LUDWIG-MAXIMILIANS-UNIVERSITÄT
TECHNISCHE UNIVERSITÄT MÜNCHEN

Institut für Informatik XII

Diplomarbeit
in Bioinformatik

Hierarchical Classification Using Ensembles of Nested Dichotomies (ENDs)
Hierarchische Klassifikation mit Ensembles von verschachtelten Dichotomien

Arthur Zimek

Aufgabensteller:	Prof. Dr. Stefan Kramer
Betreuer:	Prof. Dr. Stefan Kramer
Abgabedatum:	15.01.2005

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

15.01.2005 _____

Arthur Zimek

Abstract

Two tasks among others are often considered in machine learning research: On the one hand to reduce polytomous classification problems to dichotomous ones in order to be able to apply binary classifiers to the multi-class problem. On the other hand to build ensembles of classifiers that improve upon their respective ensemble members. The method of 'ensembles of nested dichotomies' (ENDs), recently proposed by Frank and Kramer [66], copes well with both tasks.

In this thesis we propose to build ensembles of nested dichotomies with respect to a hierarchy of classes. This idea moreover tackles another general task: incorporating domain knowledge in learners. We define a subspace of the space of possible nested dichotomies with respect to a predefined hierarchy. Ensembles of hierarchically nested dichotomies (EHNDs) are built by randomly selecting nested dichotomies from the space restricted in this way.

Given the hierarchy is appropriate to the classification task and reflected in the feature space, ensembles of hierarchically nested dichotomies are shown to improve upon ensembles of nested dichotomies. Both methods perform equally well in application to protein fold classification ('fold recognition'). This finding motivates to question the sufficiency of feature representations of proteins known so far. Nevertheless, in fold recognition datasets EHNDs are found to perform equally well as the best performing machine learning approaches we are aware of and reach also the level of accuracy of alignment based methods on difficult data.

Zusammenfassung

Die Disziplin des Maschinellen Lernens befasst sich unter anderem mit den beiden Fragestellungen: Reduktion polytomer Klassifikationsprobleme auf dichotome, um binäre Klassifizierer auf diese Probleme anwenden zu können, und Bildung von Ensembles von Klassifizierern, die als Ensemble bessere Genauigkeit erreichen als die einzelnen Klassifizierer. Kürzlich wurde die Methode 'Ensembles verschachtelter Dichotomien' (ENDs) von Frank und Kramer [66] eingeführt, die erfolgreich an beide Aufgaben zu gleich herangeht.

In dieser Arbeit schlagen wir vor, Ensembles von verschachtelten Dichotomien unter Berücksichtigung einer Hierarchie von Klassen zu bilden. Diese Idee bezieht eine weitere Fragestellung des Maschinellen Lernens in die Methode mit ein: die Berücksichtigung von Hintergrundwissen das Klassifikationsproblem betreffend. Wir definieren einen Unterraum des Raumes der möglichen verschachtelten Dichotomien, indem wir diesen Raum mit Rücksicht auf die vorgegebene Hierarchie beschränken. Ensembles von hierarchisch verschachtelten Dichotomien (EHNDs) werden durch eine zufällige Auswahl aus diesem beschränkten Raum gebildet.

Unter der Voraussetzung, dass die Hierarchie der Klassifikationsaufgabe angemessen ist und sich auch im Raum der Attribute wiederspiegelt, konnten Ensembles hierarchisch verschachtelter Dichotomien eine höhere Klassifikationsgenauigkeit erzielen als allgemeine Ensembles verschachtelter Dichotomien. In der Anwendung auf die Faltungs-Klassifikation von Proteinen ('fold recognition') erreichen beide Methoden, EHNDs und ENDs, das gleiche Niveau an Klassifikations-Genauigkeit. Dieses Ergebnis regt zu der Frage an, ob sich nicht bessere als die bisher bekannten Attribut-Räume für Proteine finden lassen könnten. Desungeachtet erreichen EHNDs für fold recognition-Datensätze die gleiche Klassifikationsgenauigkeit wie die besten der uns bekannten, auf dieses Problem angewandten, Ansätze des Maschinellen Lernens. Auf einem schwierigen Datensatz wird auch die Genauigkeit der gegenwärtig unseres Wissens besten alignment-basierten Methode erreicht.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Hierarchies are not only a common mean to organize and to present a large accumulation of different things like items in a department store or directories on a computer's disk, or to preserve order and effectiveness in communities like governments, companies, or churches. Hierarchies are also naturally occurring whenever things share a common root, pursuit an at least partially common purpose, or are set out to common forces or influences, like minerals, geological formations, or organisms. Considering hierarchies for living creatures to describe their common properties as well as the differences between them roots back to Aristotle, who distinguished animals and plants while treating both as living things and divided animals with respect to certain criteria [12]. His arrangement of the invertebrates seems to be superior to the classification generated 2000 years later by Linnaeus [106]. However, the scheme developed by Linnaeus in the 18^{th} century, organizing living organisms in a tree, is in use until nowadays even though some subtrees were and are rearranged over and over again. The hierarchy developed by Linnaeus was starting with three kingdoms as children of the root. Kingdoms were divided into classes and they, in turn, into orders, families, genera, and species. A few other ranks have been added, most notably phyla or divisions between kingdoms and classes, since then. Groups of organisms at any of these ranks are now called taxa, or phyla, or taxonomic groups.

While this early and classical system of organisms is based on the phenotype of whole organisms, that is their outer appearance (*morphology*), the reasons for such relationships were first revealed by Charles Darwin [42] to be based on inheritance, diversification, and selection. Ironically, Darwin's reasoning was strongly based on the hierarchical classification developed by Linnaeus a hundred years earlier, while Linnaeus himself understood his own work as revelation of the unchangeable order within the plan of divine action ('*ad majorem Dei gloriam*') and is cited with the words: '*Deus creavit, Linnaeus disposuit*' [74]. And as a matter of fact, Darwin found even in the works of Aristotle notes that seemed related to his own reasoning.

Nowadays, biology can not be imagined neglecting the theory of evolution ('*Nothing in Biology Makes Sense Except in the Light of Evolution*', as the famous statement of Dobzhansky says [54]), and modern molecular biology is concerned with evolutionary relationships, that is hierarchies, of objects on a molecular level, concerning e.g. genes or proteins. This approach revealed several misclassifications by the classical method, where analogous developments (i.e., similar developments in response to similar environments, but not based on relationship) may have been misleading. But on the other hand also a hierarchy of related molecules is unfortunately not always indisputable. And even if it were undisputed, deriving relationships of organisms based on molecular hierarchies would not be straightforward, since the respective hierarchies might differ for different genes within one organism

due to different rates of mutation. Thus biological data often could more properly be organized in general graphs than in trees of relationships [55].

Nevertheless, hierarchical organization of data in biology is ubiquitous. This fact is inspiring to the attempt of incorporation of knowledge concerning hierarchical structures of data into the task of building automated classifiers for these data which is ventured on by machine learning as a special subject of computer science. Building suitable and prosper learners is a very manifold task. The **first part** of this thesis will introduce the problems we are generally dealing with.

At first a machine learning algorithm is designed to create a model that is able to discern members of a class a from members of a class b. But often there are more than just two classes. Also in the realm of biology the world is manifold. So tackling problems that involve many classes (a so called *polytomous* problem) is an interesting and well (but obviously not exhaustively) studied problem in machine learning. We will review the problem and some established solutions to it in the **second chapter**.

The **third chapter** is concerned with the task of building ensembles of learning machines. Generally, this is a field of its own in machine learning research. But it is also related to the task described in the second chapter since some ensemble methods are implicitly or explicitly designed to handle polytomous classification problems. There is especially one method, called '*ensembles of nested dichotomies*' tackling both tasks at once, the reduction of polytomies to dichotomies and building an ensemble. This method was recently introduced by Frank and Kramer [66]. We will review this method in **chapter 4** since it is also the base for the method developed in this thesis. Our method provides the possibility to incorporate knowledge concerning hierarchical structures of classes into the method of '*ensembles of nested dichotomies*'. Of course, as stated above, hierarchically structured classification problems do not only occur in biology but in many fields of possible applications and were considered for development of other methods before. The respective field of research is generally concerned with incorporation of domain knowledge into classification tasks. There are many and diverse ways to do so but so far no general theory is developed. **Chapter 5** presents a very short view on this topic and will only shortly point out previous attempts to take hierarchical structure into account for tasks of learning. We are aware of mainly two fields, namely reinforcement learning and text mining. However, our motivation is mainly rooted in biology.

Based on this background the **second part** of our thesis will introduce and evaluate the method of '*ensembles of hierarchically nested dichotomies*'. This method builds '*ensembles of nested dichotomies*' where each system of nested dichotomies is bound to reflect a given hierarchical structure of the classes. In order to prepare a suitable definition of this reflection and to derive the method's properties **chapter 6** will firstly introduce a theoretical framework to handle hierarchies of classes, then a method to derive systems of nested dichotomies that actually reflect a given hierarchy will be introduced. Thirdly the properties of the resulting ensemble method are examined. Some remarks concerning related work will round off the introduction of the method of '*ensembles of hierarchically nested dichotomies*'.

Chapter 7 presents an evaluation of our method on synthetical data. This does not only provide a first impression of its performance. A problem one gets involved with using real-world data is the uncertainty of the hierarchy – if it is correctly assigned at all – to be sufficiently reflected in the feature space. Thus having shown the method to work better than related methods on data exhibiting a pronounced hierarchy in the feature space used for training and testing (and the synthetical data is designed to do so, of course) will also allow to evaluate other feature spaces on data where a hierarchy is claimed to exist, but the accurate exhibition of this hierarchy within the feature space is not proven.

In **part III**, **chapter 8** we therefore examine the performance of our method

on protein fold classification, a task that was several times tackled with by machine learning methods using established feature spaces and also by alignment based methods, that do not hinge on feature representations of sequences. For protein fold, several hierarchies are introduced in the literature. However, the reflection of these hierarchies in even well established feature spaces is neither proven nor questioned so far. Nevertheless, it seems quite suggesting to take the hierarchy of protein folds into account for classification tasks, as stated by Reddy and Bourne [127, p. 243]:

> *The best way to characterize fold is to look first at the major architectural features and then identify the more subtle characteristics.*

This is a prescription to help the human eye, but the same recipe could also be helpful to enhance performance of machine learning approaches.

A final **chapter 9** in **part IV** is to summarize our findings and to attempt to give an outlook on possible biological applications and some suggesting generalizations of our method.

An **appended CD** finally provides the data used for evaluation and the implementations we prepared for this thesis.

Part I

Background and Motivation

Chapter 2

Approaches to Polytomous Classification

In machine learning one large set of methods is referred to as *supervised learning*. Algorithms of that kind are trained on instances where the correct output is known: An instance i is represented by d features often thought of as a feature vector $\vec{x}_i \in \mathbb{R}^d$. Machine learning is based on the assumption that a function $f : \mathbb{R}^d \to C$ exists that assigns a class $c_j \in C$ to any instance $\vec{x} \in \mathbb{R}^d$. A supervised learner is supposed to estimate an approximation h (also referred to as *model* or *classifier*) to this function f, based on a set of k training instances $(\vec{x}_i, c_j), i \in \{1, \ldots, k\}, j \in \{1, \ldots, n\}$, where the correct output c_j of the target function f is known for a feature vector $\vec{x}_i$.

