2

Automatic Discovery of Class Hierarchies via Output Space Decomposition

Joydeep Ghosh, Shailesh Kumar and Melba M. Crawford

Summary. Many complex pattern classification problems involve high-dimensional inputs as well as a large number of classes. In this chapter, we present a modular learning framework called the Binary Hierarchical Classifier (BHC) that takes a coarse-to-fine approach to dealing with a large number of output classes. BHC decomposes a C-class problem into a set of $C-1$ two-(meta)class problems, arranged in a binary tree with C leaf nodes and $C-1$ internal nodes. Each internal node is comprised of a feature extractor and a classifier that discriminates between the two meta-classes represented by its two children. Both *bottom-up* and *top-down* approaches for building such a BHC are presented in this chapter. The Bottom-up Binary Hierarchical Classifier (BU-BHC) is built by applying agglomerative clustering to the set of C classes. The Top-down Binary Hierarchical Classifier (TD-BHC) is built by recursively partitioning a set of classes at any internal node into two disjoint groups or meta-classes. The coupled problems of finding a good partition and of searching for a linear feature extractor that best discriminates the two resulting meta-classes are solved simultaneously at each stage of the recursive algorithm. The hierarchical, multistage classification approach taken by the BHC also helps with dealing with high-dimensional data, since simpler feature spaces are often adequate for solving the two-(meta)class problems. In addition, it leads to the discovery of useful domain knowledge such as class hierarchies or ontologies, and results in more interpretable results.

2.1 Introduction

A classification problem involves identifying a set of objects, each represented in a suitable common input space, using one or more class labels taken from a pre-determined set of possible labels. Thus it may be described as a four-tuple: $(\mathcal{I}, \mathbf{\Omega}, P_{X \times \Omega}, \mathcal{X})$, where $\mathcal{I}$ is the *input space*, in which the raw data is available (e.g. the image of a character), $\mathbf{\Omega}$ is the *output space*, comprised of all the class labels that can be assigned to an input pattern (e.g. the set of 26 alphabetic characters in English), $P_{X \times \Omega}$ is the *unknown* joint probability density function over random variables $X \in \mathcal{I}$ and $\Omega \in \mathbf{\Omega}$, and $\mathcal{X} \subset \mathcal{I} \times \mathbf{\Omega}$ is the training set sampled from the distribution $P_{X \times \Omega}$. The goal is to determine

the relationship between the input and output spaces, a full specification of which is given by modeling the joint probability density function $P_{X \times \Omega}$.

Complexity in real-world classification problems can arise from multiple causes. First, the objects (and their representation) may themselves be complex, e.g. XML trees, protein sequences with 3-D folding geometry, variable length sequences, etc. [18]. Second, the data may be very noisy, the classes may have significant overlap and the optimal decision boundaries may be highly nonlinear. In this chapter we concentrate simultaneously on complexity due to high-dimensional inputs and a large number of class labels that can be potentially assigned to any input. Recognition of characters from the English alphabet ($C = 26$ classes) based on a (say) 64×64 binary input image and labeling of a piece of land into one of 10–12 land-cover types based on 100+ dimensional hyperspectral signatures are two examples that exhibit such complex characteristics.

There are two main approaches to simplifying such problems:

- **Feature extraction**: A feature extraction process transforms the input space, $\mathcal{I}$, into a lower-dimensional *feature space*, $\mathcal{F}$, in which discrimination among the classes $\boldsymbol{\Omega}$ is high. It is particularly helpful given finite training data in a high-dimensional input space, as it can alleviate fundamental problems arising from the *curse of dimensionality* [2, 15]. Both domain knowledge and statistical methods can be used for feature extraction [4, 9, 12, 16, 27, 33]. Feature selection is a specific case of linear feature extraction [33].
- **Modular learning**: Based on the divide-and-conquer precept that "learning a large number of simple local concepts is both easier and more useful than learning a single complex global concept" [30], a variety of modular learning architectures have been proposed by the pattern recognition and computational intelligence communities [28, 36, 47]. In particular, multi-classifier systems develop a set of M classifiers instead of one, and subsequently combine the individual solutions in a suitable way to address the overall problem. In several such architectures, each individual classifier addresses a simpler problem. For example, it may specialize in only part of the feature space as in the mixture of experts framework [26, 41]. Alternatively, a simpler input space may effectively be created per classifier by sampling/re-weighting (as in bagging and boosting), using one module for each data source [48]; different feature subsets for different classes (input decimation) [49], etc. Advantages of modular learning include the ease and efficiency in learning, scalability, interpretability, and transparency [1, 21, 36, 38, 42].

This chapter focuses on yet another type of modularity which is possible for multi-class problems, namely, the decomposition of a C-class problem into a set of binary problems. Such decompositions have attracted much interest recently because of the popularity of certain powerful binary classifiers, most notably the support vector machine (SVM), which was originally formulated

for binary dichotomies [50]. Although several extensions of SVMs to multi-class problems have been subsequently suggested (see papers referred to in [25]), the results of [25] show that such direct approaches are inferior to decomposing the multiclass problem into several binary classification problems, each addressed by a binary SVM.

Over the years, several approaches to decomposing the output space have been proposed. The most popular approaches, described in more detail in Section 2.2, are: (i) solving C "one-versus-rest" two-class problems; (ii) examining $\binom{C}{2}$ pairwise classifications; (iii) sequentially looking for or eliminating a single class at a time and (iv) applying error correcting output codes [10]. These approaches have been met with varying degrees of success. For the moment, we note that they typically do not take into account the natural affinities among the classes, or simultaneously determine simpler feature spaces that are tailored for specific output decompositions.

In this chapter, we propose an alternative approach to problem decomposition in output space that involves building a BINARY HIERARCHICAL CLASSIFIER (BHC) in which a C-class problem is addressed using a set of $M = C-1$ two-(meta)class feature extractor/classifier modules. These modules are arranged to form the $C-1$ internal nodes of a binary tree with C leaf nodes, one for each class. At each internal node, the partitioning of the parent meta-class into two child meta-classes is done simultaneously with the identification of an appropriately small but discriminating feature space for the corresponding classification problem. This is unlike the commonly used decision trees in which there may be several leaf nodes per class and the partitionings are explicitly done only in the *input* space. Instead the BHC can be considered as an example of a *coarse-to-fine* approach to multi-class problems. In earlier pattern recognition literature, several multistage approaches, including hierarchical ones were considered in which classes were progressively eliminated for an unlabelled sample [8, 43]. One of the goals of this work is to motivate the reader to reconsider such approaches as they often provide valuable domain information as a side-effect.

In addition to reducing the number of binary classifiers from $\mathcal{O}(C^2)$ in the pairwise classifier framework to $\mathcal{O}(C)$, the BHC framework also generates a class taxonomy that often provides useful domain knowledge. Indeed, the hierarchical problem decomposition viewpoint was motivated by the observation that many real-world classification problems have inherent taxonomies associated with them. Examples of such hierarchically structured classes can be found in domains as diverse as Biology, where all life forms are arranged in a multilevel taxonomy, and Internet portals such as YAHOO!, where all articles are arranged in a hierarchical fashion for ease of navigation and organization.

In fact, the BHC was developed by us while attempting to produce effective solutions to classification of land cover from remotely sensed hyperspectral imagery. Land covers have natural hierarchies and inter-class affinities, which the BHC was able to automatically infer and exploit. Figure 2.1 shows an example of a simple two-level hierarchy of various land-cover types
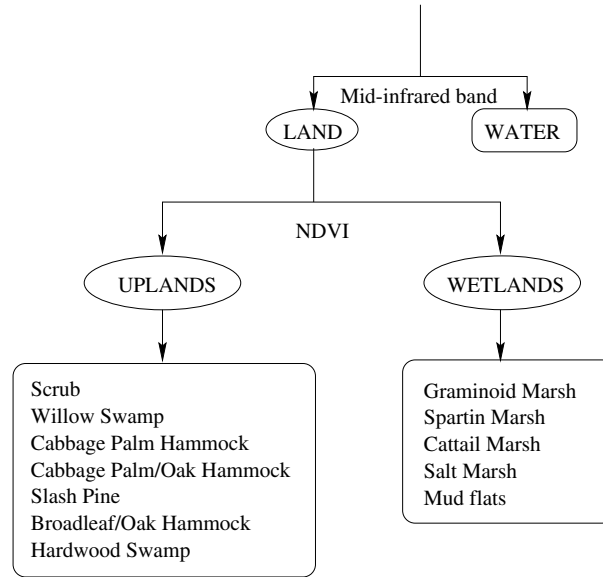
Fig. 2.1. A simple two-level hierarchy for a site with one WATER class and 12 LAND classes divided into seven UPLANDS and five WETLANDS meta-classes. The land versus water distinction is made by the response in the mid-infrared band while the distinction between uplands and wetlands is made using the Normalized Difference Vegetation Index (NDVI).

in the Bolivar peninsula [7]. In this example, 13 original (base) classes are first decomposed into two groups, LAND and WATER. WATER and LAND "meta-classes" can be readily separated based on the pixel responses in the mid-infrared frequency bands. WATER is one of the 13 base classes, while the LAND meta-class comprises 12 classes and is thus further partitioned into UPLANDS and WETLANDS meta-classes comprised of seven and five base classes respectively. The distinction between the UPLANDS and WETLANDS is made using the Normalized Difference Vegetation Index (NDVI) [45]. Instead of solving a 13-class problem, the hierarchy shown in Figure 2.1 can be used to first solve a binary problem (separating WATER from LAND), and then solve another binary problem to separate UPLANDS from WET-LANDS. Note that both the feature space as well as the output space of the two problems are different. The seven-class problem of discriminating among the UPLANDS classes and the five-class problem of discriminating among the WETLANDS classes can be further addressed in appropriate feature spaces using appropriate classifiers. Thus, a 13-class problem is decomposed using an *existing hierarchy* into simpler classification problems in terms of their output spaces.

Section 2.2 summarizes existing approaches to solving multi-class problems through output space decomposition. The BHC framework is formally

defined in Section 2.3. The Bottom-up Binary Hierarchical Classifier (BU-BHC) algorithm for building the BHC using ideas from agglomerative clustering [11] in a bottom-up fashion is described in Section 2.4. The Top-down Binary Hierarchical Classifier (TD-BHC) algorithm for building the BHC using ideas from our GAMLS framework [30] in a top-down approach is described in Section 2.5. Section 2.6 discusses both hard and soft ways of combining the results from individual binary classifiers to solve the original multi-class problem, for both top-down and bottom-up approaches. An experimental evaluation of the BHC framework over several large classification tasks follows in Section 2.7, and several class hierarchies extracted from the data are displayed in Section 2.8.

2.2 Background: Solving Multi-Class Problems

In this section we summarize and compare four main types of approaches that have been developed over the years to address multi-class problems using binary classifiers.

2.2.1 One-versus-rest

The traditional approach to multiclass problems is to develop C classifiers, each focussed on distinguishing one particular class from the rest. Often this is achieved by developing a discriminant function for each of the C classes. A new data point is assigned the class label corresponding to the discriminant function that gives the highest value for that data point. For example, in Nilsson's classic linear machine [37], the discriminant functions are linear, so the decision boundaries are constrained to be hyperplanes that intersect at a point. This is an example of the *discriminant analysis* family of algorithms, that includes *Quadratic Discriminant Analysis* [22, 34], *Regularized Discriminant Analysis* [13], and *Kernel Discriminant Analysis* [6, 20]. The essential difference among different discriminant analysis methods is the nature and bias of the discriminant function used.

