Pattern Analysis & Applications (2002)5:210–220
Ownership and Copyright
© Springer-Verlag London Limited

Hierarchical Fusion of Multiple Classifiers for Hyperspectral Data Analysis[1]

Shailesh Kumar[1], Joydeep Ghosh[1] and Melba M. Crawford[2]

[1]*Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA;*
[2]*Center for Space Research, The University of Texas at Austin, Austin, TX, USA*

Abstract: Many classification problems involve high dimensional inputs and a large number of classes. Multiclassifier fusion approaches to such difficult problems typically centre around smart feature extraction, input resampling methods, or input space partitioning to exploit modular learning. In this paper, we investigate how partitioning of the *output space* (i.e. the set of class labels) can be exploited in a multiclassifier fusion framework to simplify such problems and to yield better solutions. Specifically, we introduce a hierarchical technique to recursively decompose a C-class problem into $C-1$ two-(meta) class problems. A generalised modular learning framework is used to partition a set of classes into two disjoint groups called meta-classes. The coupled problems of finding a good partition and of searching for a linear feature extractor that best discriminates the resulting two meta-classes are solved simultaneously at each stage of the recursive algorithm. This results in a binary tree whose leaf nodes represent the original C classes. The proposed hierarchical multiclassifier framework is particularly effective for difficult classification problems involving a moderately large number of classes. The proposed method is illustrated on a problem related to classification of landcover using hyperspectral data: a 12-class AVIRIS subset with 180 bands. For this problem, the classification accuracies obtained were superior to most other techniques developed for hyperspectral classification. Moreover, the class hierarchies that were automatically discovered conformed very well with human domain experts' opinions, which demonstrates the potential of using such a modular learning approach for discovering domain knowledge automatically from data.

Keywords: Binary hierarchical classifier; Fisher discriminant; Hyperspectral data; Hyperspectral feature extraction; Output space decomposition; Pattern recognition; Remote sensing

1. INTRODUCTION

Many real world classification problems are characterised by a large number of inputs and a moderately large number of class labels that can be assigned to any input. Two popular simplifications have been considered for such problems: (i) *feature extraction*, where the *input space* is projected into a smaller *feature space*, thereby addressing the curse of dimensionality issue [1,2]; and (ii) *modular learning*, where instead of using a single classifier, a number of classifiers, each focusing on a specific aspect of the problem, are developed. Several methods for feature extraction and modular learning have been proposed in the pattern recognition and computational intelligence communities [3,4].

Discrimination among different landcover types using remotely sensed data is an important application of pattern classification. Advances in sensor technology have made possible the simultaneous acquisition of hyperspectral data in more than 200 individual bands, where each spectral band covers a fixed range of wavelengths. Although hyperspectral data are becoming more widely available, computationally tractable algorithms that exploit the potential of the higher spectral resolution provided by the narrow bands are needed. In addition to the problem of high input dimensionality, there is typically a moderately large number of classes in each scene. As the number of classes increases, the signatures of individual classes typically have greater overlap. Hence, the overall classification becomes more difficult. In our previous papers [5,6], we addressed the high input dimensionality problem by extracting features based on best bases algorithms. These features were used by our pairwise classifier architecture [7,8], wherein a C-class problem is exhaustively decomposed into a set of $\binom{C}{2}$ two-class problems. Even though this approach yielded results for these data sets that are superior to all previously reported methods for classification of hyperspectral data, we noted that this frame-

Received: 21 November 2000
Received in revised form: 2 November 2001
Accepted: 13 December 2001
[1] Some of the ideas in this paper were presented at the First International Workshop on Multiple Classifier Systems, 2000.

work requires $\mathcal{O}(C^2)$ pairwise classifiers, and therefore might not be as attractive if a large number of classes were involved. Further, combining the results of the $\binom{C}{2}$ two-class classifiers might lead to coupling problems.

In this paper, we propose a novel modular learning system comprised of an automatically generated binary hierarchy of classifiers, each solving a two-class problem and having its own feature space. The set Ω of C classes is first partitioned into two disjoint subsets, referred to as 'meta-classes'. The most appropriate partitioning, together with the linear feature extractor that best discriminates the two resulting meta-classes, is automatically learned. The meta-classes are further partitioned recursively until each meta-class is reduced to one of the C original classes. The resulting binary tree has C leaf nodes, one for each class, and $C-1$ internal nodes, each associated with a Bayesian classifier and a linear feature extractor. We illustrate the methodology by applying a hierarchical multiclassifier to data sets that involve classification of various landcover types using high dimensional hyperspectral data. The problem is a 12-class landcover prediction problem, where the input is a 180 band subset of the 224 bands (excluding water absorption bands) acquired by the NASA's Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor over Kennedy Space Center in Florida.

Apart from a significant improvement in classification accuracy, the proposed architecture also provided important domain knowledge that was consistent with a human expert's assessments in terms of the class hierarchy that was automatically discovered. A significant reduction in the number of features, as well as a reduction in the number of two-class classifiers from $\binom{C}{2}$ to only $C-1$, was also obtained using the hierarchical multiclassifier framework.

2. BACKGROUND AND RELATED WORK

The proposed hierarchical framework involves three major areas in which substantial research has been completed. We first review the relevant work in extraction of features as it is related to hyperspectral data analysis. Our algorithm utilized the Fisher discriminant and modular learning, so we also review the formulation of these approaches herein.

2.1. Feature Extraction from Hyperspectral Data

Hyperspectral sensors simultaneously acquire information in hundreds of spectral bands. A hyperspectral image is essentially a three-dimensional array $I(p,q,d)$, where (p,q) denotes a pixel location in the image, and d denotes a spectral band (wavelength). The value stored at $I(p,q,d)$ is the response (reflected or emitted energy) from the pixel (p,q) at a wavelength corresponding to spectral band d. The input space for a hyperspectral data classification problem is an ordered vector of real numbers of length D, the number of spectral bands, wherein the response of bands that are spectrally 'near' each other tend to be highly correlated.

Analysis of hundreds of simultaneous channels of data necessitates the use of either feature selection or extraction algorithms prior to classification. Feature selection algorithms for hyperspectral data classification are costly, while feature extraction methods based on KL-transforms, Fisher's discriminant or Bhattacharya distance cannot be used directly in the input space because the covariance matrices required by all these approaches are highly unreliable, given the ratio of the amount of training data to the number of input dimensions. The results are also difficult to analyse in terms of the physical characteristics of the individual classes and are not generalisable to other images.

Several authors have proposed approaches for extracting features from remotely sensed hyperspectral data. Lee and Landgrebe [9] proposed methods for *feature extraction based on decision boundaries* for both Bayesian and neural network based classifiers. In these methods, a classifier is first learned for a two-class problem in the input space. A decision boundary is computed by moving along the closest samples in the two classes, and a vector normal to the decision boundary is noted. Eigenvectors of the decision boundary feature matrix formed by collection of these normal vectors yields the direction of projection for the two-class problem.

Another feature extraction technique, which is based on *Segmented Principal Components Transformation* (SPCT) for two-class problems involving hyperspectral data, was proposed recently by Jia and Richards [10,11]. All the bands in the hyperspectral data are first partitioned into groups of highly correlated adjacent bands by applying an edge detector to the correlation matrix of the hyperspectral data. PCT is applied to each group of bands separately, and the first few eigenvectors (with the highest eigenvalues) are retained. This set of eigenvectors is subsequently pruned by applying feature selection using Bhattacharya distance as the measure of discrimination between the two classes.