If the target function f provides a *classification*, the space of possible outputs C will usually be restricted to a small range of discrete or categorical values, referred to as *classes*. Otherwise the predicted outputs will be quantitative and the prediction task is called *regression*. However, quantitative outputs of regression could be mapped to class-labels as well. Clearly, the task of classification is simpler for two possible classes than it is for more than two classes, as will be explained below.

The aim of the current chapter is to outline briefly the increasing difficulties of the classification task with an increasing number of classes and basic approaches for tackling these difficulties. For this outline and for a general view we refer to standard textbooks [80], [82], [110], and [151].

2.1 Polytomous versus Dichotomous Classification

Approaching classification problems by means of machine learning generally becomes more difficult with a raising number n of classes for several reasons:

- Firstly, the number of training instances is usually limited and thus becomes on average smaller for a single class.

- Secondly, if each class is to be distinguished from each other class, there will be (explicitly or implicitly) $\frac{n \times (n-1)}{2}$ decision boundaries to be defined, that is, the number of required decision boundaries raises with the squared number of classes.

  Decision trees – as an example of classifiers directly applicable to multi-class-problems – involve the expected reduction in entropy H of the set of training examples, corresponding to the homogeneity of examples as a possible decision criterion which attribute to select as the most promising for a decision on

the respective next level. Clearly, to gain homogeneity of examples will be the more difficult the more classes the examples are distributed on. This corresponds to the entropy H raising with the number n of classes. The entropy H is maximal for a given n when all class-probabilities are equal, i.e., $\frac{1}{n}$, and equals $\log n$ [138]. This slow (only logarithmic) increase is due to the fact that a decision tree is not bound to discern each class from each other class. Instead it can group classes together and discern the group from other groups of classes. We will keep this efficient method in mind and encounter it again. For now, our point is simply the increase of complexity, which is not avoidable, regardless how slow it is.

- Thirdly, a feature space of higher dimensionality could eventually allow separation of more classes more easily than a feature space of lower dimensionality does. Thus one would prefer high dimensional feature spaces for multi-class problems. On the other hand high dimensional feature spaces are a problem of their own kind, since they usually increase the complexity of the learning task. In addition they increase the probability of irrelevant attributes or noisy data to guide the classifier in wrong directions. And last but not least, a high dimensional feature space will lead to an overfitted classifier more likely than a low dimensional one does. The couple of problems related to high dimensional spaces of data are well known as '*curse of dimensionality*' [18].

Thus, approaching polytomous classification problems requires even more efforts in comparison to dichotomous ones. There are algorithms which are capable to learn multi-class-problems like decision-trees, some rule-learners, and in general instance-based learners, although many rule learning algorithms ought to learn a concept description based on positive and negative examples. Other algorithms are principally designed to distinguish two classes only, usually based on (mostly linear) discrimination or regression. Among them are so well-performing ones as support vector machines (SVM) [23], or the less recent perceptron [130] – not to forget the earliest formal approach to classification, *Fisher's linear discriminant analysis method* (LDA) [63]. In most cases also algorithms capable of handling multi-class problems will perform better over two than over many classes, due to the reasons mentioned above, even if they are able to treat polytomous problems. Furthermore, since especially the support vector machines are currently considered to belong to the best-performing classifiers and are therefore used preferably, the possibilities to applicate support vector machines to polytomous problems are an interesting topic of research. We will, however, address this problem not specifically for support vector machines only, but in the more general horizon of how to applicate two-class-classifiers to polytomous classification problems.

2.2 Approaches to the Reduction of Polytomies to Dichotomies

The world is not black and white – many real-world classification problems include more than two classes. In order to apply binary classifiers to multi-class-problems the latter are to be reduced to several dichotomous problems. For such a reduction there are basically three different ways one can think of, as long as one wishes to keep the number of involved classifiers limited[1]:

- learning for each class the discrimination between this class and all other classes (*one-versus-others*),

[1] For approaches involving more than a minimal number of classifiers, see chapter 3 concerning ensemble-methods.

- learning for each class the discriminations between this class and each other class (*all-versus-all*), or
- splitting the set of classes into two subsets, assigning each subset a class label $c \in \{1, -1\}$ as superclass, and learning the discrimination between this two superclasses, then repeating the process recursively for each of the subsets (*nested dichotomies*).

The first method results in n classifiers for n classes, the second results in $\binom{n}{2} = \frac{n \times (n-1)}{2}$, and the third results in $n - 1$ classifiers. Thus, the first and the third method seem preferable with respect to complexity at first glance, but the polytomous problem is not solved by simply reducing it to several dichotomous problems. The resulting classifiers will have to be combined in a proper way which is not always trivial. Also each method has advantages as well as disadvantages which we will now review shortly, partially based on [84].

2.2.1 One versus Others

The first of these methods, constructing one classifier for each class to learn discrimination between this class and all other classes, is the most simple one and also often referred to as the standard-method. This method is not claimed to be suggested by anyone in the first place. It is also called *'one-versus-rest'*, *'one-versus-all'* (one reads also *'against'* instead of *'versus'*), or *'one-per-class'*.

Consider for example the linear separation of two classes which can be generally formalized (see [39, p. 20]) by associating a weight vector w and a bias b to each class and assigning an example x to class 1, if

$$\langle w_1 \cdot x \rangle + b_1 \quad \geq \quad \langle w_{-1} \cdot x \rangle + b_{-1} \tag{2.1}$$

is fulfilled, to class -1 otherwise. Thus, the *one-versus-others*-solution of the n-class-problem with classes $i \in \mathcal{N} = \{1, \ldots, n\}$ is equivalent to assigning x to the class, where the maximum is reached for the respective weight vector and bias in

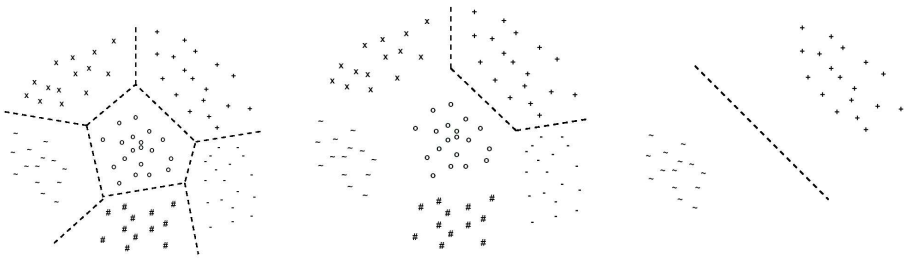
$$c(x) \quad = \quad \arg\max_{i \in \mathcal{N}} (\langle w_i \cdot x \rangle + b_i), \tag{2.2}$$

where a classifier for class i (thus providing weight vector w_i and bias b_i) is to be considered as trained with all of the examples of the i^{th} class with positive labels, and all other examples with negative labels.

Although *one-versus-others* is a very simple and also efficient method of reducing polytomies to dichotomies at first sight (the number of required classifiers being equal to the number of classes, that is of linear complexity, and thus any simple voting-scheme (e.g. Equation 2.2) for testing (or classifying, respectively) also being linear), it hides several pitfalls, partially depending on the structure of data.

Firstly, one has generally to assume that the classes are mutually exclusive. If they were not, one would have to build n *unlinked* classifiers, each deciding whether or not an instance belongs to the respective class. These n decomposed dichotomous problems would then be interesting, but not a decision over a recombination of all n classifiers. The problem will become more complicated, if the classes are somehow dependent, but this is far beyond the scope of our investigation. Thus, we will generally assume classes of a polytomous classification-problem to be mutually exclusive.

Secondly, one should assume classes to be unrelated. If several classes were related to each other, meaning they were more similar to one another than to other classes given in the namely dataset, the *one-versus-others* approach could easily become inappropriate.



(a) multi-class learning (b) one-versus-others (c) all-versus-all

Figure 2.1: Exemplary decision boundaries for a polytomous classification problem.

(a) One classifier separates all classes.

(b) n classifiers, each separating one class from all others, here: + against all other classes.

(c) $\frac{n\times(n-1)}{2}$ classifiers, each separating two classes, here: + against $\sim$.

In this example all dichotomous classification problems of the *all-versus-all* approach could be solved with a simple linear discriminant, while neither the multi-class approach nor the *one-versus-others* approach have a linear solution.
Figures taken from [69].

But even assuming independence and unrelatedness of classes, the respective 'others'-class could become very inhomogeneous. Thus, the separation would become very difficult, leading to increased complexity of the respective base classifiers. As a matter of fact, Hsu and Lin report an increased number of support vectors for most of the considered problems for the *one-versus-others* approach in comparison to *all-versus-all* leading also to an increased testing time [84]. See also Figure 2.1 for an illustration.

Furthermore, the resulting dichotomies will usually be very unbalanced in terms of size, since the others-class will contain $(n-1)$ as much instances as the respective single class on average. Thus, most learners might be biased to the others-class.

However, Rifkin [128] (repeated in [129]) claimed the *one-versus-others*-method to be not inferior to some other methods of combining dichotomous classifiers or generalization of certain binary classifiers for polytomous problems (given the base-classifiers being well-tuned regularized). This claim was primarily based on several experimental results using support vector machines and regularized least-squares classification as base classifier. Therefore, although it is supported by some evidence, it remains unclear, whether it holds in general. Furthermore, other types of base-classifiers (even worse-performing ones) might be interesting for the exploration of real-world data, since they could provide information regarding classification-criteria in a more human-readable form, e.g. as rules.

2.2.2 All versus All

The *all-versus-all*-method (also referred to as *one-versus-one*, *round robin*, *pairwise*, or *all-pairs*) builds one classifier for each pair of classes. Thus, a single classifier is trained with examples belonging to either one of the respective two classes (examples belonging to other classes being ignored) and it should also classify any instance during testing to either one of the classes it learned about.

Although there are $O(n^2)$ classifiers needed for an *all-versus-all*-approach, the performance is not necessarily inferior to an *one-versus-others*-approach, since the time-complexity for training is often super-linear dependent on the number of instances. Thus, each of the n *one-versus-others*-classifiers could require considerably more training time as one of the $\frac{n\times(n-1)}{2}$ *all-versus-all*-classifiers. Assuming m

instances for each of n classes, *one-versus-others* training-complexity would be

$$T_{\text{training}}(n, m) \quad = \quad n \times t(nm) \tag{2.3}$$

(i.e., involving $n \times m$ instances for training each of n classifiers, $t : \mathbb{N} \to \mathbb{R}_0^+$ being the time-complexity of training one single classifier dependent on the number of instances).