2.2.2 Pairwise classification

Also known as round robin classification [17], these approaches learn one classifier for each pair of classes (employing a total of $\binom{C}{2}$ classifiers in the process) and then combine the outputs of these classifiers in a variety of ways to determine the final class label. This approach has been investigated by several researchers [14, 23, 39, 46]. Typically the binary classifiers are developed and examined in parallel, a notable exception being the efficient DAG-structured ordering given in [39]. A straightforward way of finding the winning class is through a simple voting scheme used for example in [14], which evaluates

pairwise classification for two versions of CART and for the nearest neighbor rule. Alternatively, if the individual classifiers provide good estimates of the two-class posterior probabilities, then these estimates can be combined using an iterative hill-climbing approach suggested by [23].

Our first attempts at output space decomposition [7, 31] involved applying a pairwise classifier framework for land-cover prediction problems involving hyperspectral data. Class-pair-specific feature extraction was used to obtain superior classification accuracies. It also provided important domain knowledge with regard to what features were more useful for discriminating specific pairs of classes. While such a modular-learning approach for decomposing a C-class problem is attractive for a number of reasons including focussed feature extraction, interpretability of results and automatic discovery of domain knowledge, the fact that it requires $\mathcal{O}(C^2)$ pairwise classifiers might make it less attractive for problems involving a large number of classes. Further, the combiner that integrates the results of all the $\binom{C}{2}$ classifiers must resolve the couplings among these outputs that might increase with the number of classes.

2.2.3 Error correcting output codes (ECOC)

Inspired by distributed output representations in biological systems, as well as by robust data communication ideas, ECOC is one of the most innovative and popular approaches to have emerged recently to deal with multi-class problems [10]. A C-class problem is encoded as $\bar{C}$ binary problems. For each binary problem, one subset of the classes serves as the positive class (target $= 1$) while the rest form the negative class (target $= 0$). As a consequence, each original class is encoded into a $\bar{C}$-dimensional binary vector. The $C \times \bar{C}$ binary matrix is called the coding matrix. A given test input is labelled as belonging to the class whose code is closest to the code formed by the outputs of the $\bar{C}$ classifiers in response to that input.

2.2.4 Sequential methods

These approaches impose an ordering among the classes, and the classifiers are developed in sequence rather than in parallel. For example, one can first discriminate between class "1" and the rest. Then for data classified as "rest", a second classifier is designed to separate class "2" from the other remaining classes, and so on. Problem decomposition in the output space can also be accomplished implicitly by having C classifiers, each trying to solve the complete C-class problem, but with each classifier using input features most correlated with only one of the classes. This idea was used in [49] for creating an ensemble of classifiers, each using different *input decimations*. This method not only reduces the correlation among individual classifiers in an ensemble, but also reduces the dimensionality of the input space for classification problems. Significant improvements in misclassification error together with reductions

in the number of features used were obtained on various public-domain data sets using this approach.

2.2.5 Comments and comparisons

A common characteristic of the approaches described above is that they do not take into account the underlying affinities among the individual classes (for example, how close or separated they are) while deciding on class selection/grouping for binary classification. Both one-versus-rest and pairwise methods treat each class the same way while, in ECOC, design of the code matrix is based on the properties of this matrix rather than the classes they represent. That is why it is helpful to have a strong base learner when applying ECOC since some of the groupings may lead to complicated decision boundaries. In contrast, the groupings in BHC are determined by the properties of the class distributions. Not being agnostic to class affinities helps us in determining natural groupings that facilitate both the discrimination process and the interpretation of results.

Three noteworthy studies have emerged recently that compare the three major approaches. Furnkranz [17] shows that the $\binom{C}{2}$ learning problems of pairwise classification can be learned more efficiently than the C problems of the one-versus-rest technique. His analysis is independent of the base learning algorithm. He also observes that both these approaches are more efficient than ECOC. A large number of empirical results are shown using RIPPER and C5.0 as base classifiers. The BHC uses only $C-1$ classifiers, similar to one-versus-rest, but since the class groupings are based on affinities, the binary classifications are simpler in general. Hence BHCs do not compromise much on efficiency in the process of reducing the number of classifiers needed. Hsu and Lin [25] did a detailed study comparing one-versus-rest and pairwise classification, both using the SVM as base classifier, to two approaches for directly generalizing the SVM algorithm to multi-class problems. The pairwise method performed best both in terms of accuracy and training time. One-versus-rest was second and both methods were better than the direct generalizations of SVM. Finally, a recent intriguing study [44] shows that no one of these methods performs significantly better than any other as far as test errors are concerned. The study is carefully done, but it is not clear whether the results are affected by the choice of SVMs with Gaussian kernels as the base classifiers.

2.3 The BINARY HIERARCHICAL CLASSIFIER Framework

Definition 1 A binary hierarchical classifier for a C-class problem $\mathcal{P}(\mathcal{I}, \mathbf{\Omega}, P_{X \times \Omega}, \mathcal{X})$ is defined as an ensemble of $C-1$ two-(meta)class problems, arranged as a binary tree $\mathcal{T}(\mathbf{\Omega}_1)$, $(\mathbf{\Omega}_1 = \mathbf{\Omega})$ recursively defined as follows:

$$
\mathcal{T}(\boldsymbol{\Omega}_n) = \begin{cases} \left[\mathcal{P}(\mathcal{F}_n, \tilde{\boldsymbol{\Omega}}_n, P_{Y_n \times \tilde{\Omega}_n}, \mathcal{Y}_n), \mathcal{T}(\boldsymbol{\Omega}_{2n}), \mathcal{T}(\boldsymbol{\Omega}_{2n+1})\right] & \text{if } |\boldsymbol{\Omega}_n| > 1 \\ \boldsymbol{\Omega}_n & \text{if } |\boldsymbol{\Omega}_n| = 1, \end{cases}
$$
$$(2.1)$$

in which each internal node n (i.e. $n : |\boldsymbol{\Omega}_n| > 1$) has an associated two-(meta)class problem:

$$
\mathcal{P}(\mathcal{F}_n, \tilde{\boldsymbol{\Omega}}_n, P_{Y_n \times \tilde{\Omega}_n}, \mathcal{Y}_n) \tag{2.2}
$$

where n is the index of a node in the tree. For each node n, $\boldsymbol{\Omega}_n$ is a set of classes in the associated meta-class. For each *internal* node n, $\{2n, 2n+1\}$ are indices of the left and right children, $\tilde{\boldsymbol{\Omega}}_n = \{\boldsymbol{\Omega}_{2n}, \boldsymbol{\Omega}_{2n+1}\}$, $\mathcal{F}_n$ is the feature space for the binary problem, Y_n are random variables in $\mathcal{F}_n$, and $\tilde{\Omega}_n$ are random variables in $\tilde{\boldsymbol{\Omega}}_n$. Further, each internal node n is comprised of meta-class feature extractors $\psi_n : \mathcal{I} \to \mathcal{F}_n$, such that discrimination between $\boldsymbol{\Omega}_{2n}$ and $\boldsymbol{\Omega}_{2n+1}$ is high in $\mathcal{F}_n$, and meta-class classifiers ϕ_n for classes $\tilde{\boldsymbol{\Omega}}_n$. Finally, a tree combiner Ξ integrates the outputs of all the internal node classifiers $\{\phi_n\}$ into a single output. The classifiers ϕ_n can be *hard classifiers* defined by the mapping $\phi_n^H : \mathcal{F}_n \to \tilde{\boldsymbol{\Omega}}_n$, or *soft classifiers* given by the mapping $\phi_n^S : \mathcal{F}_n \to P_n(\tilde{\Omega}_n = \boldsymbol{\Omega}_{2n}|Y_n)$. (Note that $P_n(\tilde{\Omega}_n = \boldsymbol{\Omega}_{2n+1}|\mathbf{y}_n(\mathbf{x}))$ is simply $1 - P_n(\tilde{\Omega}_n = \boldsymbol{\Omega}_{2n}|\mathbf{y}_n(\mathbf{x}))$.)
Correspondingly the combiner Ξ can be a *hard combiner* $\Xi_H : \{\tilde{\Omega}_n\}_{n:|\boldsymbol{\Omega}_n|>1} \to \boldsymbol{\Omega}$, where inputs to Ξ_H are the $C-1$ (meta)class labels and output is one of the C class labels in $\boldsymbol{\Omega}$, or a *soft combiner* $\Xi_S : \{P_n(\tilde{\Omega}_n|Y_n)\}_{n:|\boldsymbol{\Omega}_n|>1} \to \{P(\omega|X)\}_{\omega \in \boldsymbol{\Omega}}$, where inputs to Ξ_S are the meta-class posterior probabilities generated by the $C-1$ classifiers.

Figure 2.2 shows an example of a five-class BHC with four internal nodes and five leaf nodes. In general, the BHC tree $\mathcal{T}(\boldsymbol{\Omega})$ contains $C = |\boldsymbol{\Omega}|$ leaf nodes and $C-1$ internal nodes. Each internal node n has its own feature extractor and classifier that discriminates the two meta-classes $\boldsymbol{\Omega}_{2n}$ and $\boldsymbol{\Omega}_{2n+1}$. The decomposition of the set of classes $\boldsymbol{\Omega}_n$ into two disjoint subsets $\boldsymbol{\Omega}_{2n}$ and $\boldsymbol{\Omega}_{2n+1}$ is an NP problem with $\mathcal{O}(2^{|\boldsymbol{\Omega}_n|})$ possible alternatives. Further, the feature space $\mathcal{F}_n$ depends on the decomposition of $\boldsymbol{\Omega}_n$. Hence the two coupled problems of finding the best possible decomposition of $\boldsymbol{\Omega}_n$ and the best feature space that discriminates the two resulting meta-classes must be solved simultaneously. The bottom up and top-down approaches of building such binary hierarchical classifiers are described next.

2.4 Bottom-up BHC

The Bottom-up Binary Hierarchical classifier (BU-BHC) algorithm is analogous to hierarchical agglomerative clustering [11]. Instead of merging data points or clusters at each stage,] classes or meta-classes are merged in the BU-BHC algorithm. Starting from the set of C meta-classes $\Pi_C = \{\boldsymbol{\Omega}^{(c)}\}_{c=1}^C$, where $\boldsymbol{\Omega}^{(c)} = \{\omega_c\}$, a sequence $\Pi_C \to \Pi_{C-1} \to \dots \Pi_2 \to \Pi_1$ with
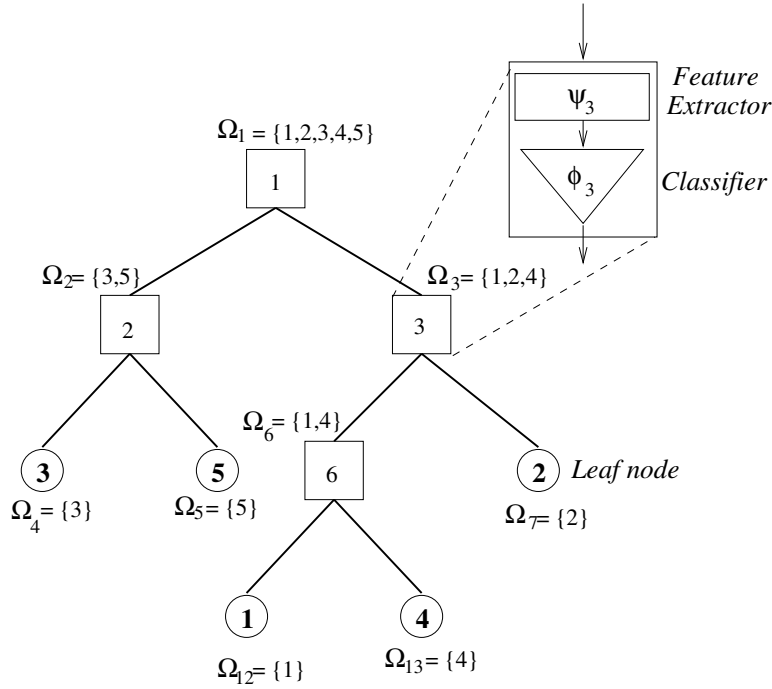
Fig. 2.2. An example of a BINARY HIERARCHICAL CLASSIFIER for a $C = 5$ class problem with four internal nodes and five leaf nodes. Each internal node n comprises a feature extractor ψ_n and a classifier ϕ_n. Each node n is associated with a set of classes $\mathbf{\Omega}_n$. The left and right children of internal node n are indexed $2n$ and $2n+1$, respectively.

an associated decreasing number of meta-classes is generated by merging two meta-classes Ω_α and Ω_β in Π_K to obtain the set Π_{K-1}.