The three key desirable properties of a feature extraction technique for hyperspectral data, as identified in Kumar et al [5], are:

1. **Class dependence**: different subsets of classes are best distinguished by different feature sets. Hence, feature extractors for specific groups of classes should be determined separately. Most classifiers seek only one set of features that distinguishes among all the classes simultaneously. This not only increases the complexity of the potential decision boundary, but also requires a large number of features and reduces the interpretability of the resulting features.
2. **Ordering constraint**: the characteristics that bands are ordered and adjacent bands are correlated should be exploited by the feature extraction algorithm. A Fisher or KL-transform on all the bands does not treat the input vector as a signal, and hence is not ideal for hyperspectral data feature extraction. Both the SPCT based feature extractor [10] and the projection pursuit based algorithm [12] utilise the ordering and locality properties of hyperspectral data. In general, any transformation should involve adjacent groups of bands.
3. **Discriminating transforms**: the transformations should try to maximise discrimination among classes, and thus use class label information. The KL-transform is suited

for preserving the variance in the data, but does not necessarily increase the discriminatory capacity of the feature space. Use of Fisher discriminant or Bhattacharya distance is therefore more desirable for feature extraction.

Recently, we developed a set of best bases algorithms [5,6] for feature extraction in hyperspectral data sets for two-class problems. The best-bases algorithms combined the highly correlated adjacent bands and extended the Local Discriminant Bases (LDB) approach [13], developed for signal and image classification. These algorithms satisfy the three properties desired of a feature extraction technique for hyperspectral data.

2.2. Fisher's Discriminant

Fisher's linear discriminant [14] is a classical feature extraction technique that linearly projects a D-dimensional Euclidean input space into a $min\{C-1,\ D\}$-dimensional feature space in which the discrimination among the C classes is maximum. In the binary hierarchical multiclassifier developed in Section 3, only two-class problems are solved at each internal node. Hence, we consider Fisher discriminant for two-class problems. Let ω_α and ω_β denote the two classes with D-dimensional mean vectors μ_α and μ_β and $D \times D$ covariance matrices Σ_α and Σ_β, respectively. Let $P(\omega_\alpha)$ and $P(\omega_\beta)$ denote the prior probabilities of the two classes. For this two-class problem, the Fisher discriminant projects the D-dimensional space onto a one-dimensional feature space. This projection is defined in terms of the WITHIN CLASS covariance matrix $\mathbf{W}$, a weighted sum of the covariances of the two classes and is given by:

$$\mathbf{W} = P(\omega_\alpha)\Sigma_\alpha + P(\omega_\beta)\Sigma_\beta, \tag{1}$$

and the BETWEEN CLASS covariance matrix $\mathbf{B}$ given by:

$$\mathbf{B} = (\mu_\alpha - \mu_\beta)\ (\mu_\alpha - \mu_\beta)^T. \tag{2}$$

The Fisher projection $\tilde{\mathbf{w}}$ that maximises the discriminant

$$\mathcal{T}(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{B}\mathbf{w}}{\mathbf{w}^T\mathbf{W}\mathbf{w}} \tag{3}$$

is given by:

$$\tilde{\mathbf{w}} = \mathbf{W}^{-1}\ (\mu_\alpha - \mu_\beta) \tag{4}$$

In the hierarchical multiclassifier architecture presented in this paper, the first and third properties are satisfied. Features for each group of classes are extracted independently, and the features are discriminating in nature as they are based on the Fisher's discriminant. Property 2 is not satisfied in the current version of the algorithm as the Fisher's discriminant is applied over all the bands. However, the method can be applied after preprocessing the data to combine highly correlated adjacent bands (using SPCT or a type of best bases algorithm, for example), in which case the overall system will also satisfy Property 2.

2.3. Modular Learning

Recently, there has been considerable interest in modular learning approaches where, instead of computing parameters for only one classifier, an 'ensemble' of classifiers is developed and applied, and their results are combined [3,4]. Inspired by the divide and conquer precept, modular learning has been found to learn concepts more effectively (better performance) and more efficiently (faster learning) because *learning a large number of simple local concepts is both easier and more useful than learning a single complex global concept* [15]. Modular learning, therefore, is a two-stage process wherein the first stage deals with a useful decomposition of the problem into simpler subproblems, and the second stage deals with solving the subproblems and finding a means of combining their solutions to yield the final solution. Advantages of problem decomposition include the ease and efficiency in learning, scalability, interpretability, and transparency [3,16–19].

Different ways of dividing a problem into simpler subproblems have been investigated by the pattern recognition and computational intelligence communities. Each sub-problem, for example, could focus on a different subset of input features (e.g. input decimation [20]), different parts of the input space (e.g. mixture of experts and its hierarchical versions [21]), or different training samples (e.g. boosting [22] and bagging [23]). Various approaches of modifying the output space have also been investigated. In error correcting output codes [24,25], for example, each class is assigned a unique binary code with additional error correcting bits, making it robust for a few bit errors. For each bit position of this code, we have a two meta-class problem to separate the classes with that bit being set to 1 from the rest. One problem with this is that the grouping of classes may not be natural, since it is not based on any similarity between them, but purely dependent on the codes assigned. The intermediate two-class problems can then become difficult to solve. We also note that discriminant analysis methods such as Linear Discriminant Analysis (LDA) [26], Quadratic Discriminant Analysis (QDA) [27,28] and Kernel Discriminant Analysis (KDA) [29–31] learn a 'discriminant function' for each class. Building C models, one each for discriminating a specific class from all the rest, has been shown to outperform a single complex model for discriminating all the classes at the same time [32].

Given a situation with both a large number of features and many classes, a hierarchical approach to partitioning classes has an extra benefit that the number of features required to realize a specific partition may be much lower than that required to partition the entire dataset into C classes simultaneously. This astute observation was first made in the context of hyperspectral classification by Kim and Landgrebe [33], who proposed a hierarchical classifier design algorithm that alternates between the bottom-up agglomeration of the entire dataset into two subgroups and then redividing the classes in each subgroup into two 'clusters' by a top-down approach. Another example that effectively employs classifiers and feature selectors in a hierarchical framework is provided in Chakrabarti et al [34], where a

hierarchical topic taxonomy is used to organise large text databases.

Decomposing a C-class problem into $\binom{C}{2}$ two-class problems, one for each unique pair of classes, has also been proposed previously. Friedman [35] and Tibshirani [36] proposed different methods to combine the outputs of the two-class models to yield an overall C-class system. We have investigated the pairwise classifier framework [37] in detail, and applied it to several remote sensing [8,38,6] and machine learning problems [7]. Such architectures are convenient when built using classifiers such as support vector machines that are better suited for two-class problems [39], and for feature extraction/evaluation methods such as Bhattacharya distance, KL-distance, etc., that are more meaningful for two-class problems. Even though good results have been obtained with such a framework, it suffers from one notable drawback: the number of pairwise classifiers required grows quadratically with the number of classes, so it is practical only for a moderate number of classes.

In this paper, a hierarchical multiclassifier is developed that recursively decomposes the output space into a binary tree, instead of the flat ensemble of $\binom{C}{2}$ classifiers used in the pairwise-classifier architecture described previously. Such a hierarchical ensemble is built using a top-down approach based on our Generalised Associative Modular Learning System (GAMLS) [15], in which modularity is introduced through soft association of each training sample with every module. GAMLS can be seen as a generalisation of the fuzzy c-means clustering, and can be applied to both supervised and unsupervised learning problems. Initially, a data point is equally associated with all the modules. The learning phase in GAMLS is comprised of two alternate steps: (i) for the current associations, update all the module parameters; and (ii) for the current module parameters, update the associations of all the training samples with each module. Using ideas from deterministic annealing, a temperature parameter is used to slowly converge the associations to hard partitions in order to induce specialisation and decoupling among the modules. A growing and pruning mechanism is also proposed for GAMLS that automatically leads to the right number of modules required for the data set. The GAMLS framework has previously been applied to unsupervised learning problems (e.g. clustering, density estimation using mixture of Gaussians, etc.) and supervised learning problems (e.g. mixture of experts). In building the class hierarchy, we utilise the basic ideas in GAMLS as follows. Instead of softly associating a *data point* with a module, a *class* is softly associated with one of the two meta-classes at each stage. These associations are updated over several iterations, and are hardened so that the set of classes is completely partitioned into two disjoint subsets.