All-versus-all on the other hand has a time-complexity of

$$T_{\text{training}}(n, m) \quad = \quad \frac{n \times (n-1)}{2} \times t(2m). \tag{2.4}$$

If, however, t was linear, both approaches would be in $O(n^2 m)$, *all-versus-all* actually saving m instances overall in comparison to *one-versus-others*. With increasing complexity of t the advantage for *all-versus-all* becomes more distinct. (Fürnkranz [69] provides a more detailed discussion regarding the time required for training *one-versus-others* and *all-versus-all*, respectively.) Related to the decreased complexity of training for *all-versus-all* classifiers may be the fact that two single classes might be easier separable than one class is from all other classes, as it was stated above and illustrated by Figure 2.1.)

Rifkin's claim mentioned above was partially motivated by the efficiency of *one-versus-others* decomposition-methods. Whilst his claim may be true, its motivation seems not totally convincing in general, even though he was aware of the better training-efficiency of the *all-versus-all*-method [128, p. 179]. But it is noteworthy that, on the other hand, space-complexity could become a considerable problem (depending on the chosen base-classifier) in case of *all-versus-all* for many classes (as Rifkin also has noted at the same place) due to the squared number of required classifiers. Furthermore, the time-complexity of testing (which is often of greater importance to a user, as Rifkin also points out [129]) is generally advantageous for *one-versus-others*, although there are several testing procedures, that is different ways of combining the built and trained base classifiers, established for *all-versus-all*, which we will shortly recall.

2.2.2.1 Conjunction of Predictions

Knerr et al. [91] proposed a method for building and training a neural network designed to perform polytomous classification tasks where some classes are not linearly separable. This is sometimes referenced to as first method of a connection of *all-versus-all*-classifiers. However, the method is closely connected to the background of neural networks and, as a matter of fact, primarily tends to use *one-versus-others*. Nevertheless, for classes that are not linearly separable by *one-versus-others*, an *all-versus-all* classification is performed. Thus for an n-class problem where m classes are linearly separable via *one-versus-others*, *all-versus-all* is performed for $n-m = o$ classes. This procedure leads to $o - 1$ neurons for each of o classes discerning the respective class against the $o - 1$ other classes. The output of these neurons is directed to a layer of o AND-neurons (one for each of the remaining o classes). This is the output-layer for the *all-versus-all* related part of the network. Each of the output-neurons gets therefore possibly input of $o - 1$ pairwise-classifier-neurons (and, perhaps, from the m *one-versus-others*-neurons voting for 'others'). Since the output-neurons are AND-functions, the output for class i will be given, if all $o - 1$ pairwise classifiers discerning class i from some other class classify a given instance to class i (and m *one-versus-others*-neurons vote for 'others').

Of course the method of Knerr et al. is easily generalizable to any base classifier. The essence of the method is to vote finally for class i, if and only if all pairwise base-classifiers built by the *all-versus-all*-method for class i versus some other class

vote for class i. This is a very strict decision rule possibly allowing only very few wrong classifications. But on the other hand many instances will possibly not be classified at all (or labeled as unknown, respectively).

2.2.2.2 Voting

Apparently not being aware of the method proposed by Knerr et al. [91] Friedman proposed another approach to polytomous classification [68] which nevertheless overcomes the possible problems the earlier method was resulting in due to its strictness. Friedman does not demand a concordantly voting of all base classifiers related to a certain class, but simply counts the votes of the single classifiers and votes for the relative winner. Thus Friedman's method is also referred to as '*max wins*'-strategy.

Obviously *max wins* would classify some instances labeled as unknown by the method of Knerr et al. – not necessarily all of them, since the count of the votes could also result in a tie (which is not discussed by Friedman). This generosity could increase the true positive rate ($TP/(TP+FN)$). But on the other hand also the false positive rate ($FP/(FP+TN)$) could be increased, of course. However, the latter seems more unlikely and would rather be a problem of not well-tuned base classifiers than of Friedman's method.

2.2.2.3 Pairwise Coupling

Moreover, *max wins* was refined by Hastie and Tibshirani [81], resulting in the rather frequently used so called method of '*pairwise coupling*'. This method is based on estimated class probabilities. Some classifiers might provide class probabilities. For k-nearest-neighbors as well as for support vector machines Hastie and Tibshirani propose also approaches for deriving probabilities based on the classification. For coupling of probability estimates an iterative procedure is presented. Recently Wu et al. [153] extended this method.

The *pairwise coupling* method breaks the ties Friedman's method is possibly resulting in. But the error-rate is not always decreased in comparison to *max wins*. Furthermore, the time-complexity of the testing-procedure is increased by the procedure for coupling the probabilities which is to be performed for each instance separately.

2.2.2.4 Decision Directed Acyclic Graph

Platt et al. [124] proposed another testing procedure based on $\binom{n}{2}$ classifiers. Their method's aim is to increase speed during testing, since not all of the $\binom{n}{2}$ classifiers are to be considered. When the one classifier (trained e.g. for classes a versus b) on top classifies for a they conclude the instance would *not* belong to class b. According to this conclusion, only classifiers unrelated to class b are to be consulted in the remainder of the testing procedure. To obtain a decision the respective classifiers are therefore connected in a directed acyclic graph. Hence the method is referred to as *decision directed acyclic graph*, or DDAG for short (see Figure 2.2 for an illustration).

The idea is striking, since it overcomes also some open questions related to the *max wins* as well as to the *pairwise coupling* strategy: Could the instances of a single class tending to be closer on average in feature space to a randomly chosen instance than instances of the other classes receive more votes of classifiers unrelated to the actual class? This seems a relevant problem (discussed also in [81]) which is avoided by the negative decision.

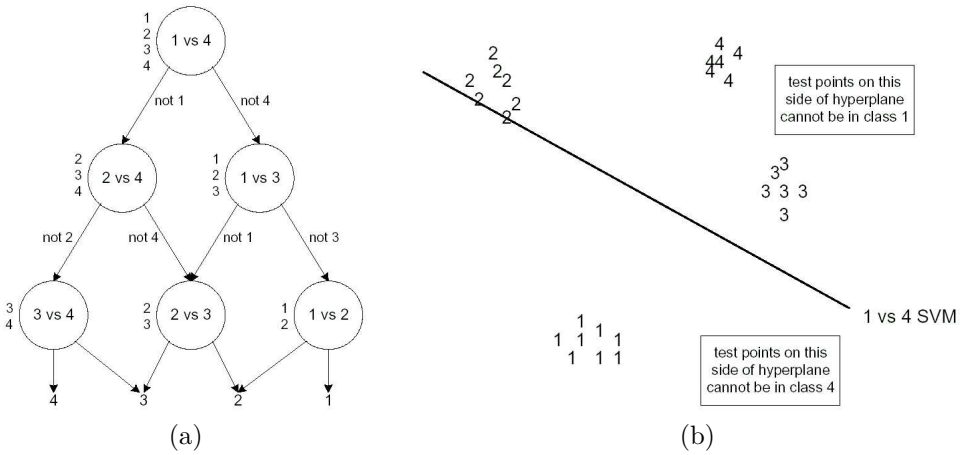


Figure 2.2: Illustration of a decision directed acyclic graph.

(a) The decision DAG for finding the best class out of four classes.

(b) The decision boundary related to a 1-versus-4 SVM. In *max wins* instances of classes 2 and 3, respectively, would have been voted for class 4. For DDAG the only conclusion for them would be not to belong to class 1.

Figures taken from [124].

One the other hand, if the one classifier on top would classify wrongly (meaning it would not recognize a given instance of class b to belong to class b, which would be rather unexpected, of course), for the remaining classifiers involved with class b during training would have arisen no opportunity of correcting the one wrong classifier.

The time-complexity for testing in a DDAG is linear, since only $n-1$ decision nodes will be evaluated in order to derive an answer for a problem with n classes.

2.2.2.5 Summary

While *max wins* and *pairwise coupling* are frequently used methods (despite their weaknesses), with DDAG an elegant and efficient method of *all-versus-all* classification was proposed. However, there are still $\binom{n}{2}$ classifiers to be built and trained. Furthermore, for classification accuracy of DDAG no superiority in comparison to *max wins* or *one-versus-others* is claimed [124]. Thus, a look at the third method of decomposition of polytomies may be interesting, although this last method is less frequently used.

2.2.3 Nested Dichotomies

Fürnkranz defines a decomposition of an n-class polytomy into $n-1$ dichotomies called '*ordered class binarization*', where the i^{th} classifier is trained with examples of class i as positive examples and the examples of classes $j > i$ as negative examples [69, Def. 3]. This technique could more generally be referred to as *nested dichotomies* [65]: The classes are splitted into two subsets, and each subset is recursively splitted further. Obviously the method described by Fürnkranz is a special case of nested dichotomies, the first of two subsets always containing exactly one class.

In terms of the number of required classifiers $(n-1)$ *nested dichotomies* improve over the other two methods. The complexity of a single classifier depends: There

might be as well classifiers discriminating several classes versus several others as classifiers discriminating one class versus one or several others. In case of '*ordered class binarization*' there are only *one-versus-others*-classifiers, where the size of the set of 'others' is continuously decreasing one by one. This special case is especially recommendable in case of ordered classes (as the method's name is already indicating), that is, the latter classes are including the earlier ones. Fox [65] gives as an example the level of education of adults, where a higher level of education also includes a former one. Generally, this is the case for classes resembling some sort of progress through the stages of a process.

When the classes are not ordered *nested dichotomies* are not generally recommendable since there are many different *nested dichotomies* for a single polytomous problem, each of them imposing a certain ordering. Thus, for unordered classes one could not possibly select a proper nesting of dichotomies.

However, in contrast to the other two decomposition-methods, note that *nested dichotomies* could be expected to improve in case of structured data if one is able to use the information of the structure for selecting the proper dichotomies.

2.3 Conclusion

We reviewed several approaches to decomposition of multi-class problems to several dichotomous problems of classification. The approaches presented so far were approaches that tend to use only a somehow minimal number of base classifiers, nevertheless exhaustively considering each class at least once. In the following chapter we will also take approaches into account that are not restricted in this way. However, for all of these approaches some properties can be formalized. To address the questions arising from the decomposition of polytomous classification problems more formally further on, we will firstly define some basic terms (according to [69]):

Definition 2.1 (Binarization)
*A **binarization** is a mapping of a multi-class learning problem to several two-class learning problems, that allows a prediction for the multi-class problem based on the predictions for all the two-class problems involved.*

As binarization procedures we discussed *one-versus-others*, *all-versus-all*, and *nested dichotomies*. The idea behind all binarizations is not to burden a single classifier with the task of discriminating several classes, but to hand down this task to the combination of several classifiers.

Definition 2.2 (Decoding)
***Decoding** is the derivation of a prediction for a multi-class problem from the predictions of the set of two-class classifiers built for a binarization of the multi-class problem.*

Decoding of *one-versus-others* and *nested dichotomies*, respectively, is straightforward. Several decoding procedures are established for *all-versus-all*.