In order to decide which of the K meta-classes in Π_K are to be merged to obtain Π_{K-1}, a "distance" between every pair of meta-classes, $\vartheta(\Omega_\alpha, \Omega_\beta)$ is defined as the *separation between the two meta-classes in the most discriminatory feature space* $\mathcal{F}(\Omega_\alpha, \Omega_\beta)$. Any suitable family of feature extractors can be used to quantify the distance between two meta-classes. In this chapter, since we are largely concerned with numeric data, two variants of the Fisher discriminant based linear feature extractors are proposed: FISHER(1), in which a one-dimensional projection of the D-dimensional input space is sought for the two-meta class problem, and FISHER(m), in which an m-dimensional feature space where $m = \min\{D, |\Omega_\alpha| + |\Omega_\beta| - 1\}$ is sought.

2.4.1 FISHER(1) Feature Extraction

The dimensionality of the Fisher projection space for a C-class problem with a D-dimensional input space is $\min\{D, C-1\}$. At each internal node in the

BHC, a two-class problem is solved, and hence only a one-dimensional feature space can be obtained for discriminating these two meta-classes. The distance function and the feature space obtained by the FISHER(1) feature extractor for the two meta-classes Ω_α and Ω_β are defined in this section.

Let $\{\mu^\rho \in \Re^{D \times 1}, \rho \in \{\alpha, \beta\}\}$ and $\{\Sigma^\rho \in \Re^{D \times D}, \rho \in \{\alpha, \beta\}\}$ be the means and covariances of the two meta-classes and let $\{P(\Omega_\rho), \rho \in \{\alpha, \beta\}\}$ be their priors. The statistics of meta-class Ω_ρ can be defined in terms of the estimated mean vectors, $\{\hat{\mu}_\omega \in \Re^{D \times 1}, \omega \in \Omega_\rho\}$, covariance matrices, $\{\hat{\Sigma}_\omega \in \Re^{D \times D}, \omega \in \Omega_\rho\}$ and class priors $\{\hat{P}(\omega), \omega \in \Omega_\rho\}$ as follows:

$$\hat{P}(\Omega_\rho) = \sum_{\omega \in \Omega_\rho} \hat{P}(\omega) = \frac{\sum_{\omega \in \Omega_\rho} |X_\omega|}{\sum_{\gamma \in \Omega} |\mathcal{X}_\gamma|}, \quad \rho \in \{\alpha, \beta\}, \tag{2.3}$$

$$\hat{\mu}^\rho = \frac{\sum_{\omega \in \Omega_\rho} \sum_{\mathbf{x} \in \mathcal{X}_\omega} \mathbf{x}}{\sum_{\omega \in \Omega_\rho} |\mathcal{X}_\omega|} = \frac{\sum_{\omega \in \Omega_\rho} \hat{P}(\omega)\hat{\mu}_\omega}{\sum_{\omega \in \Omega_\rho} \hat{P}(\omega)}, \quad \rho \in \{\alpha, \beta\}, \tag{2.4}$$

$$\begin{aligned}\hat{\Sigma}^\rho &= \frac{\sum_{\omega \in \Omega_\rho} \sum_{\mathbf{x} \in \mathcal{X}_\omega} (\mathbf{x} - \hat{\mu}^\rho)(\mathbf{x} - \hat{\mu}^\rho)^T}{\sum_{\omega \in \Omega_\rho} |\mathcal{X}_\omega|} \\ &= \frac{\sum_{\omega \in \Omega_\rho} \hat{P}(\omega)\left[\hat{\Sigma}_\omega + (\hat{\mu}^\rho - \hat{\mu}_\omega)(\hat{\mu}^\rho - \hat{\mu}_\omega)^T\right]}{\sum_{\omega \in \Omega_\rho} \hat{P}(\omega)}\end{aligned} \tag{2.5}$$

The Fisher discriminant depends on the $D \times D$ symmetric within class covariance matrix $\mathbf{W}_{\alpha,\beta}$ given by:[1]

$$\mathbf{W}_{\alpha,\beta} = P(\Omega_\alpha)\Sigma^\alpha + P(\Omega_\beta)\Sigma^\beta, \tag{2.6}$$

and the $D \times D$, rank 1, between class covariance matrix $\mathbf{B}_{\alpha,\beta}$ given by:

$$\mathbf{B}_{\alpha,\beta} = P(\Omega_\alpha)P(\Omega_\beta)(\mu^\alpha - \mu^\beta)(\mu^\alpha - \mu^\beta)^T. \tag{2.7}$$

The corresponding one-dimensional Fisher projection is given by:

$$\mathbf{v}_{\alpha\beta} = \arg \max_{\mathbf{v} \in \Re^{D \times 1}} \frac{\mathbf{v}^T \mathbf{B}_{\alpha,\beta} \mathbf{v}}{\mathbf{v}^T \mathbf{W}_{\alpha,\beta} \mathbf{v}} \propto \mathbf{W}_{\alpha,\beta}^{-1} \left(\mu^\alpha - \mu^\beta\right). \tag{2.8}$$

Thus, the FISHER(1) feature extractor $\psi_{fisher}^{(1)}(X|\Omega_\alpha, \Omega_\beta) = \mathbf{v}_{\alpha\beta}^T \mathbf{x}$, where $\mathbf{x} \in \Re^{D \times 1}$ and $y \in \Re$ is a one-dimensional feature. The distance between the two meta-classes Ω_α and Ω_β is the FISHER(1) discriminant along the Fisher projection $\mathbf{v}_{\alpha\beta}$ of Equation (2.8).

2.4.2 FISHER(m) Feature Extraction

The basic assumption in Fisher's discriminant is that the two classes are unimodal. Even if this assumption is true for individual classes, it is not true

---

[1]Substituting estimated parameters for expected ones (e.g. $\hat{P} \equiv P$, $\hat{\mu} \equiv \mu$, and $\hat{\Sigma} \equiv \Sigma$).

for meta-classes comprised of two or more classes. Moreover, as the number of classes in the meta-classes Ω_α and Ω_β increases, the dimensionality of the feature space should also increase to compensate for the more complex decision boundaries between the two meta-classes. In the FISHER(1) feature extractor, irrespective of the sizes of the two meta-classes (in terms of the number of original classes), the Fisher projection is always one-dimensional because the rank of the between-class covariance matrix $\mathbf{B}_{\alpha,\beta}$ defined in Equation (2.7) is 1.

To alleviate this problem, we replace $\mathbf{B}_{\alpha,\beta}$ by a pairwise between-class covariance matrix $\tilde{\mathbf{B}}_{\alpha,\beta}$ that is defined in terms of the between-class covariances $\mathbf{B}_{\omega,\omega'} = P(\omega)P(\omega')(\mu_\omega - \mu_{\omega'})(\mu_\omega - \mu_{\omega'})^T, \quad \forall (\omega, \omega' \in \Omega_\alpha \times \Omega_\beta$ as follows:

$$\tilde{\mathbf{B}}_{\alpha,\beta} = \sum_{\omega \in \Omega_\alpha} \sum_{\omega' \in \Omega_\beta} P(\omega)P(\omega')(\mu_\omega - \mu_{\omega'})(\mu_\omega - \mu_{\omega'})^T = \sum_{\omega \in \Omega_\alpha} \sum_{\omega' \in \Omega_\beta} \mathbf{B}_{\omega,\omega'}.$$

$$(2.9)$$

The rank of $\tilde{\mathbf{B}}_{\alpha,\beta}$ is $m_{\alpha\beta} = \min\{D, |\Omega_\alpha| + |\Omega_\beta| - 1\}$. The within-class covariance matrix for FISHER(m) is the same as in Equation (2.6). The Fisher projection matrix $\mathbf{V}_{\alpha\beta} \in \Re^{D \times m_{\alpha\beta}}$ for the FISHER(m) feature extractor is given by:

$$\mathbf{V}_{\alpha\beta} = \arg \max_{\mathbf{V} \in \Re^{D \times m_{\alpha\beta}}} \operatorname{tr}\left\{ \left(\mathbf{V}^T \mathbf{W}_{\alpha,\beta} \mathbf{V}\right)^{-1} \left(\mathbf{V}^T \mathbf{B}_{\alpha,\beta} \mathbf{V}\right) \right\}. \qquad (2.10)$$

The optimal solution is the first $m_{\alpha\beta}$ eigenvectors of $\left(\mathbf{W}_{\alpha,\beta}^{-1} \mathbf{B}_{\alpha,\beta}\right)$. Thus, the FISHER($m$) feature extractor $\psi_{fisher}^{(m)}(X|\Omega_\alpha, \Omega_\beta) = \mathbf{V}_{\alpha\beta}^T \mathbf{x}$, where $\mathbf{y} \in \Re^{m_{\alpha\beta} \times 1}$ is an $m_{\alpha\beta}$-dimensional feature vector. The distance between the two meta-classes Ω_α and Ω_β is the FISHER(m) discriminant along the projection $\mathbf{V}_{\alpha\beta}$ of Equation (2.10).

The dimensionality of the feature space using the FISHER(m) feature extractor depends on the size of the meta-classes that are merged. In terms of the notation of the BHC introduced in Definition 1, the dimensionality of the feature space $\mathcal{F}_n$ at the internal node n is $\min\{D, |\mathbf{\Omega}_n| - 1\}$. In particular, the dimensionality at the root node $n = 1$ is $\min\{D, |\mathbf{\Omega}_1| - 1\} = \min\{D, C - 1\}$. This is the same as the dimensionality of the Fisher projection of the original C-class problem, the key difference being that in BHC, a two meta-class problem is solved in this space instead of the C-class problem. The tradeoff between the reduction in the number of classes from C to two and the increase in the complexity of the two meta-classes determines the utility of such a feature space.