The proposed hierarchical multiclassifier architecture is presented in Section 3. The new hierarchical multiclassifier requires only $C-1$ two-(meta)class classifiers. The resulting hierarchically arranged two-(meta)class classifiers are combined easily, and there is often automatic discovery of domain knowledge pertaining to the relationship among various classes, i.e. discovery of a class taxonomy for the domain.

3. THE HIERARCHICAL MULTICLASSIFIER ARCHITECTURE

In this section we describe a new hierarchical multiclassifier architecture that requires only $C-1$ pairwise classifiers arranged as a binary tree with C leaf nodes, one for each class, and $C-1$ internal nodes, each with its own feature space. The root node (indexed 1) of the binary tree represents the original C-class problem with its 'class-set' $\Omega_1 = \mathbf{\Omega}$. The two children of an internal node with index n are indexed $2n$ and $2n + 1$, and its class-set is denoted by Ω_n. Figure 1 shows an example of a Binary Hierarchical Classifier (BHC) for a $C = 5$ class problem. Each of the four internal nodes consists of a two-(meta)class classifier and a feature extractor specific to these two meta-classes.

The binary tree can be built via either a bottom-up or top-down approach [37]. The bottom-up method is similar to an 'agglomerative clustering' algorithm, in which the two 'closest' clusters[2] in the current set are merged at each stage until all data points are merged. The distance between any two (meta)classes, defined in the most discriminating feature space, is used to merge two classes and build the BHC. This approach requires computation of the distance between all pairs of classes at the very first stage, leading to the time complexity of the bottom-up approach at least $\mathcal{O}(C^2)$. In

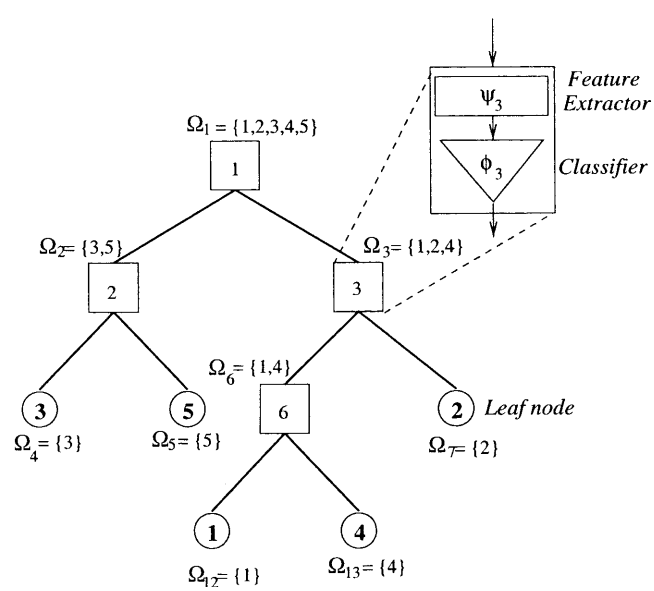

Fig. 1. An example of a BINARY HIERARCHICAL CLASSIFIER for a $C = 5$ class problem with 4 internal nodes and 5 leaf nodes. Each internal node n is comprised of a feature extractor ψ_n and a classifier ϕ_n. Each node n is associated with a set of classes Ω_n. The left and right children of internal node n are indexed $2n$ and $2n + 1$, respectively.

[2] In this case, a cluster is a collection of classes instead of a collection of data points. A cluster of classes is also referred to as a meta-class.

this paper, a top-down approach for building the BHC is presented. Here, the tree construction starts at the root node with all C classes, and tries to partition this set into two disjoint subsets in such a way that the classes that are assigned to one set are more similar to each other than to the classes in the other set. The process is recursively repeated at each resulting leaf node of the tree until all leaf nodes contain only one of the original C classes. The top-level recursive partitioning algorithm is:

BUILD TREE(Ω_n)

1. Partition Ω_n into two subsets: $(\Omega_{2n}, \Omega_{2n+1}) \leftarrow$ PARTITION NODE(Ω_n)
2. Recurse on each child:
   - if $|\Omega_{2n}| > 1$ then BUILD TREE(Ω_{2n})
   - if $|\Omega_{2n+1}| > 1$ then BUILD TREE(Ω_{2n+1})

The PARTITION NODE function and node n finds a partition of the set of classes Ω_n into two disjoint subsets such that the discrimination between the two meta-classes Ω_{2n} and Ω_{2n+1} is high. It also finds a linear projection of the original D-dimensional space onto a one-dimensional space in which such discrimination is maximum. The two problems of finding a class-based partition and determining the feature extractor that maximises discrimination between the meta-classes obtained as a result of this partition are coupled. Moreover, the problem of optimally partitioning $|\Omega_n|$ classes into two disjoint subsets is an NP time complexity problem. Soft partitioning using the GAMLS approach and simulated annealing, together with a deterministic initialisation of the initial partition are used in the PARTITION NODE algorithm described next.

3.1. Partitioning a Set of Classes

Let Ω be any class set with $K = |\Omega| > 2$ classes that needs to be partitioned into two meta-classes, Ω_α and Ω_β. The 'association' between a class $\omega \in \Omega$ and a meta-class Ω_γ, $(\gamma \in (\alpha, \beta\})$ is interpreted as the posterior probability of ω belonging to Ω_γ, and is denoted by $P(\Omega_\gamma|\omega)$:

$$P(\Omega_\alpha|\omega) + P(\Omega_\beta|\omega) = 1, \quad \forall \omega \in \Omega. \tag{5}$$

Let μ_ω and Σ_ω denote the estimated sample mean vectors and covariance matrices, respectively, of the classes $\omega \in \Omega$. Let χ_ω denote the training set comprised of $N_\omega = |\chi_\omega|$ examples of class ω. Given the current settings for the associations of classes to meta-classes (i.e. $\{P(\Omega_\gamma|\omega), \gamma \in \{\alpha, \beta\}\}_{\omega\in\Omega}$, the mean μ_γ and covariance Σ_γ $(\gamma \in \{\alpha, \beta\})$ of the meta-classes are computed. These meta-classes are then projected along the Fisher direction, and the associations are updated in this one dimensional projected space. Given the updated associations, a lower temperature is used in the next iteration of re-estimating the means and covariance of the meta-classes so the associations obtained in the next iteration are sharper.

The detailed steps of the partitioning algorithm for the set of classes Ω is as follows:

1. Initialise $P(\Omega_\alpha|\omega_1) = 1$ for some $\omega_1 \in \Omega$ and $P\Omega(_\alpha|\omega) = 0.5$, $\forall \omega \in \Omega - \omega_1$. Temperature $T = T_0$ (user defined parameter).
2. Compute the means and covariances of the meta-classes:

$$\mu_\gamma = \sum_{\omega\in\Omega} P(\omega|\Omega_\gamma)\mu_\omega, \quad \gamma \in \{\alpha, \beta\} \tag{6}$$

$$\Sigma_\gamma = \sum_{\omega\in\Omega} \frac{P(\omega|\Omega_\gamma)}{N_\omega}\left[\sum_{\mathbf{x}\in\chi_\omega} (\mathbf{x} - \mu_\gamma)\,(\mathbf{x} - \mu_\gamma)^T\right], \tag{7}$$

$\gamma \in \{\alpha, \beta\}$.

Note that Eq. (7) is $\mathcal{O}(N)$ but can be reduced to $\mathcal{O}(|\Omega|)$ by a simple manipulation, leading to:

$$\Sigma_\gamma = \sum_{\omega\in\Omega} \mathrm{P}(\omega|\Omega_\gamma)\,[\Sigma_\omega + (\mu_\omega - \mu_\gamma)\,(\mu_\omega - \mu_\gamma)^T], \tag{8}$$

$\gamma \in \{\alpha, \beta\}$.