Definition 2.3 (Base Learner)
*The learner originally providing predictions that a decoding could be based on is referred to as **base learner**.*

For a given problem, some binarizations as well as certain base learners will be more appropriate than others, depending on the time- and space-complexity for training and testing, respectively, one is willing to accept, and finally also depending on the structure of the given data. If data are structured in a very pronounced

manner and some classes are more closely related to each other than to other classes, certain simple strategies of binarization will become easily inappropriate or at least they might become more complicated.

On the other hand, data exhibiting pronounced hierarchical structure of classes are not always a curse. On the contrary, if one knows about the structure, one might be able to use this knowledge to improve effectiveness of classification. While all binarization-techniques presented so far possibly could be adapted to hierarchically structured classes, we choose *nested dichotomies* for further examination of adaptation to hierarchically structured classes, since for them this adaptation is most natural. Furthermore, *nested dichotomies* are more appropriate for hierarchically structured classes as a matter of principle, compared to *one-versus-others* and *all-versus-all*, respectively.

Later on we will particularly examine an adaptation of the *nested dichotomies*-binarization to hierarchically structured classes based on ensembles of nested dichotomies (ENDs). Therefore, we will shortly consider ensemble-techniques in machine learning in the following chapter before we recall ENDs afterwards.

Chapter 3

Approaches to Classification by Ensembles of Learners

One could describe the task of a single machine learning algorithm to search for the best hypothesis to explain the training data. Ensembles of learning machines thus are constructing a set of hypotheses based on training instances and then have these hypotheses vote for classification of unknown instances. Ensembles of hypotheses are often found to be more accurate than any single one of their hypotheses. Building ensembles of learners might especially be a good idea if the single learner is necessarily biased but not necessarily towards the correct solution. In this case ensembles could balance the differently biased single members to a relatively unbiased overall learner, as we will see. Recall e.g. the method of binarization by *nested dichotomies*, stated in the previous chapter 2, section 2.2.3. While any single binarization by *nested dichotomies* imposes a certain order on the classes of the polytomy, this bias could be compensated by several other binarizations using other *nested dichotomies* and therefore each assuming a different order of classes.

As a foundation for the following chapter 4, where especially ensembles of nested dichotomies will be represented, we will shortly and more or less abstractly survey ensemble-based approaches to machine learning. Unless other sources are cited we refer to the reviews [46], [49], and [146].

3.1 Ensembles in General

Recall that a classifier is to assign a class-label c to a vector of features $\vec{x}_i$, assuming some underlying 'true' function f such that $f(\vec{x}_i) = c_j$ for each training instance $(\vec{x}_i, c_j)$. The goal of any learning algorithm therefore is to find a good approximation h to f. The learned function h is referred to as *classifier*. It could also be understood as a *hypothesis* regarding the dependency of class c_j of the features $\vec{x}_i$. For any learning algorithm there is a certain space of hypotheses $\mathcal{H}$ the algorithm could represent, whereas many hypotheses may not be representable by a certain algorithm and thus they never could be used as an approximation to f. This restriction of any learning algorithm is also called its *bias*. Of course, any learning algorithm is to be biased to a certain degree in this notion since a totally unrestricted space of hypotheses is most likely to be infinite. Thus the search for the best hypothesis could be a never-ending quest. And even an infinite hypothesis space is still likely to be biased. An unrestricted space of hypotheses would also provide at least one hypothesis approximating f perfectly for the training data (by learning the classes by heart), but only poorly for any new data, and thus causing the learning algorithm to overfit.

However, the best hypothesis h for a given learning problem available in a restricted space of hypotheses may or may not be a good approximation to the 'true' function f. Thus, ensemble learning algorithms approach the solution of the classification problem in a different way, constructing a *set* (also *committee* or *ensemble*) of hypotheses and having those hypotheses 'vote' to classify a given instance $\vec{x}$. Different ensemble methods may differ as well with respect to the method of constructing the set as with respect to the procedure of voting. Given a set of hypotheses $\{h_1, \ldots, h_n\}$ for a dichotomous classification problem for the classes $\{+1, -1\}$, i.e., $h_i : \mathbb{R}^d \to \{-1, 1\}$, voting could be generally performed by choosing a set of weights $\{w_1, \ldots, w_n\}$ and constructing the overall classifier

$$\bar{h}(\vec{x}) \quad = \quad \sigma\left(\sum_{i=1}^{n} w_i h_i(\vec{x})\right), \tag{3.1}$$

where $\sigma : \mathbb{R} \to \{-1, 1\}$ is the tie-breaking *signum function* (Definition 3.1).

Definition 3.1 (Tie-Breaking Signum Function)
*The **tie-breaking signum function** $\sigma : \mathbb{R} \to \{-1, 1\}$ is given by:*

$$x \quad \mapsto \quad \begin{cases} 1 & \text{if} \quad x \geq 0 \\ -1 & \text{if} \quad x < 0 \end{cases}$$

Methods of voting for polytomous problems could resemble e.g. the methods described in the previous chapter 2, sections 2.2.2.1 through 2.2.2.4, and also in Equation 2.2. Many ensemble methods – as well as these standard binarization techniques – are constructing the ensemble members independently. Thus the voting is commonly unweighted, i.e.,$\forall i \in \{1, \ldots, n\} : w_i = 1$ in Equation 3.1 above. Otherwise one would have to apply a rule to define the weights of hypotheses. Also a second level classifier could be trained to learn an appropriate rule for weighting the hypotheses. Since these approaches are out of the scope of this investigation, we will focus on unweighted combination of hypotheses.

By combining several hypotheses, ensemble methods can partly overcome some general problems of those algorithms which are learning only one single hypothesis:

- If the space of hypotheses is too large with respect to the amount of available training instances there might be several different hypotheses giving equal accuracy on the training data. Choosing one of them exposes the learning algorithm to the risk of not performing well over future data. Voting of all these equally good classifiers can possibly reduce this risk. This problem is also known as the *statistical variance* of a learning algorithm.

- Another kind of *variance* arises when the learning algorithm is not guaranteed to find the best hypothesis within the hypothesis space, e.g. since this task is computationally intractable (thus this kind of *variance* is also referred to as *computational variance*). Heuristics to overcome the computational restrictions can get stuck in local optima and hence fail to find the optimal hypothesis. Obviously, trying several times reduces the risk of choosing the wrong local optimum.

- Finally, as we have indicated above, a learning algorithm might suffer from a too strong restriction of the hypothesis space, i.e., the hypothesis space $\mathcal{H}$ does not contain any good approximation to the 'true' function f. Thus the learning algorithm has a high *bias*. A weighted sum of hypotheses could expand the space of representable functions in some cases, so an ensemble might be able to find a better approximation of f than any of its components could ever do.

These improvements of ensembles over single classifiers are stated conditionally since there are mainly two conditions for an ensemble to improve over its individual members: The individual classifiers must be *accurate* and *diverse*:

Definition 3.2 (Accuracy of Classifiers)
A classifier will be ***accurate****, if its error rate is better than random guessing on new instances.*

Definition 3.3 (Diversity of Classifiers)
Two classifiers will be ***diverse*** *w.r.t. each other, if they make different errors on new instances.*

It is easy to understand why these two conditions are necessary and also sufficient. If several individual classifiers are not diverse, then all of them will be wrong whenever one of them is wrong. Thus nothing is gained by voting over wrong predictions. On the other hand, if the errors made by the classifiers were uncorrelated, more individual classifiers may be correct while some individual classifiers are wrong. Therefore, a majority vote by an ensemble of these classifiers may be also correct. More formally, suppose an ensemble consisting of k hypotheses, and the error rate of each hypothesis is equal to a certain $p < 0.5$ (assuming a dichotomous problem), though independently. The ensemble will be wrong, if more than $k/2$ of the ensemble members are wrong. Thus the overall error rate $\bar{p}$ of the ensemble is given by the area under the binomial distribution, where $k \geq \lceil k/2 \rceil$, that is for at least $\lceil k/2 \rceil$ hypotheses being wrong:

$$\bar{p}\,(k,p) \quad = \quad \sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} p^i (1-p)^{k-i}. \tag{3.2}$$

The overall error-rate is rapidly decreasing for an increasing number of ensemble members, as illustrated by Figure 3.1. However, the error-rate remains higher for even numbers than for the respective previous odd number, since ties are counted as errors. Thus an ensemble having $2k$ members will decide wrongly, if k members decide wrongly. So will an ensemble having $2k-1$ members, but for this one k is a greater fraction of the overall number of classifiers.

3.2 Methods for Constructing Ensembles

In order to construct an ensemble one could take several approaches. Any approach is to use accurate and diverse classifiers as members and to combine them tackling the general problems of statistical and computational variance as well as the representational bias. Generally one may recall several possibilities:

3.2.1 Bayesian Voting

The Bayes Optimal Classification is the most accurate available classifier under certain circumstances. It will choose the prediction of a class c, if this choice is maximizing the weighted sum of all hypotheses $h_i \in \mathcal{H}$, given the data $\mathcal{D}$:

$$P(c|\mathcal{D}) \quad = \quad \sum_{h_i \in \mathcal{H}} P(c|h_i) P(h_i|\mathcal{D}). \tag{3.3}$$

Since it depends on a complete enumeration of all possible hypotheses it is usually not applicable to problems where the space of hypotheses is not too small. Furthermore, it depends on reasonable prior probabilities of the hypotheses which are

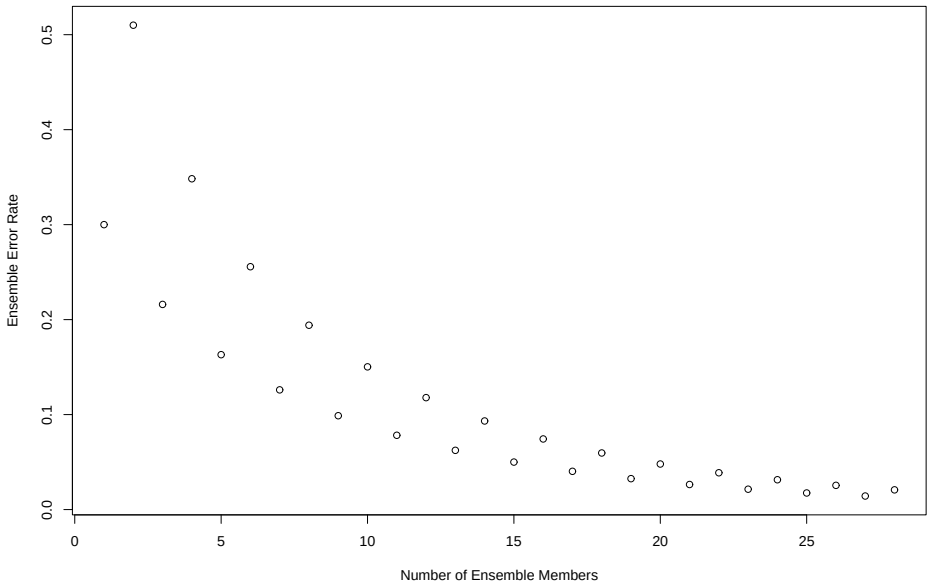


Figure 3.1: Ensemble-error-rate over number of ensemble-members.
Decreasing of ensemble-error-rate with increasing number of ensemble members according to Equation 3.2. For each ensemble-member an error of $p = 0.3$ is assumed. Obviously, for an even number of members the error-rate remains always higher than for the closest odd numbers, since ties are counted as errors.

often assumed in a simplified manner. Thus the 'Bayes Optimal Classifier' is often just not feasible in actual applications. (Besides the sources given above we refer also to [110, chapter 6] for Bayesian Learning.)