2.4.3 Merging the Meta-Classes

Let Ω_α and Ω_β be the two closest (in terms of the Fisher projected distances defined in Sections 2.4.1 and 2.4.2) classes that are merged to form the meta-class $\Omega_{\alpha\beta} = \operatorname{MERGE}(\Omega_\alpha, \Omega_\beta)$. The estimated mean vector $\hat{\mu}^{\alpha\beta} \in \Re^{D \times 1}$,

covariance matrix $\hat{\Sigma}^{\alpha\beta} \Re^{D \times D}$, and prior probability $\hat{P}(\Omega_{\alpha\beta})$ of the meta-class $\Omega_{\alpha\beta}$ are related to the means, covariances, and priors of the two merged meta-classes as follows:

$$\hat{P}(\Omega_{\alpha\beta}) = \sum_{\omega \in \Omega_{\alpha\beta}} \hat{P}(\omega) = \hat{P}(\Omega_\alpha) + \hat{P}(\Omega_\beta), \qquad (2.11)$$

$$\hat{\mu}^{\alpha\beta} = \frac{\sum_{\omega \in \Omega_{\alpha\beta}} \sum_{\mathbf{x} \in \mathcal{X}_\omega} \mathbf{x}}{\sum_{\omega \in \Omega_{\alpha\beta}} |\mathcal{X}_\omega|} = \frac{\hat{P}(\Omega_\alpha)\hat{\mu}^\alpha + \hat{P}(\Omega_\beta)\hat{\mu}^\beta}{\hat{P}(\Omega_\alpha) + \hat{P}(\Omega_\beta)}, \qquad (2.12)$$

$$\begin{aligned}
\hat{\Sigma}^{\alpha\beta} &= \frac{\sum_{\omega \in \Omega_{\alpha\beta}} \sum_{\mathbf{x} \in \mathcal{X}_\omega} (\mathbf{x} - \hat{\mu}^{\alpha\beta})(\mathbf{x} - \hat{\mu}^{\alpha\beta})^T}{\sum_{\omega \in \Omega_{\alpha\beta}} |\mathcal{X}_\omega|} \\
&= \frac{\sum_{\rho \in \{\alpha,\beta\}} \hat{P}(\Omega_\rho)\left[\hat{\Sigma}^\rho + (\hat{\mu}^\rho - \hat{\mu}^{\alpha\beta})(\hat{\mu}^\rho - \hat{\mu}^{\alpha\beta})^T\right]}{\hat{P}(\Omega_\alpha) + \hat{P}(\Omega_\beta)}.
\end{aligned} \qquad (2.13)$$

Once the mean and covariance of the new meta-class $\Omega_{\alpha\beta}$ are obtained, its distance from the remaining classes $\Omega_\gamma \in \Pi_K - \{\Omega_\alpha, \Omega_\beta\}$ is computed as follows. The within-class covariance $\mathbf{W}_{\alpha\beta,\gamma}$ is given by:[2,3]

$$\begin{aligned}
\mathbf{W}_{\alpha\beta,\gamma} &= P(\Omega_{\alpha\beta})\Sigma^{\alpha\beta} + P(\Omega_\gamma)\Sigma^\gamma \\
&= \tfrac{1}{2}\left[\mathbf{W}_{\alpha,\gamma} + \mathbf{W}_{\beta,\gamma} + \mathbf{W}_{\alpha,\beta}\right] + \frac{\mathbf{B}_{\alpha,\beta}}{P(\Omega_\alpha) + P(\Omega_\beta)}.
\end{aligned} \qquad (2.14)$$

Similarly, the between-class covariance $\mathbf{B}_{\alpha\beta,\gamma}$ for the FISHER(1) case is defined as:

$$\begin{aligned}
\mathbf{B}_{\alpha\beta,\gamma} &= P(\Omega_{\alpha\beta})P(\Omega_\gamma)\left(\mu^{\alpha\beta} - \mu^\gamma\right)\left(\mu^{\alpha\beta} - \mu^\gamma\right)^T \mathbf{B}_{\alpha\beta,\gamma} \\
&= \mathbf{B}_{\alpha,\gamma} + \mathbf{B}_{\beta,\gamma} - \frac{P(\Omega_\gamma)}{P(\Omega_\alpha) + P(\Omega_\beta)}\mathbf{B}_{\alpha,\beta}.
\end{aligned} \qquad (2.15)$$

Finally, the pairwise between-class covariance $\tilde{\mathbf{B}}_{\alpha\beta,\gamma}$ for FISHER(m) case is defined as:

$$\tilde{\mathbf{B}}_{\alpha\beta,\gamma} = \sum_{\omega \in \Omega_{\alpha\beta}} \sum_{\omega' \in \Omega_\gamma} P(\omega)P(\omega')\left(\mu_\omega - \mu_{\omega'}\right)\left(\mu_\omega - \mu_{\omega'}\right)^T = \tilde{\mathbf{B}}_{\alpha,\gamma} + \tilde{\mathbf{B}}_{\beta,\gamma}$$

$$(2.16)$$

The recursive updates of $\mathbf{W}_{\alpha\beta,\gamma}$, $\mathbf{B}_{\alpha\beta,\gamma}$ and $\tilde{\mathbf{B}}_{\alpha\beta,\gamma}$ can be used to efficiently compute the distance $\vartheta(\Omega_{\alpha\beta}, \Omega_\gamma)$ and continue to build the tree bottom-up efficiently.

2.5 Top-down BHC

The bottom-up BHC algorithm is $\mathcal{O}(C^2)$ as the distance between all pairs of classes must be computed at the very first stage. Each of the $C-1$ subsequent stages is $\mathcal{O}(C)$. For a large number of classes this might make the

---

[2]Substituting estimated parameters for expected ones (e.g. $\hat{P} \equiv P$, $\hat{\mu} \equiv \mu$, and $\hat{\Sigma} \equiv \Sigma$).

[3]See [29] for details of simplifications.

BU-BHC algorithm less attractive. In this section, we propose an alternate approach to building the BHC, i.e., the TOP-DOWN BINARY HIERARCHI-CAL CLASSIFIER (TD-BHC) algorithm. This algorithm is motivated by our GAMLS framework [30]. In TD-BHC, starting from a single meta-class set Π_1 at the root node comprising of all the C classes, an increasing sequence $\Pi_1 \rightarrow \Pi_2 \rightarrow \ldots \Pi_{C-1} \rightarrow \Pi_C$ of meta-classes is obtained. At each stage, Π_K, one of the meta-classes is partitioned into two disjoint subsets leading to Π_{K+1}. Using the notation introduced in Definition 1, the basic TD-BHC algorithm, BUILDTREE($\mathbf{\Omega}_n$), can be written as follows:

1. Partition $\mathbf{\Omega}_n$ into two meta-classes $(\mathbf{\Omega}_{2n}, \mathbf{\Omega}_{2n+1}) \leftarrow$ PARTITIONNODE($\mathbf{\Omega}_n$)
2. Recurse on each child:
   - if $|\mathbf{\Omega}_{2n}| > 1$ then BUILDTREE($\mathbf{\Omega}_{2n}$)
   - if $|\mathbf{\Omega}_{2n+1}| > 1$ then BUILDTREE($\mathbf{\Omega}_{2n+1}$)

The purpose of the PARTITIONNODE function is to find a partition of the set of classes $\mathbf{\Omega}_n$ into two disjoint subsets such that the discrimination between the two meta-classes $\mathbf{\Omega}_{2n}$ and $\mathbf{\Omega}_{2n+1}$ is high. The feature space that best discriminates between the two meta-classes is also discovered simultaneously. FISHER(1) and FISHER(m) are two examples of such feature extractors. The two problems of finding a partition, as well as the feature extractor that maximizes discrimination between the meta-classes obtained as a result of this partition, are coupled. These coupled problems are solved simultaneously using association and specialization ideas of the GAMLS framework [30].

2.5.1 The PARTITIONNODE Algorithm

When partitioning a set of classes into two meta-classes, initially each class is associated with both the meta-classes. The update of these associations and meta-class parameters is performed alternately while gradually decreasing the temperature, until a hard partitioning is achieved. The complete PARTITIONNODE algorithm which forms the basis of the TD-BHC algorithm is described in this section.

Let $\Omega = \mathbf{\Omega}_n$ be some meta-class at internal node n with $K = |\mathbf{\Omega}_n| > 2$ classes that needs to be partitioned into two meta-classes, $\Omega_\alpha = \mathbf{\Omega}_{2n}$ and $\Omega_\beta = \mathbf{\Omega}_{2n+1}$. The "association" $\mathbf{A} = [a_{\omega,\rho}]$ between class $\omega \in \Omega$ and meta-class Ω_ρ, $(\rho \in \{\alpha, \beta\})$ is interpreted as the posterior probability of ω belonging to Ω_ρ: $P(\Omega_\rho|\omega)$. The *completeness constraint* of GAMLS [30] implies that $P(\Omega_\alpha|\omega) + P(\Omega_\beta|\omega) = 1$, $\forall \omega \in \Omega$.

PARTITIONNODE(Ω)

1. **Initialize associations** $\{a_{\omega,\alpha} = P(\Omega_\alpha|\omega), \omega \in \Omega\}$ ($a_{\omega,\beta} = 1 - a_{\omega,\alpha}$):

$$P(\Omega_\alpha|\omega) = \begin{cases} 1 & \text{for some } \omega = \omega_{(1)} \in \Omega \\ 0.5 \ \forall \ \omega \in \Omega - \{\omega_{(1)}\} \end{cases} \qquad (2.17)$$

The association of one of the classes $\omega_{(1)} \in \Omega$ with the meta-class Ω_α is fixed to 1, while all other classes are associated equally with both the meta-classes. This deterministic, non-symmetric and unbiased association initialization is possible only because PARTITIONNODE seeks to divide Ω into two meta-classes only and not more. As a result of this initialization, the TD-BHC algorithm always yields the same partition for a given data set and learning parameters, irrespective of the choice of $\omega_{(1)}$. The temperature parameter T is initialized to 1 in this chapter, and then decayed geometrically, as indicated in Step 6 of the algorithm below. Although the partition is not affected by the choice of the class $\omega_{(1)}$, the class that is "farthest" (in terms of e.g. Bhattacharya distance) from the meta-class Ω should be chosen for faster convergence.

2. **Find the most discriminating feature space** $\mathcal{F}(\Omega_\alpha, \Omega_\beta)$: For the current set of "soft" meta-classes $(\Omega_\alpha, \Omega_\beta)$ defined in terms of the associations $\mathbf{A}$, the feature extractor $\psi(X|\mathbf{A}) : \mathcal{I} \to \mathcal{F}(\Omega_\alpha, \Omega_\beta)$ that maximally discriminates the two meta-classes is sought. This step depends on the the feature extractor used. Section 2.5.3 describes how the FISHER(1) and FISHER(m) feature extractors can be extended to *soft meta-classes*.

3. **Compute the mean log-likelihoods** of classes $\omega \in \Omega$ in the feature space $\mathcal{F}(\Omega_\alpha, \Omega_\beta)$:

$$\mathcal{L}(\omega|\Omega_\rho) = \frac{1}{N_\omega} \sum_{\mathbf{x} \in \mathcal{X}_\omega} \log p(\psi(\mathbf{x}|\mathbf{A})|\Omega_\rho), \quad \rho \in \{\alpha, \beta\}, \ \ \forall \, \omega \in \Omega, \quad (2.18)$$

where the PDF $p(\psi(\mathbf{x}|\mathbf{A})|\Omega_\rho)$ can be modeled using any distribution function. A single Gaussian per class is used in this chapter.

4. **Update the meta-class posteriors** by optimizing Gibb's free energy [30]:

$$a_{\omega,\alpha} = P(\Omega_\alpha|\omega) = \frac{\exp(\mathcal{L}(\omega|\Omega_\alpha)/T)}{\exp(\mathcal{L}(\omega|\Omega_\alpha)/T) + \exp(\mathcal{L}(\omega|\Omega_\beta)/T)}. \quad (2.19)$$

5. Repeat Steps 2 through 4 until the increase in Gibb's free energy is insignificant.

6. If $\left(\frac{1}{|\Omega|} \sum_{\omega \in \Omega} \mathcal{H}(\mathbf{a}_\omega) \right) < \theta_H$ (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.