The probabilities $P(\omega|\Omega_\gamma)$ are computed using Bayes rule: $P(\omega|\Omega_\gamma) = \dfrac{P(\omega)P(\Omega_\gamma|\omega)}{P(\Omega_\gamma)}$, $\gamma \in \{\alpha, \beta\}$, and the meta-class priors $P(\Omega_\gamma)$ are given by: $P(\Omega_\gamma) = \Sigma_{\omega\in\Omega}P(\Omega_\gamma|\omega)P(\omega|\Omega)$, $\gamma \in (\alpha, \beta\}$, where the conditional class priors $P(\omega|\Omega)$ are given by: $P(\omega|\Omega) = \dfrac{P(\omega)}{P(\Omega)} = \dfrac{P(\omega)}{\Sigma_{\rho\in\Omega}P(\rho)}$, since $\omega \in \Omega$. The class priors $P(\omega) = \dfrac{N_\omega}{N}$, where $N = \Sigma_{\omega\in\Omega}N_\omega$.

3. Compute the Fisher projection vector $\mathbf{w}$ using Eq. (4) with the sample mean and covariance of the meta-classes for the parameters in Eqs (6) and (8).
4. Compute the mean log-likelihood of meta-classes $\Omega_\gamma(\gamma \in \{\alpha, \beta\})$:

$$\mathcal{L}(\Omega_\gamma|\omega) = \frac{1}{N_\omega}\sum_{\mathbf{x}\in\chi_\omega} \log p(\mathbf{w}^T\mathbf{x}|\Omega_\gamma), \tag{9}$$

$\gamma \in \{\alpha, \beta\}, \quad \forall\ \omega \in \Omega$

where the pdf of each Ω_γ in the one dimensional projected space is modelled as Gaussian:

$$p(\mathbf{w}^T\mathbf{x}|\Omega_\gamma) = \mathcal{N}(\mathbf{w}^T\mathbf{x}; \mathbf{w}^T\mu_\gamma, \mathbf{w}^T\Sigma_\gamma\mathbf{w}), \quad \gamma \in \{\alpha, \beta\} \tag{10}$$

5. Update the meta-class posteriors:

$$P(\omega_\alpha|\omega) = \frac{\exp(\mathcal{L}(\Omega_\alpha|\omega)/T)}{\exp(\mathcal{L}(\Omega_\alpha|\omega)/T) + \exp(\mathcal{L}(\Omega_\beta|\omega)/\mathbf{T})}, \tag{11}$$

6. Repeat Steps 2 through 4 until the incremental increase in $\mathcal{T}(\mathbf{w})$ (Eq. (3)) is insignificant (e.g. less than 5%).
7. Compute the entropy of meta-class posteriors:

$$\mathcal{H} = -\frac{1}{|\Omega|}\sum_{\omega\in\Omega} [P(\Omega_\alpha|\omega)\log_2 P(\Omega_\alpha\omega) + P(\Omega_\beta|\omega)\log_2 P(\Omega_\beta|\omega)]. \tag{12}$$

8. If $\mathcal{H} < \theta_H$ (user defined threshold) stop, otherwise:
   - Cool temperature: $T \leftarrow T\theta_T$ ($\theta_T < 1$ is a user defined cooling parameter)
   - Go to Step 2.

Each internal node n of the binary tree contains a projection vector $\mathbf{w}(n)$, and the estimates of parameters (μ_k, Σ_k,

$k \in \{2n, 2n+1\}$). The pairwise classifier at node n can be either a soft or a hard classifier. The *soft classifier* ϕ_n^S: $\mathcal{R}^D \rightarrow [0, 1]$ generates the posterior probability $P(\Omega_{2n}|\mathbf{x}, \Omega_n)$. The posterior $P(\Omega_{2n+1}|\mathbf{x}, \Omega_n) = 1 - \phi_n^s(\mathbf{x})$. The *hard classifier* ϕ_n^H: $\mathcal{R}^D \rightarrow \{\Omega_{2n}, \Omega_{2n+1}\}$ maps the input $\mathbf{x}$ into one of the two class labels corresponding to the two child nodes of class Ω_n. Essentially, the output of the soft classifier is thresholded to obtain the hard classifier:

$$\phi_n^H(\mathbf{x}) = \begin{cases} \Omega_{2n} & \text{if } \phi_n^S(\mathbf{x}) > \zeta_n \\ \Omega_{2n+1} & \text{otherwise} \end{cases} \tag{13}$$

where ζ_n is the threshold based on misclassification costs and the priors $P(\Omega_{2n})$ and $P(\Omega_{2n+1})$ [40].

3.2. Generalisation

A novel test example is classified by the hierarchical multiclassifier by pushing it from the root node to the leaf node(s). Depending on the nature of internal classifiers. (i.e. hard or soft), there are two ways of assigning a class label to a novel input $\mathbf{x}$.

1. **Hard classification**: the output of the hard classifier at internal node n, $\phi_n^H(\mathbf{x})$, is a class label. In this case, at each internal node, the point is pushed to one of the two children. More specifically, the hard classification algorithm can be written as:
   (i) Initialise $n = 1$ (start at root node)
   (ii) while node n is an internal node, do
   Push point $\mathbf{x}$ to the appropriate child:
   $$n \leftarrow \begin{cases} 2n & \text{if } \phi_n^H(\mathbf{x}) = \Omega_{2n} \\ 2n+1 & \text{if } \phi_n^H(\mathbf{x}) = \Omega_{2n+1} \end{cases} \tag{14}$$
   (iii) Assign the (unique) class label Ω_n at the leaf node n to $\mathbf{x}$.
2. **Soft classification**: if a soft classifier is used at each internal node, the results of these hierarchically arranged classifiers can be combined by first computing the overall posteriors $P(\omega|\mathbf{x})$, and then applying the Maximum Aposterior Probability (MAP) rule:
   $$\omega(\mathbf{x}) = arg \max_{\omega \in \mathbf{\Omega}} P(\omega|\mathbf{x}), \tag{15}$$
   to assign the class label $\omega(\mathbf{x})$ to $\mathbf{x}$. The posteriors $P(\omega|\mathbf{x})$ can be computed from product of the posterior probabilities of all the internal node classifiers on the path to the corresponding leaf node (Theorem 1).

   Theorem 1. *The posterior probability $P(\omega|\mathbf{x})$ for any input $\mathbf{x}$ is the product of the posterior probabilities of all the internal classifiers along the unique path from the root node to the leaf node $n(\omega)$ containing the class ω, i.e.*
   $$P(\omega|\mathbf{x}) = \prod_{\ell=0}^{\mathcal{D}(\omega)-1} P(\Omega_{n(\omega)}^{(\ell+1)}|\mathbf{x}, \Omega_{n(\omega)}^{(\ell)}), \tag{16}$$
   where $\mathcal{D}(\omega)$ is the depth of $n(\omega)$ (depth of the root node is 0), $\Omega_n^{(\ell)}$ is the meta-class at depth ℓ in the path from the root node to $n(\omega)$, such that $\Omega_{n(\omega)}^{(\mathcal{D}(\omega)} = \{\omega\}$ and $\Omega_{n(\omega)}^{(0)} = \Omega_1$ = root node.

   Proof. See Appendix A.

4. EXPERIMENTAL RESULTS

The efficacy of the proposed multiclassifier architecture for hyperspectral data analysis is demonstrated by an experiment on a hyperspectral data set acquired over Kennedy Space Center, Florida. The wetlands located on the west shore of Kennedy Space Center (KSC) and the Indian River are critical habitats for several species of water fowl and aquatic life. Mapping the landcover and its response to wetland management practices using remotely sensed data from a variety of sensors is the focus of a multi-year project between NASA and The University of Texas at Austin. In 1996 hyperspectral data were acquired by NASA JPL at 20 m spatial resolution using the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). The test site for this study consists of a series of impounded marshes with vegetation communities ranging from low, halophyte marshes to high, graminoid savannahs to forested wetlands. Discrimination between individual species of marsh vegetation and of woodland vegetation types is quite difficult due to similarity of spectral signatures.

