtracking to just the next best branch produces a search space of 7219 nodes for an 8-depth search on a 12-class problem, compared to 504 nodes for a 30-depth search without backtracking. The deeper search consistently found a better solution.)

Depth-bounded directed depth-first searches were performed on a selection of simulated datasets with the number of classes ranging from 8 to 32 and the number of features ranging from 7 to 93. The searches were initialised using the solution from an agglomerative design procedure and all but one search showed an improvement on this initial solution, the improvements becoming more significant as the number of classes increased. Figure 5 shows the increase in performance as the number of classes increase.

Figure 6 shows for a dataset of 32 classes the improvement in the objective function as the search progresses. Figures 7 and 8 show the class hierarchies generated by the agglomerative clustering procedure and the improved hierarchy given by the search. The hierarchy in Figure 8 displays a more balanced and representative tree.

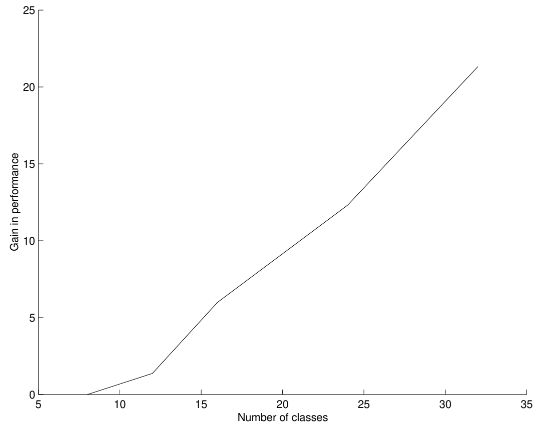

Figure 5: Gain in performance (sum of distances) against number of classes

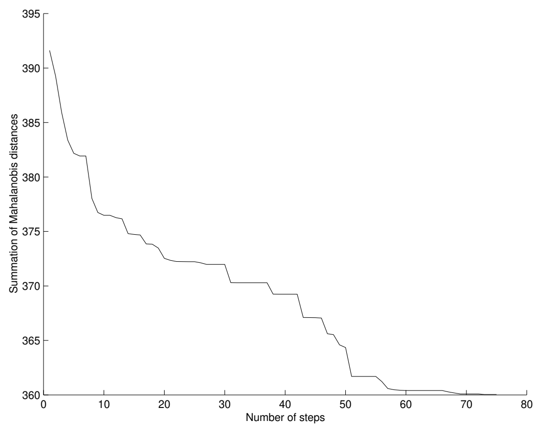

Figure 6: Search criterion against search steps

8 Conclusions

A novel approach to defining the class hierarchy for a hierarchical classifier using a search in solution space has been proposed and shown to improve upon results given by an existing method.

Contextual features have been defined and it has been shown how they may assist a search as described above by producing a constraint that can reduce the size of the search space.

9 Acknowledgements

The author would like to acknowledge IBM UK for their support of this work.

References

[1] C. J. Harris, A. Bailey, and T. J. Dodd. Multi-sensor data fusion in defence and aerospace. *The Aeronautical Journal*, 102(1015):229–244, 1998.

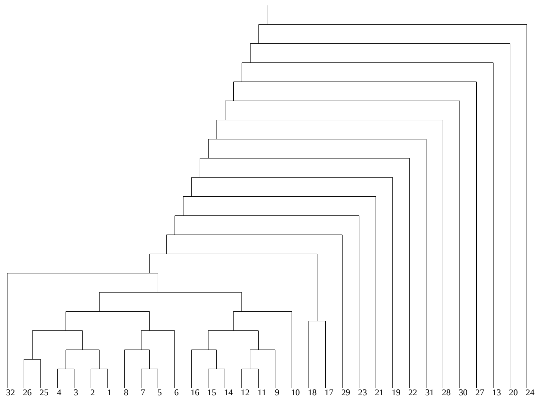

Figure 7: Hierarchy resulting from agglomerative design (depth of hierarchy = 20)

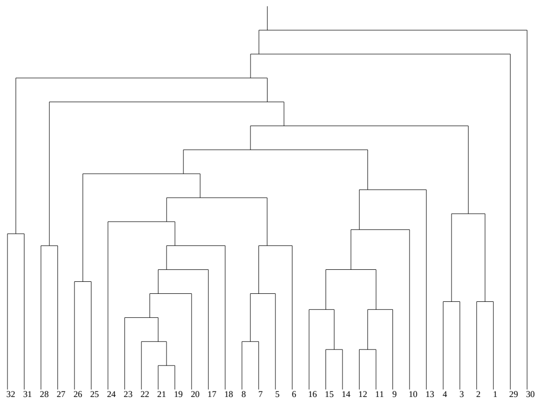

Figure 8: Hierarchy after search (depth of hierarchy = 16)

[2] B. Kim and D. A. Landgrebe. Hierarchical classifier design in high-dimensional numerous class cases. *IEEE Transactions on Geoscience and Remote Sensing*, 29(4):518–528, 1991.

[3] J. Schurmann. *Pattern Classification: A unified view of statistical and neural networks*. Wiley-Interscience, 1996.

[4] M. Sahami. Using machine learning to improve information access. Phd thesis, Stanford University, 1998.

[5] K. Rose, E. Gurewitz, and G. C. Fox. Constrained clustering as an optimization method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):785–794, 1993.

[6] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. 21(3):660–674, 1991.

[7] G. C. Toussaint. The use of context in pattern recognition. *Pattern Recongnition*, 10:189–204, 1978.

[8] R. T. Antony. Database support to data fusion automation. *Proceedings of the IEEE*, 85(1):39 – 53, 1997.

[9] J. R. Quinlan. *C4.5 : Programs for machine learning*. Morgan Kaufmann, 1993.

[10] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[11] A. Grama and V. Kumar. State of the art in parallel search techniques for discrete optimization problems. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):28–35, 1999.