As the temperature cools sufficiently and the entropy decreases to near zero ($\theta_{\mathcal{H}} = 0.01$ in our implementation), the associations or the posterior probabilities $\{P(\Omega_\alpha|\omega), \omega \in \Omega\}$ become close to 0 or 1. The meta-class $\Omega = \mathbf{\Omega}_n$ is then split as follows:

$$\begin{aligned}
\mathbf{\Omega}_{2n} &= \{\omega \in \mathbf{\Omega}_n | a_{\omega,\alpha} = P(\Omega_\alpha|\omega) > P(\Omega_\beta|\omega) = a_{\omega,\beta}\} \\
\mathbf{\Omega}_{2n+1} &= \{\omega \in \mathbf{\Omega}_n | a_{\omega,\beta} = P(\Omega_\beta|\omega) > P(\Omega_\alpha|\omega) = a_{\omega,\alpha}\}
\end{aligned}. \quad (2.20)$$

2.5.2 Soft Meta-Class Parameter Updates

For any set of associations $\mathbf{A}$, the estimates of the meta-class mean vectors $\{\hat{\mu}^\rho \in \Re^{D \times 1}, \rho \in \{\alpha, \beta\}\}$, the covariance matrices $\{\hat{\Sigma}^\rho \in \Re^{D \times D}, \rho \in \{\alpha, \beta\}\}$, and priors $\{\hat{P}(\Omega_\rho), \rho \in \{\alpha, \beta\}\}$ are updated using the mean vectors $\{\hat{\mu}_\omega \in \Re^{D \times 1}, \omega \in \Omega\}$, covariance matrices $\{\hat{\Sigma}_\omega \in \Re^{D \times D}, \omega \in \Omega\}$, and class priors $\{\hat{P}(\omega), \omega \in \Omega\}$, of the classes in Ω. Let $\mathcal{X}_\omega$ denote the training set comprising $N_\omega = |\mathcal{X}_\omega|$ examples of class ω. For any given associations or posterior probabilities $\mathbf{A} = \{a_{\omega, \rho} = P(\Omega_\rho | \omega), \rho \in \{\alpha, \beta\}, \omega \in \Omega\}$, the estimate of the mean is computed by $\hat{\mu}^\rho = \sum_{\omega \in \Omega} P(\omega | \Omega_\rho) \hat{\mu}_\omega$, $\rho \in \{\alpha, \beta\}$. The corresponding covariance is:

$$
\begin{aligned}
\hat{\Sigma}^\rho &= \sum_{\omega \in \Omega} \frac{P(\omega | \Omega_\rho)}{N_\omega} \left[\sum_{\mathbf{x} \in \mathcal{X}_\omega} (\mathbf{x} - \hat{\mu}^\rho)(\mathbf{x} - \hat{\mu}^\rho)^T \right] \\
&= \sum_{\omega \in \Omega} P(\omega | \Omega_\rho) \left[\hat{\Sigma}_\omega + (\hat{\mu}_\omega - \hat{\mu}^\rho)(\hat{\mu}_\omega - \hat{\mu}^\rho)^T \right], \ \rho \in \{\alpha, \beta\}.
\end{aligned}
\tag{2.21}
$$

Using Bayes theorem, $P(\omega | \Omega_\rho) = \frac{\hat{P}(\omega) P(\Omega_\rho | \omega)}{\hat{P}(\Omega_\rho)}$, where

$$
\hat{P}(\Omega_\rho) = \frac{1}{\hat{P}(\Omega)} \sum_{\omega \in \Omega} P(\Omega_\rho | \omega) \hat{P}(\omega) \quad : \rho \in \{\alpha, \beta\}.
\tag{2.22}
$$

2.5.3 Soft FISHER-Based Feature Extractor

The FISHER(1) feature extractor is computed exactly as described in Section 2.4.1. The only difference is that in the soft meta-classes case the mean and covariance of the two meta-classes are estimated as shown in the previous section. Using these, the within-class covariance $\mathbf{W}_{\alpha, \beta}$ and the between-class covariance $\mathbf{B}_{\alpha, \beta}$ are computed as in Equation (2.6) and Equation (2.7) respectively. The one-dimensional Fisher projection is given by Equation (2.8). The one-dimensional projection obtained byFISHER(1) may not be sufficient for discriminating meta-classes with a large number of classes. Thus, the FISHER(m) feature extractor proposed in Section 2.4.2 is also extended to the soft meta-classes case.

In the BU-BHC algorithm at any merge step, each class belongs to either of the two meta-classes while in the TD-BHC, at any stage of the PARTITION-NODE algorithm, a class $\omega \in \Omega$ partially belongs to *both* the meta-classes. To reflect this soft assignment of classes to the two meta-classes, the pairwise between-class covariance matrix $\tilde{\mathbf{B}}_{\alpha, \beta}$ used in FISHER(m) is modified as follows:

$$
\begin{aligned}
\tilde{\mathbf{B}}_{\alpha, \beta} &= \tfrac{1}{2} \sum_{\omega \in \Omega} \sum_{\omega' \in \Omega - \{\omega\}} |a_{\omega, \alpha} - a_{\omega', \alpha}| P(\omega) P(\omega') (\mu_\omega - \mu_{\omega'})(\mu_\omega - \mu_{\omega'})^T \\
&= \tfrac{1}{2} \sum_{\omega \in \Omega} \sum_{\omega' \in \Omega - \{\omega\}} |a_{\omega, \alpha} - a_{\omega', \alpha}| \mathbf{B}_{\omega, \omega'},
\end{aligned}
\tag{2.23}
$$

where $|a_{\omega, \alpha} - a_{\omega', \alpha}|$ is large if the associations of ω and ω' with the two meta-classes are different. Thus, the weight corresponding to the between-

class covariance component is large only when the associations with the respective classes are different. In the limiting case, when the associations become hard i.e. 0 or 1, then Equation (2.23) reduces to Equation (2.9). The rank of the pairwise between-class covariance matrix is $\min\{D, |\Omega| - 1\}$ and hence the dimensionality of the feature space $\mathcal{F}_n$ at internal node n remains $\min\{D, |\mathbf{\Omega}_n| - 1\}$ as it was in the BU-BHC algorithm. Either FISHER(1) or FISHER(m) can be used as the feature extractors $\psi(X|\mathbf{A})$ in Step 2 of the PARTITIONNODE algorithm.

If the original class densities are Gaussian ($\mathcal{G}(\mathbf{x}|\mu, \Sigma)$), the class density functions in Step 3 of the PARTITIONNODE algorithm in Equation (2.18) for FISHER(1) is:

$$p(\psi_{fisher}^{(1)}(\mathbf{x}|\mathbf{A})|\Omega_\rho) = \mathcal{G}\left(\mathbf{v}_{\alpha\beta}^T\mathbf{x}|\mathbf{v}_{\alpha\beta}^T\mu^\rho, \mathbf{v}_{\alpha\beta}^T\Sigma^\rho\mathbf{v}_{\alpha\beta}\right), \quad \rho \in \{\alpha, \beta\}, \quad (2.24)$$

where $\mathbf{v}_{\alpha\beta}$ is defined in Equation (2.8). Similarly the class density functions for the FISHER(m) feature extractor can be defined as a multivariate ($m_{\alpha\beta}$-dimensional) Gaussians,

$$p(\psi_{fisher}^{(m)}(\mathbf{x}|\mathbf{A})|\Omega_\rho) = \mathcal{G}\left(\mathbf{V}_{\alpha\beta}^T\mathbf{x}, \mathbf{V}_{\alpha,\beta}^T\mu^\rho, \mathbf{V}_{\alpha\beta}^T\Sigma^\rho\mathbf{V}_{\alpha\beta}\right), \quad \rho \in \{\alpha, \beta\}, \quad (2.25)$$

where $\mathbf{V}_{\alpha\beta}$ is defined in Equation (2.10).

2.6 Combining in BHCs

As mentioned in Definition 1, either a hard or a soft classifier can be used at each internal node in the BHC, leading to two types of combiners: hard and soft. In this section both the hard and soft combining schemes are presented. The *hard combiner* Ξ_H essentially uses ideas from decision tree classifiers [3] to propagate a novel example to one of the leaf nodes based on the outputs of all the internal nodes, while the *soft combiner* Ξ_S estimates the true posteriors of the leaf-node classes from the posteriors of the internal node classifiers.

2.6.1 The Hard Combiner

A novel test example is classified by the hard combiner Ξ_H of BHC by pushing it from the root node to a leaf node. The output of the hard classifier at internal node n, $\phi_n^H(\psi_n(\mathbf{x}))$, is a class label $\mathbf{\Omega}_{2n}$ or $\mathbf{\Omega}_{2n+1}$. Depending on the output at node n, $\mathbf{x}$ is pushed either to the left child or the right child. The basic hard combiner is implemented as follows:

1. Initialize $n = 1$ (start at root node).
2. while node n is an internal node, recursively push point $\mathbf{x}$ to the appropriate child:
$$n \leftarrow \begin{cases} 2n & \text{if } \phi_n^H(\psi_n(\mathbf{x})) = \mathbf{\Omega}_{2n} \\ 2n + 1 & \text{if } \phi_n^H(\psi_n(\mathbf{x})) = \mathbf{\Omega}_{2n+1} \end{cases} \quad (2.26)$$
3. Assign the (unique) class label $\mathbf{\Omega}_n$ at the leaf node n to $\mathbf{x}$.

2.6.2 The Soft Combiner

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 $\{P(\omega|\mathbf{x}), \omega \in \mathbf{\Omega}\}$ and then applying the maximum *a posteriori* probability (MAP) rule: $\omega(\mathbf{x}) = arg\max_{\omega\in\mathbf{\Omega}} P(\omega|\mathbf{x})$, to assign the class label $\omega(\mathbf{x})$ to $\mathbf{x}$. The posteriors $P(\omega|\mathbf{x})$ can be computed by multiplying the posterior probabilities of all the internal node classifiers on the path to the corresponding leaf node.

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.*

$$P(\omega|\mathbf{x}) = \prod_{\ell=0}^{\mathcal{D}(\omega)-1} P(\Omega_{n(\omega)}^{(\ell+1)}|\mathbf{x}, \Omega_{n(\omega)}^{(\ell)}), \tag{2.27}$$

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 $\Omega_{n(\omega)}^{(0)} = \Omega_1 = root\ node.$ (See [32] for proof.)

Remark 1 *The posterior probabilities $P_n(\mathbf{\Omega}_k|\mathbf{x}, \mathbf{\Omega}_n)$, $k \in \{2n, 2n+1\}$ are related to the overall posterior probabilities $\{P(\omega|\mathbf{x}), \omega \in \mathbf{\Omega}\}$ as follows:[4]*

$$P_n(\mathbf{\Omega}_k|\mathbf{x}, \mathbf{\Omega}_n) = \frac{\sum_{\omega\in\mathbf{\Omega}_k} P(\omega|\mathbf{x})}{\sum_{\omega\in\mathbf{\Omega}_n} P(\omega|\mathbf{x})}, \quad k \in \{2n, 2n+1\} \tag{2.28}$$

2.7 Experiments

Both BU-BHC and TD-BHC algorithms are evaluated in this section on public-domain data sets available from the UCI repository [35] and National Institute of Standards and Technology (NIST) and two additional hyperspectral data sets. The classification accuracies of eight different combinations of the BHC classifiers (bottom-up vs top-down, FISHER(1) vs FISHER(m) feature extractor and soft vs hard combiners) are compared with multilayered perceptron-based and maximum likelihood classifiers. The class hierarchy that is automatically discovered from both the BU-BHC and TD-BHC for these data sets are shown for some of these data sets to provide concrete examples of the domain knowledge discovered by the BHC algorithms.