The hierarchical multiclassifier was applied to a $D = 180$ band subset of the 224 bands (excluding water absorption bands) of the AVIRIS data set. The seven upland and five wetland cover types identified for classification are listed in Table 1: Classes 3–7 [cabbage palm hammock (3), cabbage palm/oak hammock (4), slash pine (5), broadleaf/oak hammock (6), and hardwood swamp (7)] are all trees. Class 4 is a mixture of Class 3 and oak hammock. Class 6 is a mixture of broadleaf trees (maples and laurels) and oak

Table 1. The twelve landcover classes in the KSC/AVIRIS hyperspectral data

Num	Class name
	Upland classes
1	Scrub
2	Willow Swamp
3	Cabbage palm hammock
4	Cabbage oak hammock
5	Slash pine
6	Broadleaf/oak hammock
7	Hardwood swamp
	Wetland classes
8	Graminoid marsh
9	Spartina marsh
19	Cattail marsh
11	Salt marsh
12	Mud flats

hammock. Class 7 is also a broadleaf tree. These classes have similar spectral signatures and are very difficult to discriminate in multispectral and even hyperspectral data using traditional methods.

There were typically 350 examples for each class,[3] hand labelled by a domain expert. These were randomly partitioned into 50% training and 50% test sets. If the number of data points in the training set was less than 180, then additional examples were randomly selected from the samples so that there were at least 180 examples per class in the training set. The hierarchical tree shown in Fig. 2 was obtained independently, in 8 out of 10 experiments. The 12 classes were also grouped by a human expert based on traditional characterisation of vegetation into seven upland and five wetland classes (Table 1). Classes 1, 3, 4, 5 and 6 are all trees that grow in an upland environment. Classes 2 and 7 are also trees, but the soil is saturated if not inundated much of the time. Classes 8–12 are generally characterised as marsh grasses. Here, the soil is usually saturated and periodically inundated. Even though willow swamp (Class 2) and hardwood swamp (Class 7) are actually wetland species, they were designated as members of the uplands group by the expert due to their biomass. In light of these observations, the class partitioning shown in Fig. 2 obtained by the proposed multiclassifier architecture from the training data is remarkable as it not only conforms to the expert's opinions, but is also able to designate Classes 2 and 7 as members of the same group.

The meta-class associated with the root node contains all 12 classes. It is then split into two meta-classes. Figure 3 shows the changes in the association of each of the 12 classes at the root node with the meta-class of its right child. As per the initialisation step in the PARTITION NODE algorithm, the association of Class 1 with the right child meta-class is fixed at 1 (for all iterations), while the association of all other classes is 0.5. As the iterations progress,

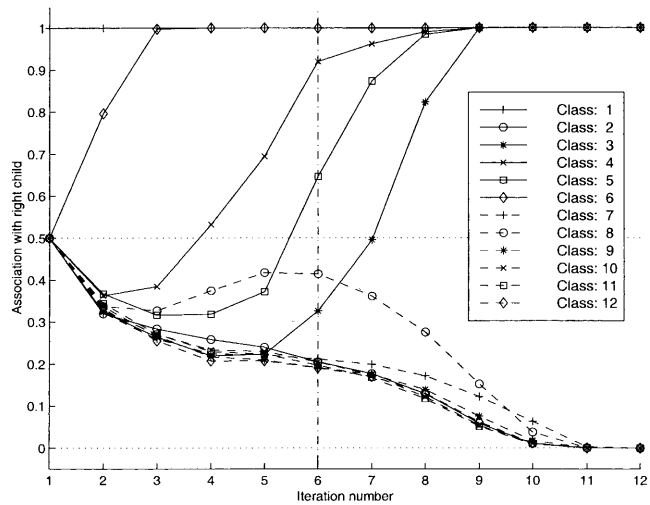

Fig. 3. Split diagram for the root node in the multiclassifier tree of the AVIRIS dataset shown in Figure (2). Starting from associations of 0.5 (except for class 1), the classes align themselves either with the right child (top) containing class 1 or with the left child (bottom). The only phase transition, i.e., change in temperature, was at iteration 6.

Class 6 immediately moves towards Class 1 while all other classes move away. Then, Classes 3, 4 and 5 also move toward Class 1 or the right child, while the other seven classes (i.e. Class 2 and Classes 6–12) continue to move toward the left child. The vertical line (at Iteration 6) denotes the 'phase transition' [41] or decrease in temperature from $T_0 = 1$ to 0.9 ($\theta_T = 0.9$ in Step 7 of the PARTITION NODE algorithm). Figure 4 shows the changes in the value of Fisher discriminant (top) and entropy of the association (Eq. (15) (bottom) as the 12 classes are progressively split into two meta-classes by the PARTITION NODE algorithm at the first (root node) split in the tree. The Fisher discriminant

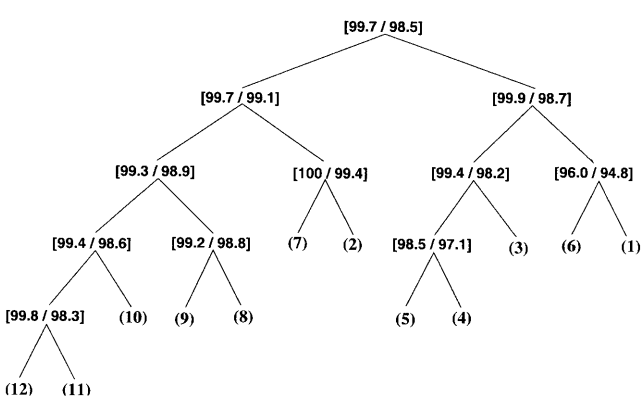

Fig. 2. Multiclassifier binary tree for AVIRIS data: The 12 classes are listed in Table (1). Each leaf node in the binary tree is labeled with one of the 12 classes it represents. The numbers on an internal node represent the classification accuracy of the two-class classifier at that node on the training data and the test data respectively.

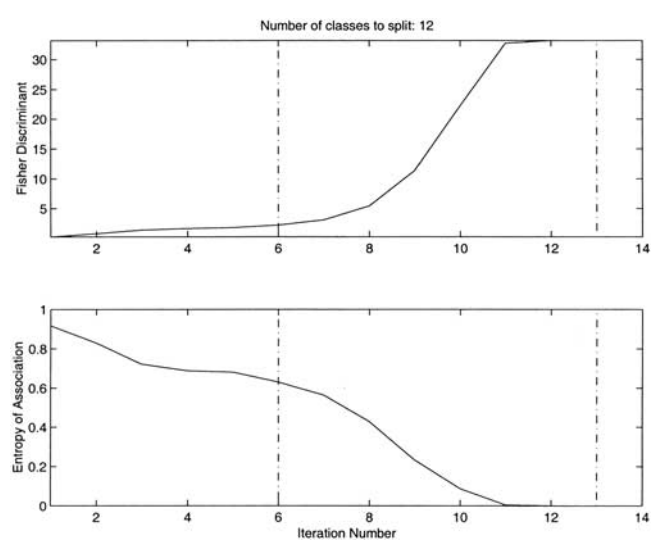

Fig. 4. (a) (top) the Fisher discriminant between the two meta-classes at the two children nodes of the root node. (b) (bottom) Corresponding change in the entropy of the associations.

[3] For some classes less than 200 labeled examples were available causing severe data sparsity for the 180-dimensional space.

increases, indicating that the separation between the two meta-classes increases. The entropy $\mathcal{H}$ of the association of all 12 classes with the two meta-classes decreases to less than .05 ($\theta_{\mathcal{H}}$) in 13 iterations. As the entropy decreases, the association of each class with either of the two meta-classes becomes 'harder' (i.e. 0 or 1) after starting at 0.5.