3.2.2 Resampling

Some learning algorithms are *unstable*, meaning the learned hypothesis undergoes major changes in response to even small changes in the training data. To build ensembles of diverse hypotheses a common method is to resample the training data several times and train an unstable algorithm for each sample. Best-known methods of this category are bagging [27], cross-validated committees [120], and boosting-methods (see also [82, chapter 10]). While bagging forms an ensemble by learning over bootstrap replicates of the training data (sampling with replacement) and cross-validated committees use subsets of data created by sampling without replacement, in boosting-methods the training instances get a different weighting at each iteration.

3.2.3 Feature Selection

In case of features of high dimensionality an ensemble approach could choose several subsets of features, simultaneously fighting the curse of dimensionality. Apparently these methods work well especially when the features are highly redundant.

3.2.4 Manipulating Output Targets

This general method provides also (like the two previous approaches) different datasets for training single classifiers. But instead of resampling or choosing subspaces,

now the class-labels are changed, i.e., the multi-class-problem is decomposed to several two-class-problems.

The concept of 'decomposition' has already been introduced in chapter 2. It could also be thought of as an ensemble method: Single classifiers are trained each for one of several dichotomies provided by a binarization of a polytomous classification-problem. The ensemble tackling the polytomous problem consists of all these single classifiers. For an ensemble method one could think of using more dichotomies as in the methods described above (chapter 2), or even less. This most naturally leads to the method of Error Correcting Output Coding (ECOC) [50] which will be further examined below, since it provides a general description that is also applicable to the method presented in the following chapter 4.

3.2.5 Injecting Randomness

Some of the hitherto described methods already use randomness, e.g. for resampling data or for choosing feature-subspaces by chance. But also some learning algorithms could be influenced directly by randomness, if they were based on initial values, e.g. initial weights of a neural network for the backpropagation-algorithm. If individual classifiers were initialized with different, i.e., random, values, they would be diverse. Thus it would make sense to build an ensemble of such classifiers.

3.2.6 Summary

Ensemble methods are well established for obtaining highly accurate classifiers as a combination of less accurate but diverse ones. Nevertheless, a unifying theoretical framework of ensemble methods has not yet been developed, although the topic can be considered to be one of the main directions in machine learning research currently.

However, there does not seem to be any general superiority of one single ensemble method. Which method can be expected to perform better strongly depends on the actual problem (as it holds true for any individual learner). So for a huge amount of training data randomized methods will usually perform better than resampling-methods, since bootstrap replicates of a large set will be very similar to the set itself and hence the diversity of individual hypotheses is not sufficiently supported. Boosting methods on the one hand will generally work badly for very noisy data, since they will weight the mislabeled noise more heavily, resulting in overfitting. On the other hand they address the representational problem very directly by optimizing the weighted vote and hence gain usually good results in case of low noise in data.

Initially we noted the possibility of averaging the bias of single *nested dichotomies*-binarizations by building an ensemble. The natural ensemble-method for doing this is evidently randomizing. We will keep this in mind and refer the respective method of ensembles of nested dichotomies, which was proposed recently [66], in the following chapter 4.

3.3 Ensembles as Binarization Technique

It was already noted (section 3.2.4) that ensembles built by output coding decomposition methods are also tackling the task of binarization of polytomous problems to several dichotomous ones. The most popular and best studied of these methods is error correcting output coding (ECOC) [50]. We will now shortly survey this method and a generalization of ECOC that provides the means to represent also binarization techniques that were presented in the previous chapter 2.

3.3.1 Error Correcting Output Codes

The idea of error correcting output codes (ECOC) is based on a comparison of *one-versus-others* and a *distributed output code* approach to multi-class problems [50]. The *distributed output code* approach was introduced by Sejnowski and Rosenberg [137]. They assigned a unique binary string of length n to each class. Then for each bit position in these binary strings a binary function is learned. Usually these binary functions are chosen to be meaningful properties in the domain. Often the properties related to a function are to be independent. For classifying new instances each of the n binary functions is evaluated resulting in an n-bit string s. The instance is then assigned to the class that has assigned the n-bit string closest to s according to a certain distance measure.

Dietterich and Bakiri [50] defined a common framework to compare both approaches, *one-versus-others* and *distributed output code*. Obviously, the n-bit strings of the latter approach can be represented by a matrix whose rows are corresponding to the classes and whose columns are related to the n binary functions corresponding to the bit positions in the strings. The one-versus-others approach can be represented similarly. For k classes the matrix will then have k rows and k columns. The entries on the diagonal will be 1. All other entries will be 0. That is, the bit string of length k corresponding to the value 2^{k-i} is assigned to class i. In both cases the bit string resulting from predictions of all binary classifiers is decoded to the class having the nearest of the bit strings assigned to. These bit strings assigned to classes are called 'output codes'.

As Dietterich and Bakiri now are pointing out, the prediction will become more stable on average for well distributed output codes. For a minimum distance d between any two output codes $\lfloor (d-1)/2 \rfloor$ erroneous binary classifiers can be corrected. Assuming the Hamming distance as distance measure for output codes, the code corresponding to the *one-versus-others* approach has a minimum distance of 2. Thus it cannot correct any errors. Since output codes for the *distributed output code* often are meaningful with bits corresponding to independent properties they usually will also be separated by small Hamming distances. This motivates the use of error correcting codes maximizing the Hamming distance between any two output codes. On the other hand the bits in error correcting output codes will not be meaningful any more, but constitute arbitrary disjunctions of the original k classes. The question remains then whether these arbitrary disjunctions are easier to learn than meaningful disjunctions. Dietterich and Bakiri report that the error correcting output codes improved the generalization performance of a system. But they note also that the binarizations based on error correcting output codes do not produce results that are easy to understand since the individual binary functions were indeed more difficult to be learned.

Note that the length l of codes and thus the number of employed binary classifiers can be increased. This will improve the performance unless the correlation of errors committed in each pair of bit positions will become to high. But for meaningless codes the errors committed by each of the l binary functions are substantially uncorrelated. This property explains why ECOCs do generally perform well and especially better than *distributed output code* where the bits are not generally independent [92].

In summary, for good error correcting codes there are mainly two properties required [51]:

1. **Row separation:** Each pair of codes should be well separated in Hamming distance.

2. **Column separation:** The single binary classifiers related to the bits of codes should be uncorrelated with respect to each other.

Note that ECOC as a binarization technique reduces the number of required base learners for k classes to $\lceil \log_2 k \rceil$ since this number is sufficient to create k different binary codes of length $\lceil \log_2 k \rceil$. Consider e.g. a problem of four different classes. The problem can be completely binarized by two dichotomies each separating two complementary subsets of classes from each other e.g. by the following decomposition matrix:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

However, it is recommendable to employ more than the minimal number of codes to improve performance since arbitrary splits of the set of classes can not generally be expected to be learned easily.

3.3.2 Generalization of ECOC

Mayoraz and Moreira [105] propose a generalization of ECOC by introducing a third possibility. The binarizations are now considered to map the classes of the original polytomy to $\{-1, 1\}$ instead of $\{0, 1\}$. So 0 can be assigned to classes not involved in the current task. This allows to consider also the *all-versus-all* binarization in the same framework. As another decomposition the *complete* set of non-trivial splits of the set of classes is evaluated by Allwein et al. [6] in the same framework. Generally, Allwein et al. did not observe an improvement in performance by using ternary codes. But different binarization techniques can be seen as special cases of generalized ECOC and thus be directly compared in a common theoretical framework. So by means of this generalization of ECOC following non-random decomposition matrices for an example consisting of four classes relate to special cases of binarization:

one-versus-others:

$$\begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix}$$

all-versus-all:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

minimal:

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{pmatrix}$$

complete:

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

The classes are mapped to $\{-1, 1\}$. An entry $m_{ij} = 0$ designates the omitting of instances belonging to class i during training of base learner j. Interestingly, not considering class probability estimates the decoding of an *all-versus-all* ECOC scheme resembles to the *max wins* strategy.

Introducing the third possibility, 0, does also require a redefinition of the distance measure. Allwein et al. [6] provide therefore two distance measures. The Hamming decoding $d_H : (\mathbb{R}^l, \mathbb{R}^l) \to \mathbb{R}$, assessing the distance between a row of the decomposition matrix m_i and the output $\vec{f}(x) = (f_1(x), \ldots, f_l(x))$ of the l binary classifiers $f_j, j \in \{1, \ldots, l\}$ for an instance x, is now defined as:

$$\left(m_i, \vec{f}(x)\right) \quad \mapsto \quad \sum_{j=1}^{l} \frac{1 - signum(m_{ij} f_j(x))}{2}, \tag{3.4}$$

where $signum : \mathbb{R} \to \{-1, 0, 1\}$ is given by:

$$x \quad \mapsto \quad \begin{cases} 1 & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x = 0 \\ -1 & \text{if} \quad x < 0 \end{cases} \tag{3.5}$$

Note that this definition additionally provides the possibility of employing abstaining classifiers, where $f_j(x)$ may return 0. The value 0, however, occurring either in m_{ij} or in $f_j(x)$ (or even in both), will contribute 1/2 to the sum. The finally predicted class i is obtained as

$$\arg\min_i d_H(m_i, \vec{f}(x)). \tag{3.6}$$

Seen in this way of representation, *all-versus-all* again seems more favorable than *one-versus-others*. Since the Hamming distance between any two rows in the decomposition matrix for *all-versus-all* is bigger than in the decomposition matrix for *one-versus-others*, the former will be able to compensate for more errors – unsurprisingly, since *one-versus-others* can not compensate any error at all.

Allwein et al. [6] consider also a second distance measure that enables the decoding procedure to take a confidence level of the binary classifiers into account by evaluating not only the sign of $f_j(x)$ but also the actual value. For an arbitrary loss function L the distance measure $d_L : (\mathbb{R}^l, \mathbb{R}^l) \to \mathbb{R}$ is then given as:

$$\left(m_i, \vec{f}(x)\right) \quad \mapsto \quad \sum_{j=1}^{l} L\left(m_{ij} f_j(x)\right), \tag{3.7}$$

and the class is analogous to Hamming decoding

$$\arg\min_i d_L(m_i, \vec{f}(x)). \tag{3.8}$$

Employing the proper loss-function L this distance function does allow also to decode the *all-versus-all* decomposition matrix according to the *pairwise coupling* procedure.