2.7.1 Data Sets Used

The BHC was originally formulated by us to tackle the challenging problem of labeling land cover based on remotely-sensed hyperspectral images, but it

---

[4]This relationship can also be used to indirectly prove Theorem 1.

Table 2.1. The twelve classes in the AVIRIS/KSC hyperspectral data set

Num	Class Name
Upland Classes	
1	Scrub
2	Willow Swamp
3	Cabbage palm hammock
4	Cabbage oak hammock
5	Slash pine
6	Broad leaf/oak hammock
7	Hardwood swamp
Wetland Classes	
8	Graminoid marsh
9	Spartina marsh
10	Cattail marsh
11	Salt marsh
12	Mud flats

clearly has broader applicability. Therefore in this section we shall evaluate it on five public-domain data sets in addition to two hyperspectral data sets. The four public-domain data sets obtained from the UCI repository [35] consist of two 26-class English letter recognition data sets (LETTER-I and LETTER-II) with classes A–Z, a 10-class DIGITS data set with classes 0–9 and a six-class SATIMAGE data set with the following classes: red soil, cotton crop, gray soil, damp gray soil, soil with vegetation stubble, and very damp gray soil. See [29] for more details about these data sets.

The two high-dimensional hyperspectral data sets are AVIRIS and HYMAP, both obtained from NASA. AVIRIS covers 12 classes or land-cover types, and we used a 183-band subset of the 224 bands (excluding water absorption bands) acquired by NASA's Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor over Kennedy Space Center in Florida. The seven upland and five wetland cover types identified for classification are listed in Table 2.1. Classes 3–7 are all trees. Class 4 is a mixture of Class 3 and oak hammock. Class 6 is a mixture of broad leaf trees (maples and laurels) and oak hammock. Class 7 is also a broad leaf tree. These classes have similar spectral signatures and are very difficult to discriminate in multispectral, and even hyperspectral, data using traditional methods.

The HYMAP data set represents a nine-class land-cover prediction problem, where the input is 126 bands across the reflective solar wavelength region of 0.441–2.487 μm with contiguous spectral coverage (except in the atmospheric water vapor bands) and bandwidths between 15 and 20 nm. This data set was obtained over Stover Point (South Texas) in September of 1999. The vegetation here consists of common high estuarine marsh species including *Spartina spartinae*, *Borrichia frutescens*, *Monanthochloa littoralis*, and *Batis*

Table 2.2. The nine classes in the STOVER/HYMAP data set.

Num	Class Name
1	Water
2	*Spartina Spartinae*
3	*Batis maritima*
4	*Borrichia frutescens + Spartina spartinae + Monanthocloa littoralis*
5	Sand flats (bare soil)
6	Pure *Borrichia frutescens*
7	Trees
8	Dense bushes
9	*Borrichia frutescens + Spartina spartinae*

maritima. Adjacent to the resaca (which is a generic term that refers to an old river bed which has been cut off by the meandering of the river resulting in an ox-bow) is an almost impregnable layer of dense shrubs and trees. The nine classes determined for Stover Point are listed in Table 2.2.

2.7.2 Classification Results

The eight versions of the BHC framework that are evaluated on the data sets described in the previous section are generated by the following sets of choices:

- **Building the tree**: The BHC tree can be built either bottom-up or top-down. The biases of the BU-BHC and the TD-BHC algorithms are different. The BU-BHC tries to find the most similar meta-classes from the available set and hence is more greedy at each step than the TD-BHC, which attempts to partition a meta-class into two subsets with a more global perspective. As a result of the differences of these biases, different BHC trees and therefore different classification accuracies can be obtained.
- **Feature extractor used**: Both the FISHER(1) and FISHER(m) feature extractors based on Fisher's discriminant are investigated. While the tree structure for the two Fisher projections may be different, the discrimination between classes at any internal node using FISHER(m) projections is higher than the discrimination with FISHER(1) projection and therefore FISHER(m)-based BHC performs better in general than the corresponding BHC with FISHER(1) feature extractor.
- **Nature of combiner**: Both hard and soft combiners were investigated. In general, the soft combiner performs slightly better than the hard combiner, as is expected.

The classification accuracy averaged over 10 experiments on each data set is reported in Table 2.3. In each experiment, stratified sampling was used to partition the data set into training and test sets of equal size.[5] Eight versions

---

[5] 10-fold cross validation is currently in vogue in some circles but is an overkill for fairly large data sets.

of the BHC classifiers were compared to two standard classifiers leading to
the following 10 classifiers for each data set:

- MLP: a finely-tuned multilayered perceptron-based classifier for each data
  set;
- MLC: a maximum-likelihood classifier using a full covariance matrix wher-
  ever possible and using a diagonal covariance matrix if the full covariance
  matrix is ill-conditioned due to high input dimensionality;
- BU-BHC(1,H): BU-BHC with FISHER(1) and hard combiner;
- BU-BHC(1,S): BU-BHC with FISHER(1) and soft combiner;
- BU-BHC(m,H): BU-BHC with FISHER(m) and hard combiner;
- BU-BHC(m,S): BU-BHC with FISHER(m) and soft combiner;
- TD-BHC(1,H): TD-BHC with FISHER(1) and hard combiner;
- TD-BHC(1,S): TD-BHC with FISHER(1) and soft combiner;
- TD-BHC(m,H): TD-BHC with FISHER(m) and hard combiner;
- TD-BHC(m,S): TD-BHC with FISHER(m) and soft combiner.

Table 2.3. Classification accuracies on public-domain data sets from the UCI repos-
itory [35] (SATIMAGE, DIGITS, LETTER-I) and NIST(LETTER-II) and remote-sensing
data sets from the Center for Space Research, The University of Texas at Austin
(HYMAP, AVIRIS). The input dimensions and number of classes are also indicated for
each data set.

	SATIMAGE	DIGITS	LETTER-I	LETTER-II	HYMAP	AVIRIS
Dimensions	36	64	16	30	126	183
Classes	6	10	26	26	9	12
MLP	79.77	82.33	79.28	76.24	78.21	74.54
MLC	77.14	74.85	82.73	79.48	82.73	72.66
BU-BHC(1,H)	83.26	88.87	71.29	78.45	95.18	94.97
BU-BHC(1,S)	84.48	89.00	72.81	79.93	95.62	95.31
BU-BHC(m,H)	85.29	91.71	76.55	80.94	95.12	95.51
BU-BHC(m,S)	85.35	91.95	78.41	81.11	95.43	95.83
TD-BHC(1,H)	83.77	90.11	70.45	74.59	95.31	96.33
TD-BHC(1,S)	84.02	90.24	72.71	75.83	95.95	97.09
TD-BHC(m,H)	84.70	91.44	77.85	81.48	96.48	97.15
TD-BHC(m,S)	84.95	91.61	79.13	81.99	96.64	97.93

The finely tuned MLP classifiers and the MLC classifiers are used as bench-
marks for evaluating the BHC algorithms. Almost all the BHC versions per-
formed significantly better than the MLP and MLC classifiers on all data sets
except LETTER-I and LETTER-II. In general the TD-BHC was slightly bet-
ter than the BU-BHC mainly because its global bias leads to less greedy trees
than the BU-BHC algorithm. Further, the FISHER(m) feature extractor con-
sistently yields slightly better results than the FISHER(1) feature extractor,

as expected. Finally, the soft combiner also performed sightly better than the hard combiner. This again is an expected result as the hard combiner loses some information as it thresholds the posteriors at each internal node.

2.7.3 Discussion of Results and Further Comparative Studies

From Table 2.3, we see that the BHC classifiers did not perform as well on the LETTER-I data set. This turns out to be due to the presence of some bimodal classes in this data set, which is problematic for the simple Fisher discriminant. For such data sets it is preferable to use more powerful binary classifiers at the internal nodes of the BHC, i.e. use the BHC framework only to obtain the class hierarchy and then use other, more appropriate, feature-extractors/classifiers for the two-class problems at each internal node. This intuition is borne out in our recent work [40] where gaussian-kernel based SVMs were used for the internal nodes, leading to statistically significant performance improvements for all the nine data sets considered. Also of interest is the comparison of this BHC-SVM architecture with an ECOC-based ensemble (using well-tuned SVMs as the base classifiers) given in this work. The outcome of this comparison is not obvious since two very different philosophies are being encountered. While the BHC groups the classes according to their natural affinities in order to make each binary problem easier, it cannot exploit the powerful error correcting properties of an ECOC ensemble that can provide good results even when individual classifiers are weak. This empirical study showed that while is no clear advantage to either technique in terms of classification accuracy, the BHCs typically achieve this performance using fewer classifiers. Note that each dichotomy in an ECOC setup can be addressed using all the training data, while for the BHC the data available decreases as one moves away from the root since only a subset of the classes are involved in lower-level dichotomies. Thus one may expect the ECOC approach to be less affected by a paucity of training data. However the experiments in [40] showed that BHC was competitive even for small sample sizes, indicating that the reduction in data is compensated for by the simpler dichotomies resulting from affinity-based grouping of classes.

All the results above assume equal penalty for each type of misclassification. In many real applications, classification into a nearby class is less costly than being labeled as a distant class. For example, wet gray soil being classified as damp gray soil is not as costly as being labeled as red, dry soil. If such asymmetric costs are considered, the coarse-to-fine approach of the BHC framework provides an additional advantage over all the other methods considered.

2.8 Domain Knowledge Discovery

One of the important aspects of the BHC classifiers is the domain knowledge that is discovered by the automatic BU-BHC and TD-BHC tree construction

algorithms. The trees constructed by the BU-BHC(m) and TD-BHC(m) are shown in Figures 2.3 to 2.10 for the most common of the trees obtained in the ten experiments for each data set. The numbers at the internal nodes of the binary trees represent the mean training and test set classification accuracies at that internal node over all the experiments for which this tree is obtained.