Using the hard and soft combining techniques described in Section 3.2, novel examples from the test were classified using the new hierarchical approach and compared to results from other algorithms. The overall classification accuracy on the test set averaged over the 10 experiments using *hard combining* was 94.7% while it was 96.8% using soft combining. This was a significant improvement over the 93% classification accuracy obtained by our previously developed Bayesian pairwise classifier architecture that uses class pair dependent feature selection and a maximum likelihood classifier[4] for each pair of classes. The pairwise classifier with best-bases features had previously yielded the best classification accuracies, and its results are also compared to those obtained here. To compare with a single classifier approach, an MLP with 50 hidden units, 183 inputs and 12 output units was trained until the change in training accuracy was insignificant. The test accuracy averaged over 10 experiments was found to be only 74.5%. This performance can be attributed at least partially to the lack of sufficient labelled examples for the complexity of the model. An ensemble of C '1 vs rest' MLP's, each with six hidden units and one output unit was also trained to compare such a modular learning architecture with the hierarchical architecture. Referred to as MLPC classifier in Table 2, this architecture yielded an accuracy of 84.8%, a significant improvement over the single MLP classifier.

Additionally, the classification accuracy of the hierarchical multiclassifier was at least 10% higher than that of traditional feature extraction methods based on principal component analysis [10] and MNF transforms [42]. Table 2 shows the comparative accuracies of the various classifiers, including the traditional Maximum Likelihood Classifiers (MLC). These classifiers use a Gaussian distribution with full covariance matrix to model the probability density function for each class and assume equal class priors. However, due to the high dimensionality of the data, a small constant that is proportional to the trace of the covariance matrix is added to the diagonal elements of the covariance matrix whenever the covariance matrices are ill-conditioned due to the small amount of training data in some classes.

The BHC is the binary hierarchical classifier proposed herein. The GLDB-BU and GLDB-TD are the bottom-up and top-down algorithms for best bases feature extraction used with the pairwise classifier architecture. LDB and SPCT are the local discriminant bases [13] and segmented principal components transform [10], also used with the pairwise classifier architecture. Classification accuracies of the GLDB-BU and BHC approaches were comparable, but the computational requirements and model complexity of the BHC algorithm are significantly lower. Note that the GLDB-BU and GLDB-TD classifiers have two potential advantages over the proposed BHC classifier: (i) the features extracted in the GLDB classifiers are more suited to hyperspectral data than simple Fisher discriminants that are currently used in BHC. Moreover, in GLDB, some class pairs utilise more than 1-dimensional features, while the features in BHC are all 1-dimensional; and (ii) the data sparsity problem for high dimensional data is better addressed by the GLDB algorithms as they employ subsets of adjacent bands. The Fisher discriminant used in the BHC classifiers use all 180 dimensions simultaneously, and therefore require more data. In spite of all these advantages, the BHC classifiers perform comparable to the GLDB classifiers in terms of classification accuracy.

Table 2. Mean classification accuracies and standard deviations over 10 experiments for the KSC hyperspectral data sets on (i) Maximum Likelihood Classifier (MLC), (ii) Multilayered perceptron (MLP), (iii) Ensemble of 12 Multilayered perceptrons a 1 vs. the other 11 class modular architecture (MLPC), (iv) Local Discriminant Bases (LDB), (v) Segmented Principal Components Transform (SPCT), the two best-bases algorithms; (vi) Generalised local discriminant bases – top-down (GLDB-TD), (vii) GLDB-Bottom-Up (GLDB-BU), and (viii) Binary Hierarchical Classifier (BHC) using both soft (BHC-soft) and hard (BHC-hard) combining

Algorithm	Accuracy (Standard Deviation)
MLC	72.7% (3.11)
MLP	74.5% (2.87)
MLPC	84.8% (3.34)
LDB	79.3% (2.20)
SPCT	78.7% (2.45)
GLDB-TD	86.2% (2.71)
GLDB-BU	95.3% (2.22)
BHC-hard	94.7% (1.96)
BHC-soft	96.8% (2.17)

The data sparsity issue we encountered in our experiments due to high input dimensionality and lack of sufficient ground truth data for some classes was addressed by using 'enough' examples for each class for training. Here we mention two possible ways of addressing this problem more systematically:

- *Regularised covariance matrices*: Friedman proposed Regularised Discriminant Analysis (RDA) [43] to 'smooth out' the effect of ill-conditioned covariance estimates due to sparse data. In RDA, instead of using the class covariance matrix, a linear combination of the class covariance matrix (based on sparse data) and the overall covariance of the data is used. If $\hat{\Sigma}_c$ is the class covariance for some class c and $\hat{\Sigma}$ is the overall covariance of the data then RDA proposes the following estimate:

$$\hat{\Sigma}_c(\lambda) = \frac{(1-\lambda)N\hat{\Sigma} + \lambda N_c\hat{\Sigma}_c}{(1-\lambda)N + \lambda N_c} \tag{17}$$

[4] In this, the probability density function of each class is modelled by a full covariance Gaussian over all the selected features.

where, $\lambda \in [0, 1]$ is a user defined parameter, N is the total number of samples in all the classes, and N_c is the number of samples in the class c. The BHC framework built in a top-down fashion provides an opportunity to fine-tune the RDA approach into a hierarchical regularised discriminant analysis. Such an approach, referred to as 'Shrinkage', has been applied by McCallum et al [44] for improving text classification in a hierarchy of classes. Another regularisation of covariance matrices is proposed by Tadjudin and Landgrebe [45]. This method is based on inverted Wishart distributions with leave-one-out average log likelihood criteria to estimate covariances with limited training samples, especially for the cases when the number of samples is less than dimensionality of the data.

- *Using localised bases*: the second method for dealing with sparsity, especially for hyperspectral data, is to use localised bases, i.e. bases that depend on small subsets of adjacent spectral bands. The SPCT algorithm and the GLDB algorithms are examples of such an approach.

The use of these techniques on BHC architecture will be investigated in the future because data sparsity problem is common in hyperspectral remote sensing. The original data scene and the classified maps obtained by various classifiers can be viewed at http://www.csr.utexas.edu/rs/research/hyper.html.

Apart from automatic discovery of class taxonomy and significant improvement in classification accuracy, the hierarchical multiclassifier also reduces the number of features significantly since, only a one-dimensional feature space is used to solve a two-class problem at each internal node. As compared to the pairwise classifier that requires $\binom{C}{2}$ two-class classifiers, the hierarchical classifier requires only C–1 two-class classifiers, although these classifiers try to discriminate 'meta-classes' as opposed to the original classes. For difficult classification problems with a large number of classes, this reduction in the number of two-class classifiers could be significant. Both the pairwise and hierarchical approaches are novel alternatives of decomposing a C-class problem into simpler two-class problems. The two-class problem at each internal node in the hierarchical tree is more complex than the two-class problems in the pairwise classifiers. Thus, an inherent trade-off between the number of classifiers and their complexity and the nature of domain knowledge that can be extracted from both these architectures make them equally interesting.

5. CONCLUSIONS

A hierarchical multiclassifier architecture is proposed for the analysis of hyperspectral data in problems where there is a moderately large number of classes, C. An algorithm using the generalized associative modular learning paradigm was developed for recursively partitioning a set of classes into two groups and simultaneously finding the best feature projection that distinguishes the two groups. Much simpler features and classifiers are needed for each partitioning in the hierarchy as compared to solutions that attempt to solve the C-class problem in one step. In terms of classification accuracy, the results obtained on a 180-dimensional hyperspectral dataset for a 12-class problem were both significantly better than approaches based on other feature extraction and problem decomposition techniques. Moreover, the automatically discovered class taxonomy conforms well to expert opinion and therefore provides significant domain knowledge about the relationships between different classes.

Acknowledgements

This research was supported in part by ARO contracts DAAD19-99-1-0012 and DAAG55-98-1-0287, NSF grant ECS-9900353, the Texas Advanced Technology Research Program (CSRA-ATP-009), and a grant from Intel Corp. We would also like to thank Amy Neuenshwander, Joseph Troy Morgan, and Alexandre Henneguelle for their support and critical review of this research.