3.4 Conclusion

In this chapter we surveyed some general methods to build ensembles and considered what is required for ensembles to improve performance with respect to their members: accuracy and diversity. Based on these properties ensembles can be able to overcome one or several limitations of single learners, at least to a certain degree:

the statistical variance, the computational variance, and the error variance of single learners.

Furthermore, we have seen that ensembles can also be considered as a more general binarization technique. *All-versus-all* and *one-versus-others* were shown to be special cases of generalized ECOC. Other than Hamming distances allow thereby to decode according to arbitrary decoding procedures.

We will now have a closer look at the third binarization technique referred in the previous chapter: *systems of nested dichotomies*. This technique is especially suitable to build ensembles since different *systems of nested dichotomies* will generally expose a high error variance thus being diverse. Nevertheless, a single *system of nested dichotomies* can usually be expected to be accurate to a sufficient degree to build ensembles that improve performance with respect to their members.

Chapter 4

Ensembles of Nested Dichotomies

Systems of nested dichotomies (for convenience let us say also *nested dichotomies* meaning a *system of nested dichotomies*) were already introduced as a standard binarization technique (see chapter 2, section 2.2.3). We will now recall this method in more detail (based on [65], and [66]) in order to consider *ensembles of nested dichotomies*, a method which was recently proposed by Frank and Kramer [66].

4.1 Nested Dichotomies revisited

Any of the binarizations mentioned in chapter 2 provides a set of dichotomies derived from the original polytomy. The dichotomies will become nested, if they are produced by successive binary partitions of the set of classes of the polytomy. Thus, by nesting the dichotomies one derives a binary tree that divides the set of classes C_i associated with the internal node i into two disjoint subsets C_{i_1} and C_{i_2}, respectively. These two subsets taken together contain all the classes in C_i. The nested dichotomies' root node contains all the classes of the corresponding polytomous classification problem, whereas each leaf node contains a single class. Thus, for an n-class problem, there are n leaf nodes and $n-1$ internal nodes. Two examples of systems of nested dichotomies for a polytomy consisting of four classes are illustrated by Figure 4.1.

Pseudocode for a procedure

$$f_{ND} : \text{Class list} \quad \rightarrow \quad \text{Nested Dichotomies} \tag{4.1}$$

to derive a random system of nested dichotomies for a given class-list is provided in Table 4.1.

A classifier based on such a tree structure is built as follows: at every internal node the instances pertaining to the classes associated with that node are stored, and no other instances. Then the set of classes pertaining to each node is partitioned into two subsets so that each subset holds the classes associated with exactly one of the node's two successor nodes. And finally a binary classifier is built as base learner for the resulting dichotomous problem for each internal node.

A binarization by means of nested dichotomies requires the base learner to produce class probability estimates. But this is not a severe limitation, given that most practical learning algorithms are able to do. Otherwise there are possibilities to suit an algorithm to this requirement although this will not always be trivial. Besides, also the widely used voting-technique *pairwise coupling* for the *all-versus-all* binarization (chapter 2, section 2.2.2.3) requires this property of its base learners.

```
PROCEDURE insert(Class, Index, Tree)
{
    Subtree ← subtree of Tree at Index;
    Replace node Index in Tree with (Subtree, Class);
    RETURN Tree;
}

PROCEDURE f_ND(ClassList)
{
    IF length(ClassList) < 3 THEN
        RETURN ClassList;
    ELSE
        (First, Second|RestList) ← ClassList;
        Tree ← (First, Second);
        FOR i ← 3 TO length(ClassList) DO
            NextClass ← i^th element of ClassList;
            Index ← random number r : 0 < r < 2i − 3;
            Tree ← insert(NextClass, Index, Tree);
        RETURN Tree;
}
```

Table 4.1: Pseudocode for deriving a system of nested dichotomies for a set of classes.
The pseudocode describes a procedure

$$f_{ND} : \text{Class list} \quad \rightarrow \quad \text{Nested Dichotomies}$$

to derive a random system of nested dichotomies for a given list of classes. Classes are inserted at random into a growing binary tree.

Since the dichotomies are nested, they are statistically independent [65]. Thus decoding is straightforward: Given the base learners are providing class probability estimates one could easily obtain class probability estimates also for the original polytomous classification problem simply by multiplying the probability estimates obtained from the two-class models along the path from the root to a leaf. Hence the probability for an instance x to belong to a class c is conditioned by the probabilities to belong to all sets of classes containing c.

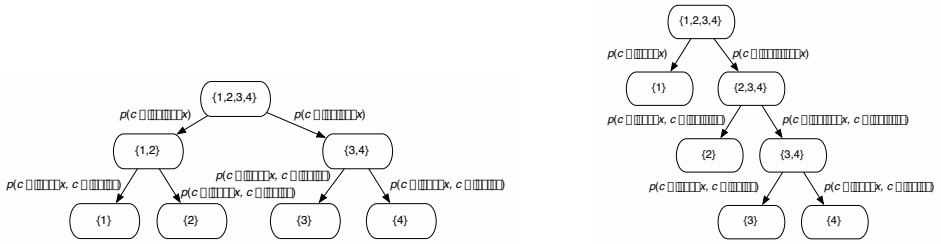
More formally, as pointed out in [66], let C_{i_1} and C_{i_2} be the two subsets of classes generated by a split of the set of classes C_i at internal node i of the system of nested dichotomies, i.e., C_{i_1} and C_{i_2} are associated with the successor nodes of node i, and let $p(c \in C_{i_1}|x,\, c \in C_i)$ and $p(c \in C_{i_2}|x,\, c \in C_i)$ be the conditional probability distribution estimated by the model at node i for a given instance x. Then the estimated class probability distribution for the original polytomous classification problem is given by

$$\begin{aligned} p(c = m|x) \quad = \quad & \prod_{i=1}^{n-1} (I(m \in C_{i_1})\, p(c \in C_{i_1}|x,\, c \in C_i) \; + \\ & I(m \in C_{i_2})\, p(c \in C_{i_2}|x,\, c \in C_i)), \end{aligned} \tag{4.2}$$

where I : boolean $\rightarrow \{0, 1\}$ is the *indicator function* (Definition 4.1), and the product is over all the internal nodes of the tree.

Definition 4.1 (Indicator Function)
*The **indicator function** I : boolean $\rightarrow \{0, 1\}$ is given by:*

$$c \quad \mapsto \quad \begin{cases} 1 & \text{if} \quad c \\ 0 & \text{if} \quad \neg\, c \end{cases}$$



(a) (b)

Figure 4.1: Two different systems of nested dichotomies for a classification problem with four classes.
(Figures taken from [66]).

Since $I(m \in C_{i_s})$ is 0 for all nodes, that are not associated with class m, Equation 4.2 is equivalent to an evaluation of the path from the root to the leaf associated with class m by multiplying together the probability estimates encountered along that path.

Consider Figure 4.1 which shows two of the 15 possible systems of nested dichotomies for a polytomous classification problem consisting of four classes. It can be seen easily that different systems of nested dichotomies may most likely provide different class probability estimates. So, using the tree in Figure 4.1 (a), the probability of class 4 for an instance x is given by:

$$\begin{aligned} p_a(c = 4|x) &= p_a(c \in \{3,4\}|x) \times \\ &\quad p_a(c \in \{4\}|x, c \in \{3,4\}). \end{aligned}$$

Based on the tree in Figure 4.1 (b) on the other hand, the probability of class 4 for the very same instance x is:

$$\begin{aligned} p_b(c = 4|x) &= p_b(c \in \{2,3,4\}|x) \times \\ &\quad p_b(c \in \{3,4\}|x, c \in \{2,3,4\}) \times \\ &\quad p_b(c \in \{4\}|x, c \in \{3,4\}). \end{aligned}$$

The class probability estimates obtained from different systems of nested dichotomies can be expected to differ, since they are based on different dichotomous classification problems. Nevertheless, if there are no a priori reasons to prefer a particular system of nested dichotomies, then there will also be no reasons to trust one of the different class probability estimates more than the others. Thus, both of the trees given in Figure 4.1 represent equally valid class probability estimators – as do the 13 other possible systems of nested dichotomies that could be generated for a four-class problem. Therefore all trees are to be treated as equally likely.

4.2 Ensembles of Nested Dichotomies

Based on this reasoning Frank and Kramer recently proposed an ensemble method for nested dichotomies (*Ensemble of Nested Dichotomies* or END for short [66]), forming overall class probability estimates by averaging the estimates obtained from different trees. But even for a moderate number of classes such an ensemble could not contain all possible systems of nested dichotomies, since the number of possible systems of nested dichotomies grows more than exponentially:

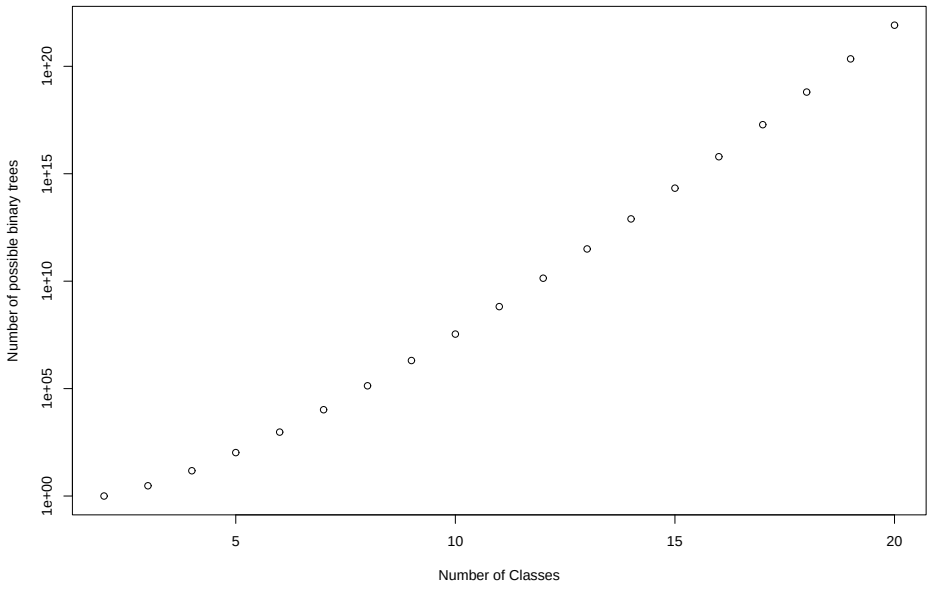


Figure 4.2: Growth of the number of possible binary-trees for a multi-class problem of n classes.
According to Lemma 4.1 there are $(2n-3)!!$ possible systems of nested dichotomies for an n-class problem, thus the number grows more than exponentially.

Lemma 4.1 (Growth of Number of Possible Nested Dichotomies) *For a set consisting of n classes there are*

$$\begin{aligned} T(n) &= (2n-3) \times T(n-1) \\ T(1) &= 1 \end{aligned}$$

possible ***systems of nested dichotomies****.*

Proof.

base case: For a one-class problem the tree has only one node, that is the root and the only leaf, representing the only class.

induction case: There are $(n-1)+(n-2) = 2n-3$ distinct possibilities to add a new class into a tree that was already built for $n-1$ classes, one for each of the $n-1$ leaf-nodes and the $n-2$ internal nodes.