- **IRIS**: It is well known that Iris Versicolour and Virginica are "closer" to each other than Iris Setosa. So, not surprisingly, the first split for both BU-BHC(m) and TD-BHC(m) algorithms invariably separates Setosa from the other two classes.
- **SATIMAGE**: Figures 2.3 and 2.4 show the BU-BHC(m) and TD-BHC(m) trees generated for the SATIMAGE data set. In the BU-BHC tree, the Classes 4 (damp gray soil) and 6 (very damp gray soil) merged first. This was followed by Class 3 (gray soil) merging in the meta-class (4,6). The right child of the root node contains the remaining three classes out of which the vegetation classes i.e. Class 2 (cotton crop) and Class 5 (soil with vegetation stubble) were grouped first. The tree formed in the TD-BHC is even more informative as it separates the four bare soil classes from the two vegetation classes at the root node and then separates the four soil classes into *red-soil* (Class 1) and *gray-soil* (Classes 3, 4, and 6) meta-classes. The *gray-soil* meta-class is further partitioned into *damp-gray-soil* (Classes 4 and 6) and *regular-gray-soil* (Class 3). Thus reasonable class hierarchies are discovered by the BHC framework for the SATIMAGE data set.
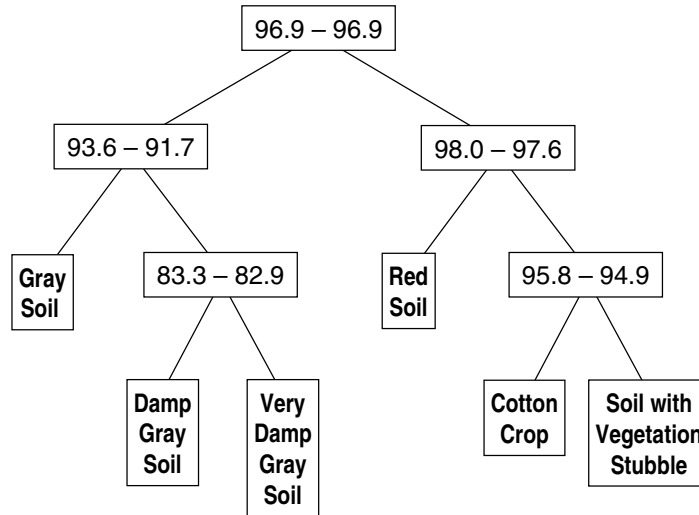
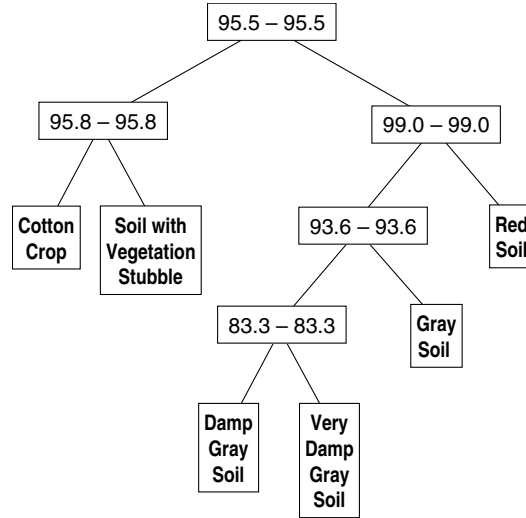**Fig. 2.3.** BU-BHC(m) class hierarchy for the SATIMAGE data set.

Fig. 2.4. TD-BHC(m) class hierarchy for the SATIMAGE data set.

- **LETTER**: The 26-class LETTER-I data set is only 16-dimensional. Although relatively lower dimensionality makes it an "easier" problem from the curse of dimensionality perspective, the fact that the number of classes is more than the dimensionality makes it a "harder" problem from the problem decomposition perspective. As seen in Table 2.3, the performance of BHC classifiers actually is poorer than other approaches, the reasons for which have already been discussed. Nevertheless, it is interesting to see the trees obtained by the BHC algorithms for such a large (in terms of output space) classification problem (Figures 2.5 and 2.6). Several interesting groups of characters are merged in the BU-BHC tree. For example meta-classes like {M,W,N,U} {F,P}, {V,Y,T}, {S,Z,B,E}, {I,J}, {K,R}, and {G,Q,C} are discovered. These conform well with the shapes of the letters. The TD-BHC tree is different from the BU-BHC tree but also has several interesting meta-classes like {M,W,U,H,N}, {K,R}, {V,Y,T}, {F,P}, {S,Z}, {C,G,O}, and {B,D,E}. Even for a small dimensional input space, as compared to the number of classes, the BHC algorithm was able to discover a meaningful class hierarchy for this 26-class problem. However, note that one could have obtained other reasonable hierarchies as well, and it is difficult to quantify the quality of a specific hierarchy other than through its corresponding classification accuracy.
- **LETTER-II**: Since the output space is still the same, the BHC trees for LETTER-II should be similar to the LETTER-I trees. In our experiments, similar interesting meta-classes such as {M,W,N,U}, {F,P}, {V,T}, {S,Z,B,E}, {I,J} and {G,Q,C} were discovered in the BU-BHC tree. The
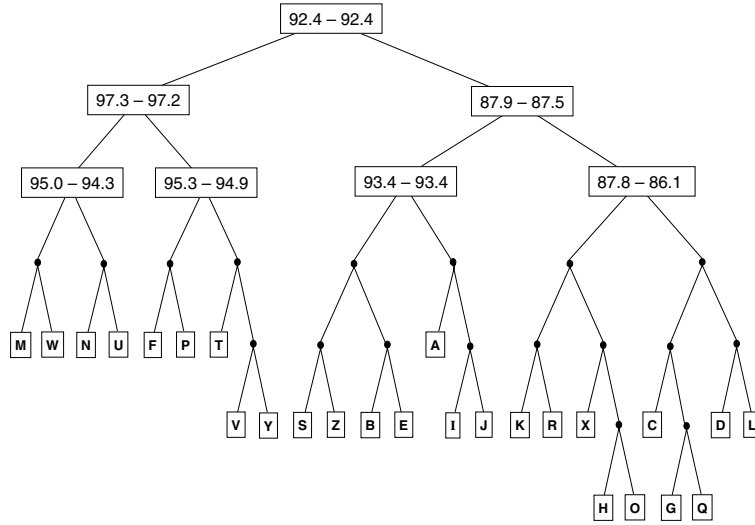
Fig. 2.5. BU-BHC(m) class hierarchy for the LETTER-I data set.
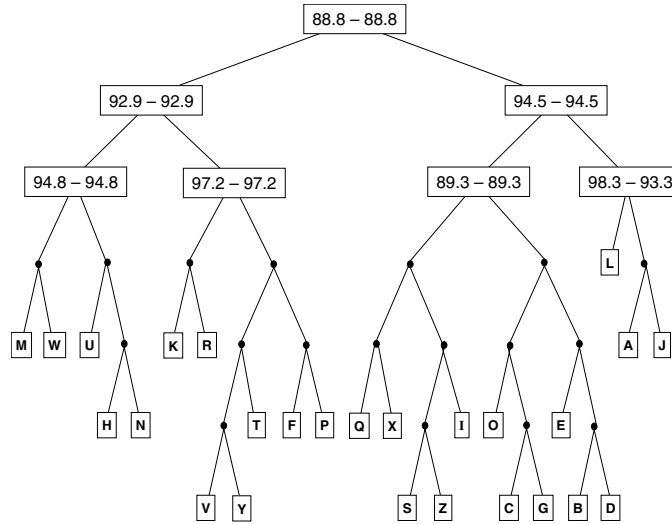
Fig. 2.6. TD-BHC(m) class hierarchy for the LETTER-I data set.

TD-BHC classifier for LETTER-II data set resulted in a few new groupings as well, including {O,Q}, {H,K,A,R} and {P,D}.

- **Hyperspectral data**: Figures 2.7, 2.8, 2.9 and 2.10 show the bottom-up and top-down trees obtained for AVIRIS and HYMAP. By considering the meaning of the class labels it is evident that this domain provided the most useful knowledge. Invariably, when water was present, it was the first to

be split off. Subsequent partitions would, for example, distinguish between marshy wetlands and uplands, as in Figure 2.1. Note that the trees shown are representative results. While there are sometimes small variations in the trees obtained by perturbing the data, invariably all the trees produce hierarchies that are meaningful and reasonable to a domain expert [24].
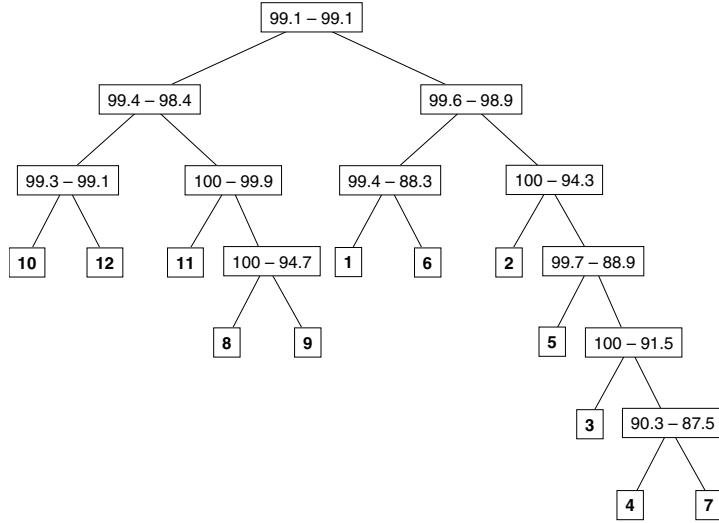
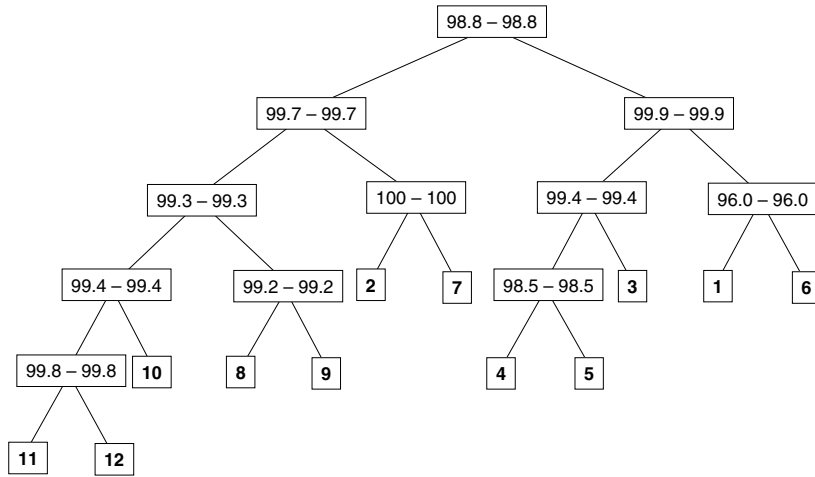**Fig. 2.7.** BU-BHC(m) class hierarchy for the AVIRIS data set.
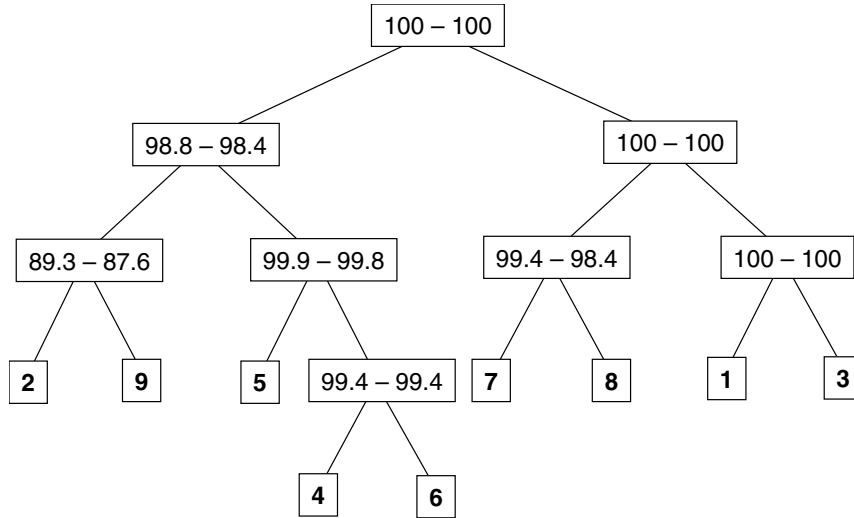
Fig. 2.8. TD-BHC(m) class hierarchy for the AVIRIS data set.