References

1. Bellman RE (ed). Adaptive Control Processes. Princeton University Press, 1961
2. Friedman JH. On bias, variance, loss, and the curse of dimensionality. Technical report, Department of Statistics, Stanford University, 1996
3. Murray-Smith R, Johansen TA. Multiple Model Approaches to Modelling and Control. Taylor & Francis, UK, 1997
4. Sharkey A (ed). Combining Artificial Neural Nets. Springer-Verlag, 1999
5. Kumar S, Ghosh J, Crawford MM. Multiresolution feature extraction for pairwise classification of hyperspectral data. Proc SPIE: Applications of Artificial Neural Networks in Image Processing V, IS&T/SPIE's Electronic Imagining, January 2000; 60–71
6. Kumar S, Ghosh J, Crawford MM. Best-bases feature extraction algorithms for classification of hyperspectral data. IEEE Trans Geoscience and Remote Sensing, July 2001; 39(7): 1368–1379
7. Kumar S, Ghosh J, Crawford MM. A versatile framework for labeling imagery with large number of classes. Proceedings International Joint Conference on Neural Networks, Washington, DC, 1999
8. Crawford MM, Kumar S, Richard MR, Gibeaut JC, Neuenschwander A. Fusion of airborne polarimetric and interferometric SAR for classification of coastal environments. IEEE Trans Geoscience and Remote Sensing 1999; 37(3): 1306–1315
9. Lee C, Landgrebe DA. Decision boundary feature extraction for neural networks. IEEE Trans Neural Networks 1997; 8(1): 75–83
10. Jia X, Richards JA. Segmented principal components transformation for efficient hyperspectral remote-sensing image display and classification. IEEE Trans Geoscience and Remote Sensing 1999; 37(1): 538–542
11. Jia X. Classification techniques for hyperspectral remote sensing image data. PhD thesis, University College, ADFA, University of New South Wales, Australia, 1996
12. Jimenez LO, Landgrebe DA. Hyperspectral data analysis and supervised feature reduction via projection pursuit. IEEE Trans Geoscience and Remote Sensing 1999; 37(6): 2653–2664
13. Saito N, Coifman RR. Local discriminant bases. Mathematical Imaging: Wavelet Applications in Signal and Image Processing II, Proc. of SPIE 1994; 2303: 2–14
14. Fisher RA. The use of multiple measurements in taxonomic problems. Ann Eugenics 1936; 7: 179–188

15. Kumar S, Ghosh J. GAMLS: A generalized framework for associative modular learning systems (invited paper). Proceedings Applications and Science of Computation Intelligence II, Orlando, FL, 1999; 24–34
16. Ballard D. Modular learning in neural networks. Proc AAAI-87, 1987; 279–284
17. Happel BLM, Murre JMJ. Design and evolution of modular neural network architectures. Neural Networks 1994; 7: 6/7: 985–1004
18. Ramamurti V, Ghosh J. Structurally adaptive modular networks for nonstationary environments. IEEE Trans Neural Networks 1999; 10(1): 152–160
19. Petridis V, Kehagias A. Predictive Modular Neural Networks: Applications to Time Series. Kluwer Academic, Boston, 1998
20. Tumer, K, Oza NC. Decimated input ensembles for improved generalization. Proceedings International Joint Conference on Neural Networks, Washington, DC, 1999
21. Jordan MI, Jacobs RA. Hierarchical mixture of experts and the EM algorithm. Neural Computation 1994; 6: 181–214
22. Freund Y, Schapire RE. Experiments with a new boosting algorithm. Proceedings 13th International Conference on Machine Learning, Morgan Kaufman, 1996; 148–156
23. Breiman L. Bagging predictors. Machine Learning 1996; 24(2): 123–140
24. Sejnowski TJ, Rosenberg CR. Parallel networks that learn to pronounce English text. J Complex Systems 1987; 1(1): 145–168
25. Dietterich TG, Bakiri G. Solving multiclass learning problems via error-correcting output codes. J Artif Intell Res 1995; 2(1): 263–286
26. Nilsson NJ. Learning Machines: Foundations of Trainable Pattern-Classifying Systems. McGraw Hill, NY, 1965
27. McLachlan GJ. Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York, 1992
28. Hastie T, Tibshirani R. Discriminant adaptive nearest neighbor classification. IEEE Trans Pattern Analysis and Machine Intelligence 1996; PAMI-18(6): 607–616
29. Hand D (ed). Kernel Discriminant Analysis. Research Studies Press, Chichester, 1982
30. Chakravarthy S, Ghosh J, Deuser L, Beck S. Efficient training procedures for adaptive kernel classifiers. Neural Networks for Signal Processing, IEEE Press, 1991; 21–29
31. Ghosh J, Chakravarthy SV. The rapid kernel classifier: A link between the self-organizing feature map and the radial basis function network. J Intelligent Material Systems and Stuctures 1994; 5: 211–219
32. Anand R, Methrotra K, Mohan CK, Ranka S. Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Networks 1995; 6(1): 117–125
33. Kim B, Landgrebe DA. Hierarchical classifier design in high-dimensional numerous class cases. IEEE Trans Geoscience and Remote Sensing July 1991; 29(4). 518–528
34. Chakrabarti S, Dom B, Agrawal R, Raghavan P. Scalable feature selection, classication and signature generation for organizing large text databases into hierarchical topic taxonomies. VLDB J 1998; 7(3): 163–178
35. Friedman J. Another approach to polychotomous classification. Technical report, Stanford Univeristy, 1996
36. Hastie T, Tibshirani R. Classification by pairwise coupling. In: Karens MJ, Jordan MI, Solla SA (eds). Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 1998; 10: 507–513
37. Kumar S. Modular Learning through Output Space Decomposition. PhD dissertation, The University of Texas at Austin, Austin, TX, 2000
38. Kumar S, Ghosh J, Crawford MM. A hierarchical multiclassifier system for hyperspectral data analysis. Lecture Notes in Computer Science 2000; 1857: 270–279
39. Weston J, Watkins C. Multi-class support vector machines, 1998
40. Domingos P. Metacost: A general method for making classifiers cost-sensitive. In Proceedings Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, CA, 1999; 155–164
41. Rose K, Gurewitz E, Fox GC. Statistical mechanics and phase transitions in clustering. Physical Review Letters 1990; 65(8).
42. Green AA, Berman M, Switzer P, Craig M. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. IEEE Trans Geoscience and Remote Sensing 1988; 26(1): 65–74.
43. Friedman J. Regularized discriminant analysis. J Am Statistical Assoc 1989; 84(405): 165–175
44. McCallum A, Rosenfield R, Mitchell T, Ng AY. Improving text classification by shrinkage in a hiearchy of classes. Proceedings 15th International Conference on Machine Learning, San Francisco, CA, 1998; 359–367
45. Tadjudin S, Landgrebe D. Covariance estimation with limited training samples. IEEE Trans Geoscience and Remote Sensing July 1999; 37(4): 2113–2118

Shailesh Kumar received his PhD in computer engineering in 2000 and Masters in Computer Science in 1998, both from the University of Texas at Austin. During his graduate studies, he was associated with the Laboratory of Artificial Neural Systems (LANS) and Center for Space Research (CSR) at the University of Texas at Austin. He received his Bachelor of Technology degree in computer science and engineering from the Institute of Technology, Banaras Hindu University (IT-BHIU), Varanasi, India in May 1995. He is currently working as a staff scientist in the Advanced Technology Solutions division of HNC Software in San Diego, CA, USA. Dr Kumar's research interests are in pattern recognition, machine learning, data mining, remote sensing, wavelet technology, and image processing. During his graduate studies, he published over 20 conference papers and several journal papers and book chapters in these areas.