□

Expanding the recurrence relation given in Lemma 4.1 results in

$$T(n) = (2n-3) \times (2n-5) \times \ldots \times 3 \times 1.$$

Using the double factorial, this could also be written as

$$T(n) = (2n-3)!!.$$

For a raising number of classes it becomes therefore infeasible to consider all possible systems of nested dichotomies even for problems with a moderate number of classes,

as is illustrated in Figure 4.2. Frank and Kramer therefore suggest to build the ensemble of nested dichotomies by taking a random sample from the space of all distinct trees for a given n-class problem (based on sampling with replacement).

The uniformity of the sampling process ensures these averages to form an unbiased estimate of the estimates that would have been obtained by building the complete ensemble of all possible distinct nested dichotomies for a given n-class problem. The ensemble could also be expected to perform better than any single system of nested dichotomies because the respective nested dichotomies are diverse, since they are created by random inserts of classes into the growing binary tree. The accuracy of the ensemble members heavily relies on the base learners and the difficulty of the dichotomous classification problems, thus the method should suffer no loss with respect to other binarization methods.

4.3 Performance of ENDs

Frank and Kramer [66] found ENDs to perform equally well in terms of accuracy with respect to other ensemble methods for multi-class learning in most cases. In many cases they are reported even to improve significantly.

As an upper bound for the training time of one system of nested dichotomies Frank and Kramer state the time it takes to build a classifier based on the *one-versus-others* binarization (see chapter 2, section 2.2.1). However, at least for certain average cases this seems not to be the lowest upper bound. While *one-versus-others* always requires n classifiers for an n-class problem, a system of nested dichotomies requires always $n-1$ base learners. Furthermore, for *one-versus-others* each base learner is usually trained using the instances of all classes, while the base learners of nested dichotomies use the less classes (and therefore also the less instances) the deeper they are nested. Since for most base learners training time is dependent from the number of instances in a worse than linear relation, nested dichotomies should gain a clear advantage even in the worst case when the tree of nested dichotomies is degenerated to a list. Assuming the training time given by $t : \mathbb{N} \to \mathbb{R}_0^+$ with respect to the number of instances used for training and nm training instances being equally distributed over n classes (thus being m instances for each class), the worst-case of training time is given by:

$$T_{\text{training-wc}}(n,m) = \sum_{i=2}^{n} t(im). \tag{4.3}$$

The best-case of training time occurs for a perfectly balanced tree where the number of training instances is divided by 2 for each level of the tree, that is:

$$T_{\text{training-bc}}(n,m) = \sum_{i=2}^{n} t\left(\frac{n}{2^{\lceil \log_2 i \rceil - 1}} m\right). \tag{4.4}$$

Recall the training time for the *one-versus-others* approach as given in Equation 2.3 is

$$T_{\text{training}}(n,m) = \sum_{i=1}^{n} t(nm).$$

Thus the training time for training an END with k members is bounded from above by $k \times T_{\text{training-wc}}$ as given by Equation 4.3, assuming the instances being equally distributed over all classes. But even when this assumption does not hold, the worst case training time for a system of nested dichotomies will be less than the training time of an *one-versus-others* binarization, since only the root of the

respective binary tree is involved with all instances, while any other node is to be trained with only a fraction of them.

4.4 Representation of ENDs by ECOCs

In chapter 3, section 3.3, we considered ECOC as a general framework to describe binarization techniques. Of course, also nested dichotomies can be defined by decomposition matrices, as was already stated by Frank and Kramer [66]. Recall the examples given in Figure 4.1. The respective decomposition matrices are given as follows:

(a)

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ -1 & 0 & 1 \\ -1 & 0 & -1 \end{pmatrix}$$

(b)

$$\begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & -1 & 1 \\ -1 & -1 & -1 \end{pmatrix}$$

Defining the proper decoding for these decomposition matrices as resembling to the decoding of nested dichotomies would require primarily a loss-function based decoding as defined in Equation 3.7. Moreover the loss-function to be employed by the decoding procedure has to be dependent on the current system of nesting and the distances provided by the loss-function for matrix (a) would have to be different from the distances provided for matrix (b).

4.5 Conclusion

We considered *systems of nested dichotomies* as a binarization technique. Nested dichotomies impose a certain order on the set of classes. A single random system of nested dichotomies can therefore be expected to be quite biased. Nevertheless a single system employed as binarization method usually can reach a reasonable degree of accuracy.

Even for a moderate increase of number of classes the space of possible systems of nested dichotomies grows rather fast. This involves also a high error variance for a set of moderate size of randomly chosen systems of nested dichotomies. Thus nested dichotomies are ideal components to build an ensemble. Concordant with theory, ensembles of nested dichotomies were shown to work well on a range of different data compared to other binarization techniques as *all-versus-all*, *one-versus-others*, and ECOC (random and exhaustive) as well as to multi-class methods [66].

Chapter 5

Incorporating Domain Knowledge into Learning Schemes

Any machine learning scheme is usually based on some assumptions concerning the hypotheses to be learned. Thus it is e.g. biased to search a hypothesis only within a certain subspace of the general hypothesis space. Indeed, learning without any bias is futile [109]. However, choosing a hypothesis space that is biased in a supporting direction is mainly a question of considering domain knowledge.

5.1 Different Possibilities for Incorporation of Domain Knowledge

First of all a user of machine learning tools conducting an experiment does implicitly incorporate domain knowledge by selecting the machine learning scheme that is most appropriate to the explored data.

On the other hand there is a broad variance of machine learning schemes allowing the incorporation of domain knowledge explicitly, as methods of inductive logic programming or Bayesian learning. Schapire et al. [134] proposed also a method to incorporate prior knowledge into boosting.

Furthermore, some algorithms may also take advantage from more specifically defined attributes. So, attributes could not only be discerned into numerical and nominal attributes, they could also exhibit an ordering of values. And such an ordering could either include a zero point (e.g. time intervals) or not (e.g. dates), it could either be linear (as Euclidean distances or educational degrees) or circular (as angles or the human perception of colors). Moreover several attributes may be related with each other, semantically, causally, or functionally. Or attributes are weighted based on domain knowledge. Besides, there could (and should) expert knowledge already be incorporated in extracting the features for a certain learning problem.

Reinforcement learning is an exemplary field of research, where possibilities for incorporating domain knowledge into the learning procedure have been discussed since several years. So the consideration of hierarchical structures is often more appropriate to complex tasks than a flat search space. It is possible to define the hierarchies in advance, as described in a general manner by Dietterich [45], [47], and [48]. For learners that have access to a complete and correct domain theory, explanation-based learning can be used to allow more appropriate generalizations

of single experiences by a reinforcement learner [52] or to improve learning rate and asymptotic performance [139]. This again can be adapted to hierarchies of goals [141].

However, there is no general theory of incorporation of domain knowledge developed so far.

5.2 Approaches Considering Hierarchically Structured Classes

A topic of special interest is taking into account relations between classes. Often there are subsets of classes whose members are more similar to one another than to classes that are not members of the respective subset. This means, the classes are organized in a hierarchy. The usage of simple binarization schemes (chapter 2) is based on the assumption that the classes are not belonging to a natural hierarchy. One would expect especially the *one-versus-others* binarization to suffer loss of accuracy, if some classes were closer to one another than to others in the respective feature-space, as also Rifkin [129, p. 34] predicts.

But it is possible to incorporate the knowledge concerning a hierarchical structure exhibited by the classes into the learning procedure. A few years ago this idea was introduced in text mining. Dumais and Chen e.g. performed a one-versus-others classification for a level-wise grouping of classes [58]. This approach was based on several efforts to establish and make use of hierarchical structure on text documents (confer the referenced related work in [58]). Interesting is also the work of Schubert et al. concerning classification of proteins with respect to their function [136]. While the emphasis in these works is put on use of hierarchical structures for sake of efficiency, we will be more interested in increase of effectiveness in terms of accuracy by incorporating knowledge concerning hierarchically structured classes: Building a classifier with respect to a predefined hierarchy is motivated for us mainly by the expectation to restrict the hypothesis space in a suitable way and therefore to reduce the error variance exhibited by the respective learners.

5.3 Conclusion

We considered possibilities to incorporate domain knowledge into learners. Of interest is thereby especially to consider pronounced hierarchical relationships between classes. We are aware of several approaches tackling this task. Most of them, however, are especially concerned with text classification problems (except for [136]). They also seem specialized to specific data. Yet particularly biology provides many different fields of application where to take into account a hierarchy of classes can be considered to be more appropriate to the respective problem. As a matter of fact, in the realm of scientific biology hierarchies are found since the days of Aristotle.

Furthermore the approaches referred above mainly focus on improving efficiency by making use of a hierarchy of classes. But obviously a properly defined hierarchy will restrict the hypothesis space in a suitable way. Thus the error variance of any learning scheme can be expected to decrease by incorporation of knowledge concerning the respective hierarchy.

We therefore consider a more general framework for hierarchical classification to be of interest. In the next part we are going to develop a method for classification where classes are related in an arbitrary hierarchy. This method will unite the fields that were introduced in this part since it is on the one hand based on ENDs, an ensemble approach tackling binarization. On the other hand it incorporates domain

knowledge concerning hierarchical structure of classes. Note that our method is motivated by biology, but it is not restricted to biological applications.

Part II

Ensembles of Hierarchically Nested Dichotomies

Chapter 6

Hierarchical Classification Using Ensembles of Nested Dichotomies

In the previous part we have reviewed some approaches to several different tasks for machine learning: Applying binary classifiers to a polytomous classification problem (chapter 2), building ensembles to improve accuracy of classification (chapter 3), and incorporating domain knowledge into the classification method, which is also expected to improve accuracy (chapter 5). The first and second of those tasks was already combined in the method END, which was introduced in chapter 4.

We are now going to propose a new machine learning method which is attacking those tasks in combination: We build ensembles of nested dichotomies dependent on domain knowledge concerning the relation between classes. That is, we consider classification problems over classes, where some classes are stronger related to each other than to some other classes. It was already shown, that nested dichotomies naturally impose such relations to the classification problem they are applied to, even if there is no structure of that kind between the classes. In general, this is a drawback for nested dichotomies. But it could turn out to be an advantage, if one were able to choose the correct subset of systems of nested dichotomies for a given classification problem over structured classes. Unless a given hierarchy describes the actual relationships between classes essentially wrongly the respective subset of nested dichotomies can be expected to exhibit a reduced error variance compared to an unrestricted sample of nested dichotomies of the same size.

The outline of this chapter is as follows: We will first give an idea of classification problems concerning classes which are structured in the notion given above. In general, such a structure can be referred to as a hierarchy of classes. A couple of definitions for handling hierarchies of classes will be provided and we will consider how to represent a hierarchy as a binary tree. This will lead us most naturally to a method for building systems of nested dichotomies that are adapted to the class hierarchy. Based on these considerations we will present the new machine learning method, ensembles of hierarchically nested dichotomies, which is most suggesting given the previous considerations.