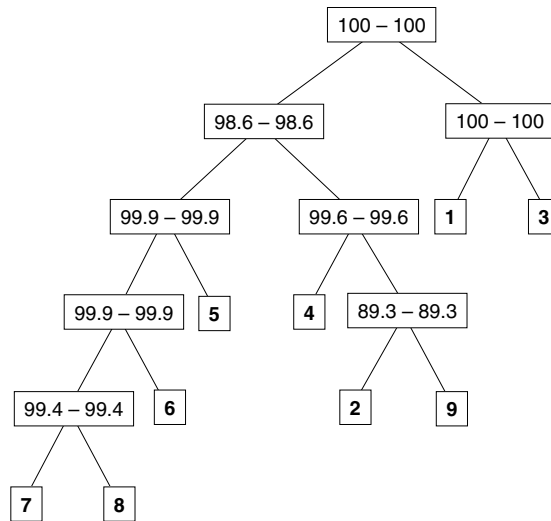**Fig. 2.9.** BU-BHC(m) class hierarchy for the HYMAP data set.

Fig. 2.10. TD-BHC(m) class hierarchy for the HYMAP data set.

2.9 Conclusions

This chapter presented a general framework for certain difficult classification problems in which the complexity is primarily due to having several classes as well as high-dimensional inputs. The BHC methodology relies on progressively partitioning or grouping the set of classes based on their affinities with one another. The BHC, as originally conceived, uses a custom Fisher's discriminant feature extraction for each partition, which is quite fast as it only involves summary class statistics. Moreover, as a result of the tree building algorithms, a class taxonomy is automatically discovered from data, which often leads to useful domain knowledge. This property was particularly helpful in our analysis of hyperspectral data.

The hierarchical BHC approach is helpful only if some class affinities are actually present, i.e. it will not be appropriate if all the classes are essentially "equidistant" from one another. In practice, this is not very restrictive since many applications involving multiple class labels, such as those based on biological or text data, do have natural class affinities, quite often reflected in class hierarchies or taxonomies. In fact it has been shown that exploiting a *known* hierarchy of text categories substantially improves text classification [5]. In contrast, the BHC attempts to induce a hierarchy directly from the data where no pre-existing hierarchy is available. Another recent approach with a similar purpose is presented in [19] where Naive Bayes is first used to quickly generate a confusion matrix for a text corpus. The classes are then clustered based on this matrix such that classes that are more confused with one another tend to be placed in the same group. Then SVMs are used in a "one-versus-all" framework within each group of classes to come up with the final result. Thus this approach produces a two-level hierarchy of classes. On text benchmarks, this method was three to six times faster than using "one-vs-all" SVMs directly, while producing comparable or better classification results.

We note that one need not be restricted to our choices of a Fisher discriminant and a simple Bayesian classifier at each internal node of the class-partitioning tree. In Section 2.7.3, we summarized our related work on using SVMs as the internal classifiers on a tree obtained via the Fisher discriminant/Bayesian classifier combination. The feature extraction step itself can also be customized for different domains such as image or protein sequence classification. In this context, recollect that the trees obtained for a given problem can vary somewhat depending on the specific training set or classifier design, indicative of the fact that that there are often multiple reasonable ways of grouping the classes. The use of more powerful binary classifiers provides an added advantage in that the overall results are more tolerant to the quality of the tree that is obtained.

The design space for selecting an appropriate feature extractor–classifier combination is truly rich and needs to be explored further. A well-known trade-off exists between these two functions. For example, a complex feature

extraction technique can compensate for a simple classifier. With this viewpoint, let us compare the top-down BHC with decision trees such as C5.0, CART and CHAID. One can view the action at each internal node of a decision tree as the selection of a specific value of exactly one variable (feature extraction stage), followed by a simple classifier that just performs a simple comparison against this value. Thus the BHC node seems more complex. However, the demands on a single node in a decision tree are not that strong, since samples from the same class can be routed to different branches of the tree and still be identified correctly at later stages. In contrast, in the hard version of BHC, all the examples of a given class have to be routed to the same child at each internal node visited by them.

References

[1] Ballard, D., 1987: Modular learning in neural networks. *Proc. AAAI-87*, 279–84.

[2] Bellman, R. E., ed., 1961: *Adaptive Control Processes*. Princeton University Press.

[3] Breiman, L., J. H. Friedman, R. Olshen and C. J. Stone, 1984: *Classification and Regression Trees*. Wadsworth and Brooks, Pacific Grove, California.

[4] Brill, F. Z., D. E. Brown and W. N. Martin, 1992: Fast genetic selection of features for neural network classifiers. *IEEE Transactions on Neural Networks*, **3**, 324–28.

[5] Chakrabarti, S., B. Dom, R. Agrawal and P. Raghavan, 1998: Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal*, **7**, 163–78.

[6] Chakravarthy, S., J. Ghosh, L. Deuser and S. Beck, 1991: Efficient training procedures for adaptive kernel classifiers. *Neural Networks for Signal Processing*, IEEE Press, 21–9.

[7] Crawford, M. M., S. Kumar, M. R. Ricard, J. C. Gibeaut and A. Neuenshwander, 1999: Fusion of airborne polarimetric and interferometric SAR for classification of coastal environments. *IEEE Transactions on Geoscience and Remote Sensing*, **37**, 1306–15.

[8] Dattatreya, G. R. and L. N. Kanal, 1985: Decision trees in pattern recognition. *Progress in Pattern Recognition 2*, L. N. Kanal and A. Rosenfeld, eds., Elsevier Science, 189–239.

[9] Deco, G. and L. Parra, 1997: Nonlinear feature extraction by redundancy reduction in an unsupervised stochastic neural network. *Neural Networks*, **10**, 683–91.

[10] Dietterich, T. G. and G. Bakiri, 1995: Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, **2**, 263–86.

[11] Duda, R. and P. Hart, 1973: *Pattern Classification and Scene Analysis*. Addison-Wesley.

[12] Etemad, K. and R. Chellappa, 1998: Separability-based multiscale basis selection and feature extraction for signal and image classification. *IEEE Transactions on Image Processing*, **7**, 1453–65.

[13] Friedman, J., 1989: Regularized discriminant analysis. *Journal of the American Statistical Association*, **84**, 165–75.

[14] — 1996: Another approach to polychotomous classification. Technical report, Stanford University.

[15] — 1996: On bias, variance, 0/1 loss, and the curse-of-dimensionality. Technical report, Department of Statistics, Stanford University.

[16] Fukunaga, K., 1990: *Introduction to Statistical Pattern Recognition* (2nd Ed.), Academic Press, NY.

[17] Furnkranz, J., 2002: Round robin classification. *Jl. Machine Learning Research*, **2**, 721–47.

[18] Ghosh, J., 2003: Scalable clustering. *The Handbook of Data Mining*, N. Ye, ed., Lawrence Erlbaum Assoc., 247–77.

[19] Godbole, S., S. Sarawagi and S. Chakrabarti, 2002: Scaling multi-class support vector machines using inter-class confusion. *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (KDD-02)*, 513–18.

[20] Hand, D., 1982: *Kernel Discriminant Analysis*. Research Studies Press, Chichester, UK.

[21] Happel, B. and J. Murre, 1994: Design and evolution of modular neural network architectures. *Neural Networks*, **7:6/7**, 985–1004.

[22] Hastie, T. and R. Tibshirani, 1996: Discriminant adaptive nearest neightbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-18**, 607–16.

[23] — 1998: Classification by pairwise coupling. *Advances in Neural Information Processing Systems*, M. J. K. Michael, I. Jordan and S. A. Solla, eds., MIT Press, Cambridge, Massachusetts, **10**, 507–13.

[24] Henneguelle, A., J. Ghosh and M. M. Crawford, 2003: Polyline feature extraction for land cover classification using hyperspectral data. *Proc. IICAI-03*, 256–69.

[25] Hsu, C. W. and C. J. Lin, 2002: A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, **13**, 415–25.

[26] Jordan, M. and R. Jacobs, 1994: Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, **6**, 181–214.

[27] Khotanzad, A. and Y. Hong, 1990: Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 28–37.

[28] Kittler, J. and F. Roli, eds., 2001: *Multiple Classifier Systems*. LNCS Vol. 1857, Springer.

[29] Kumar, S., 2000: *Modular learning through output space decomposition*. Ph.D. thesis, Dept. of ECE, Univ. of Texas at Austin, USA.

[30] Kumar, S. and J. Ghosh, 1999: GAMLS: A generalized framework for associative modular learning systems (invited paper). *Proceedings of the Applications and Science of Computational Intelligence II*, Orlando, Florida, 24–34.

[31] Kumar, S., J. Ghosh and M. M. Crawford, 1999: A versatile framework for labeling imagery with a large number of classes. *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C.

[32] — 2002: Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications, splecial issue on Fusion of Multiple Classifiers*, **5**, 210–20.

[33] Mao, J. and A. K. Jain, 1995: Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, **6 (2)**, 296–317.

[34] McLachlan, G. J., 1992: *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley, New York.

[35] Merz, C. and P. Murphy, 1996: UCI repository of machine learning databases. URL: `www.ics.uci.edu/∼mlearn/MLRepository.html`.

[36] Murray-Smith, R. and T. A. Johansen, 1997: *Multiple Model Approaches to Modelling and Control*. Taylor and Francis, UK.

[37] Nilsson, N. J., 1965: *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*. McGraw Hill, NY.

[38] Petridis, V. and A. Kehagias, 1998: *Predictive Modular Neural Networks: Applications to Time Series*. Kluwer Academic Publishers, Boston.

[39] Platt, J. C., N. Cristianini and J. Shawe-Taylor, 2000: Large margin DAGs for multiclass classification. MIT Press, **12**, 547–53.

[40] Rajan, S. and J. Ghosh, 2004: An empirical comparison of hierarchical vs. two-level approaches to multiclass problems. *Multiple Classifier Systems*, F. Roli, J. Kittler and T. Windeatt, eds., LNCS Vol. 3077, Springer, 283–92.

[41] Ramamurti, V. and J. Ghosh, 1998: On the use of localized gating in mixtures of experts networks (invited paper), *SPIE Conf. on Applications and Science of Computational Intelligence, SPIE Proc. Vol. 3390*, 24–35.

[42] — 1999: Structurally adaptive modular networks for nonstationary environments. *IEEE Trans. on Neural Networks*, **10**, 152–60.

[43] Rasoul Safavian, S. and D. Landgrebe, 1991: A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, **21**, 660–74.

[44] Rifkin, R. and A. Klautau, 2004: In defense of one-vs-all classification. *Jl. Machine Learning Research*, **5**, 101–41.

[45] Sakurai-Amano, T., J. Iisaka and M. Takagi, 1997: Comparison of land cover indices of AVHRR data. *International Geoscience and Remote Sensing Symposium*, 916–18.

[46] Schölkopf, B., C. Burges and A. J. Smola, eds., 1998: *Advances in Kernel Methods: Support Vector Learning*. MIT Press.

[47] Sharkey, A., 1999: *Combining Artificial Neural Nets*. Springer-Verlag.

[48] Sharkey, A. J. C., N. E. Sharkey, and G. O. Chandroth, 1995: Neural nets and diversity. *Proceedings of the 14th International Conference on Computer Safety, Reliability and Security*, Belgirate, Italy.

[49] Tumer, K. and N. C. Oza, 1999: Decimated input ensembles for improved generalization. *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C.

[50] Vapnik, V., 1995: *The Nature of Statistical Learning Theory*. Springer.