Joydeep Ghosh was educated at IIT Kanpur (BTech 1983) and The University of Southern California (MS, PhD, 1988). Subsequently he joined the Department of Electrical and Computer Engineering at the University of Texas, Austin, where he has been a Full Professor since 1998, and holder of the Archie Straiton Endowed Fellowship. Dr Ghosh directs the Laboratory for Artificial Neural Systems (LANS), where his research group is studying the theory and applications of adaptive pattern recognition, data mining including web mining, and multi-learner systems. Dr Ghosh has published more than 200 refereed papers and edited eight books. He has received six best paper awards, including the 1992 Darlington Prize for the Best Journal Paper from *IEEE Circuits and Systems Society*, and the Best Applications Paper at *ANNIE'97*. Dr Ghosh served as the general chairman for the SPIE/SPSE Conference on Image Processing Architectures, Santa Clara, 1990, as Conference Co-Chair of Artificial Neural Networks in Engineering (ANNIE)'93-ANNIE'96, ANNIE'98–2001, and in the program or organising committees of several conferences on neural networks and parallel processing. More recently, he co-organised workshops on Web Mining (with SIAM Int. Conf. on Data Mining, 2001) and on Parallel and Distributed Data Mining (with KDD-2000). He was a plenary speaker for ANNIE'97 and letters editor for *IEEE Trans. Neural Networks* (1998–2000). He is currently an associate editor of *Pattern Recognition, Neural Computing Surveys* and the *Int. J. of Smart Engineering Design.*

Melba M. Crawford received the BS and MS degrees in civil engineering from the Univesity of Illinois, Urbana, in 1970 and 1973 respectively, and the PhD degree in industrial and systems engineering from the Ohio State University, Columbus, in 1981. She joined the Operations research faculty in the Department of Mechanical Engineering at the University of Texas (UT), Austin, in 1980. She has been a full professor since 1991 and currently holds an Engineering Foundation Endowed Professorship. Dr. Crawford is an Associate Director of the UT Environmental Science Institute and heads the Remote Sensing Research Program at the UT Center for Space Research that includes research in multispectral, hyperspectral, SAR, and LIDAR systems and applications. Her research focuses on development of algorithms for analysis of remotely sensed data, including classification, segmentation, reconstruction, and multi-sensor fusion. She has published over 90 conference and refereed journal papers. She is a member of the EO-1 Science Validation team for ALI and Hyperion. Dr. Crawford was an Associate Editor of the TGARS

special issue on hyperspectral methods and is currently Associate Editor of the special issue on EO-1. She has served as a member of the IEEE GRS Administrative Committee since 1998, during which time she has been Director of Educational Activities and Vice President for Professional Activities.

Correspondence and offprint requests to: J. Ghosh, Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712, USA. E-mail: ghosh@ece.utexas.edu

A: Proof of Theorem 1

Theorem 1. The posterior probability $P(\omega|\mathbf{x})$ for any input $\mathbf{x}$ is the product of the posterior probabilities of all the internal classifiers along a unique path from the root node to the leaf node $n(\omega)$ containing the class ω, i.e.

$$P(\omega|\mathbf{x}) = \prod_{\ell=0}^{\mathcal{D}(\omega)-1} P(\Omega_{n(\omega)}^{(\ell+1)}|\mathbf{x}, \Omega_{n(\omega)}^{(\ell)}), \tag{18}$$

where $\mathcal{D}(\omega)$ is the depth of $n(\omega)$ (depth of the root node is 0), $\Omega_n^{(\ell)}$ is the meta-class at depth ℓ in the path from the root node to $n(\omega)$, such that $\Omega_{n(\omega)}^{(\mathcal{D}(\omega))} = \{\omega\}$ and $\Omega_{n(\omega)}^{(0)} = \Omega_1 =$ root node.

Proof: Applying Bayes theorem to each internal node:

$$P(\Omega_{n(\omega)}^{(\ell+1)}|\mathbf{x}, \Omega_{n(\omega)}^{(\ell)}) = \frac{p(\mathbf{x}|\Omega_{n(\omega)}^{(\ell+1)}, \Omega_{n(\omega)}^{(\ell)}\ P(\Omega_{n(\omega)}^{(\ell+1)}|\Omega_{n(\omega)}^{(\ell)})}{p(\mathbf{x}|\Omega_{n(\omega)}^{(\ell)})}, \tag{19}$$

$\forall \ell = 0, \ldots, \mathcal{D}(\omega) - 1.$

Because the probability density functions are computed at each internal node for the two child nodes locally,

$$p(\mathbf{x}|\Omega_{n(\omega)}^{(\ell+1)}, \Omega_{n(\omega)}^{(\ell)} = p(\mathbf{x}|\Omega_{n(\omega)}^{(\ell+1)}). \tag{20}$$

Also, the conditional prior of a child node meta-class, given its parent node meta-class, i.e. $P(\Omega_{n(\omega)}^{(\ell+1)}|\Omega_{n(\omega)}^{(\ell)})$ is:

$$P(\Omega_{n(\omega)}^{(\ell+1)}|\Omega_{n(\omega)}^{(\ell)}) = \frac{P(\Omega_{n(\omega)}^{(\ell+1)})}{P(\Omega_{n(\omega)}^{(\ell)})} = \frac{\Sigma_{\rho\in\Omega_{n(\omega)}^{(\ell+1)}}\ P(\rho)}{\Sigma_{\rho\in\Omega_{n(\omega)}^{(\ell)}}\ P(\rho)}. \tag{21}$$

Applying the simplifications (20) and (21) in (19), this is then

$$P(\Omega_{n(\omega)}^{(\ell+1)}|\mathbf{x}, \Omega_{n(\omega)}^{(\ell)}) = \frac{p(\mathbf{x}|\Omega_{n(\omega)}^{(\ell+1)})\ P(\Omega_{n(\omega)}^{(\ell+1)})}{p(\mathbf{x}|\Omega_{n(\omega)}^{(\ell)})\ P(\Omega_{n(\omega)}^{(\ell)})}. \tag{22}$$

Substituting (22), the product on the right-hand side of (18) becomes

$$\prod_{\ell=0}^{D(\omega)-1} \left(\frac{p(\mathbf{x}|\Omega_{n(\omega)}^{(\ell+1)})}{p(\mathbf{x}|\Omega_{n(\omega)}^{(\ell)})}\right) \prod_{\ell=0}^{\mathcal{D}(\omega)-1} \left(\frac{P(\Omega_{n(\omega)}^{(\ell+1)})}{P(\Omega_{n(\omega)}^{(\ell)})}\right) \tag{23}$$

Cancelling common factors in the numerators and denominators, the expression in (23) reduces to

$$\left(\frac{p(\mathbf{x}|\Omega_{n(\omega)}^{(\mathcal{D}(\omega))})}{p(\mathbf{x}|\Omega_{n(\omega)}^{(0)})}\right) \left(\frac{P(\Omega_{n(\omega)}^{(\mathcal{D}(\omega))})}{P(\Omega_{n(\omega)}^{(0)})}\right) \tag{24}$$

Since $\Omega_{n(\omega)}^{(\mathcal{D}(\omega))}$ is the leaf node containing only class ω,

$$\mathrm{p}(\mathbf{x}|\Omega_{n(\omega)}^{(\mathcal{D}(\omega))}) \equiv \mathrm{p}(\mathbf{x}|\omega) \tag{25}$$

and

$$P(\Omega_{n(\omega)}^{(\mathcal{D}(\omega))}) \equiv P(\omega). \tag{26}$$

Also, since $\Omega_{n(\omega)}^{(0)}$ is the meta-class at the root node containing all the classes,

$$p(\mathbf{x}|\Omega_{n(\omega)}^{(0)}) \equiv p(\mathbf{x}) \tag{27}$$

and

$$P(\Omega_{n(\omega)}^{(0)}) = \sum_{\rho\in\mathbf{\Omega}} P(\rho) = 1. \tag{28}$$

Applying (25) through (28), the expression in (24), which is the product term in the right-hand side of (18), reduces to:

$$\prod_{\ell=0}^{\mathcal{D}(\omega)-1} P(\Omega_{n(\omega)}^{(\ell+1)}|\mathbf{x}, \Omega_{n(\omega)}^{(\ell)}) = \frac{p(\mathbf{x}|\omega)\ P(\omega)}{p(\mathbf{x})} = P(\omega|\mathbf{x}) \tag{29}$$

QED.