6.1 Hierarchies of Classes

In real world tasks of machine learning one often encounters classification problems concerning a set of classes, where the instances of some subsets of classes are quite

more similar to each other than they are to instances belonging to other classes not contained in the respective subset. It is also possible to consider single classes to be more similar to certain subsets than to others, that is, some subsets of the original set of classes may contain only one class. On the other hand some subsets may contain even smaller subsets instead of single classes. We refer to such systems of nested sets as *hierarchies of classes*. To define a *hierarchy of classes* we are first to redefine the notion of a set of classes inductively (assuming sets of classes never to be empty):

Definition 6.1 (Higher Order Set of Classes)
*A **higher order set of classes** contains either exactly one class or several higher order sets of classes.*

1. *If a **higher order set of classes** represents a single class, it will contain exactly one class. It is then of order one.*

2. *If a **higher order set of classes** represents $n > 1$ classes, it will contain $m : 2 \leq m \leq n$ disjoint **higher order sets of classes**, whose altogether represent n classes. One could define its order as the maximum order of its members plus one.*[1]

*A set of classes $\mathcal{S}$ is corresponding to a **higher order set of classes** $\mathcal{H}$ and vice versa, if and only if it contains exactly those classes, which are represented by the higher order set of classes $\mathcal{H}$ or by those higher order sets of classes $\mathcal{T}$, where a sequence of length $i \geq 0$ of higher order sets of classes $\{\mathcal{I}_j\}_{j=1,\ldots,i}$ exists such that*

$$\mathcal{T} \in \mathcal{I}_1 \in \ldots \in \mathcal{I}_i \in \mathcal{H}.$$

*Then we say $\mathbf{S}(\mathcal{H}) := \mathcal{S}$ to be the **set of classes corresponding** to $\mathcal{H}$, and two **higher order sets of classes** $\mathcal{H}_i$ and $\mathcal{H}_j$ are said to be **disjoint**, if and only if*

$$S(\mathcal{H}_i) \cap S(\mathcal{H}_j) = \emptyset.$$

In the remainder we will often use a transitive relation of membership on higher order sets of classes. We therefore define:

Definition 6.2 (Membership on Higher Order Sets of Classes)
We define a transitive relation of membership in higher order sets of classes by $\mathcal{M} \subseteq$ (higher order set of classes $\times$ higher order set of classes) such that $(\mathcal{R}, \mathcal{S}) \in \mathcal{M} \wedge (\mathcal{S}, \mathcal{T}) \in \mathcal{M} \Rightarrow (\mathcal{R}, \mathcal{T}) \in \mathcal{M}$. More formally $\mathcal{M}$ is inductively given by:

$$\begin{aligned}
&\forall\, \mathcal{H}_i, \mathcal{H}_j \text{ higher order sets of classes}: \\
&(\mathcal{H}_i, \mathcal{H}_j) \in \mathcal{M} :\Longleftrightarrow \\
&\qquad \mathcal{H}_i \in \mathcal{H}_j \\
&\quad \vee \quad (\exists\, \mathcal{H}_m : \mathcal{H}_i \in \mathcal{H}_m \wedge (\mathcal{H}_m, \mathcal{H}_j) \in \mathcal{M})
\end{aligned}$$

Obviously then the following statement holds true:

$$(\mathcal{H}_i, \mathcal{H}_j) \in \mathcal{M} \Rightarrow S(\mathcal{H}_i) \subseteq S(\mathcal{H}_j)$$

Let us note that the relation between sets of classes and higher order sets of classes is not bijective. Although

$$S : \text{higher order sets of classes} \rightarrow \text{sets}$$

[1] Its order may also well remain undefined, since we will not hinge on the exact order of any higher order set of classes.

is an injective relation, S^{-1} is not defined since there could possibly be composed several distinct higher order sets of classes for a single set of classes. However, the higher order set of classes corresponding to a set of one class is defined and will be identical with the respective set. Given this inductive notion of higher order sets of classes we define:

Definition 6.3 (Hierarchy of Classes)
*A **hierarchy of classes** corresponding to a set of one class is identical to the respective set.*

*For a set of classes $\mathcal{C}$ where $|\mathcal{C}| > 1$ a **hierarchy of classes** $\mathcal{H}$ is a set of **hierarchies of classes**, where its members correspond to disjoint subsets $\mathcal{S}_i \subset \mathcal{C}$, $\mathcal{C} = \bigcup_i \mathcal{S}_i$. That is a **hierarchy of classes** corresponds to a higher order set of classes that contains only disjoint members each fulfilling this same condition.*

Furthermore will each subset $\mathcal{S}_i$ contain only classes which are more similar to one another than they are to classes not contained by the respective subset, given an arbitrary similarity measure.

More formally: Let

$$s : class \times class \to \mathbb{R}$$

*be a similarity measure. Let $\mathcal{S}$ be an arbitrary subset of the set of classes $\mathcal{C}$ ($\mathcal{S} \subseteq \mathcal{C}$). Then a higher order set of classes $\mathcal{H}$ is a **hierarchy of classes** which is defined w.r.t. s for $\mathcal{C}$, if and only if:*

1. $S(\mathcal{H}) = \mathcal{C}$, *and*
2. $\forall\, (\mathcal{H}_i, \mathcal{H}_j), \mathcal{H}_i \in \mathcal{H}, \mathcal{H}_j \in \mathcal{H}$:

$$S(\mathcal{H}_i) \cap S(\mathcal{H}_j) = \emptyset, \text{ and}$$

3. $\forall\, \mathcal{H}_\mathcal{S}, S(\mathcal{H}_\mathcal{S}) = \mathcal{S}, \mathcal{H}_\mathcal{S} \neq \mathcal{H}$:

$$\begin{aligned}(\mathcal{H}_\mathcal{S}, \mathcal{H}) \in \mathcal{M} \quad \Rightarrow \quad & \forall\, (c_i, c_j), (c_i, c_m), \\ & (\{c_i\}, \mathcal{H}_\mathcal{S}) \in \mathcal{M} \wedge (\{c_j\}, \mathcal{H}_\mathcal{S}) \in \mathcal{M} \wedge (\{c_m\}, \mathcal{H}_\mathcal{S}) \notin \mathcal{M} : \\ & s(c_i, c_j) > s(c_i, c_m).\end{aligned}$$

Definition 6.2 provides the means to define a reflexive, antisymmetric and transitive relation $\succeq$ as a subset of the Cartesian product of hierarchies of classes (*hierarchies of classes* $\times$ *hierarchies of classes*), that is hierarchies of classes can be partially ordered:

Definition 6.4 (Partially Ordering of Hierarchies of Classes)
A hierarchy of classes $\mathcal{H}$ is less general than or equal to another hierarchy of classes $\mathcal{K}$, if and only if all members of $\mathcal{K}$ are also members of $\mathcal{H}$. More formally this is denoted as:

$$\mathcal{H} \succeq \mathcal{K} \quad :\Longleftrightarrow \quad \forall\, \mathcal{K}_m : (\mathcal{K}_m, \mathcal{K}) \in \mathcal{M} \Rightarrow (\mathcal{K}_m, \mathcal{H}) \in \mathcal{M}.$$

The relation $\succeq$ is reflexive, antisymmetric and transitive. Antisymmetry implies:

$$\mathcal{H} = \mathcal{K} \quad \Longleftrightarrow \quad \mathcal{H} \succeq \mathcal{K} \ \wedge \ \mathcal{K} \succeq \mathcal{H}.$$

Other potentially useful relations can easily be based on this partial order:

$$\begin{aligned}\mathcal{H} \succ \mathcal{K} \quad &:\Longleftrightarrow \quad \mathcal{H} \succeq \mathcal{K} \ \wedge \ \mathcal{H} \neq \mathcal{K} \\ \mathcal{H} \prec \mathcal{K} \quad &:\Longleftrightarrow \quad \mathcal{K} \succ \mathcal{H} \\ \mathcal{H} \preceq \mathcal{K} \quad &:\Longleftrightarrow \quad \mathcal{K} \succeq \mathcal{H}\end{aligned}$$

Obviously a hierarchy of classes corresponds to a special system of nested sets and could thus be represented as an unordered tree. For the sake of further analysis we state explicitly:

Lemma 6.1 (Tree Representation of a Hierarchy of Classes)
The relation between hierarchies of classes and unordered trees is a bijection.

Proof.

injectivity The representation of a hierarchy of classes $\mathcal{H}$ for a given set of classes $\mathcal{C}$ as a tree is straightforward: The root of the tree is to be associated with the original set of classes $\mathcal{C}$, each node is to be associated with a subset $\mathcal{S}_i \subset \mathcal{C}$ which corresponds to a hierarchy of classes $\mathcal{H}_i \in \mathcal{H}_j$, when its parent is associated with the subset $\mathcal{S}_j \subset \mathcal{C}$ corresponding to the hierarchy of classes $\mathcal{H}_j$. Then the following statement obviously holds true:

$$(\mathcal{H}_i, \mathcal{H}_j) \in \mathcal{M} \wedge \forall\, \mathcal{H}_m \neq \mathcal{H}_i, (\mathcal{H}_m, \mathcal{H}_j) \in \mathcal{M} : (\mathcal{H}_i, \mathcal{H}_m) \notin \mathcal{M}.$$

Furthermore each node gets a child for each element of the hierarchy of classes $\mathcal{H}_i$ related to the subset $\mathcal{S}_i \subset \mathcal{C}$ it is associated with. Thus the classes will be represented as sets of classes of size one by the leaves of the tree representing the hierarchy of classes $\mathcal{H}$.

surjectivity Obviously a tree whose leaves are associated with disjoint sets of classes of size one can also be represented as hierarchy of classes. A node will then represent the set of the items represented by its children, and all these sets will be disjoint w.r.t. their siblings.

□

For a tree constructed as described in the proof of Lemma 6.1, obviously the children of an inner node $\texttt{node}_i$ will be more similar to one another than they will be to siblings of $\texttt{node}_i$. Each leaf will correspond uniquely to a certain class. A single class will not occur in more than one leaf.

Of course a hierarchy of classes will be defined, even if there is no inherent structure in the set of the classes. The hierarchy will then be a tree of depth one, since all the classes will be children of the root. Such a hierarchy is flat, and it makes no sense to use a flat hierarchy, since it will not provide any information which is not already provided by the set of classes.

So consider a simple example for a pronounced hierarchy: The data presented in Figure 6.1 consist of 16 classes which are referred to as A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4, C_1, C_2, C_3, C_4, D_1, D_2, D_3, and D_4. Obviously, the classes A_1, A_2, A_3, and A_4 are more related to one another than they are to any of the other classes. The same is true for each of the subsets $\{B_1, B_2, B_3, B_4\}$, $\{C_1, C_2, C_3, C_4\}$, and $\{D_1, D_2, D_3, D_4\}$. Thus the set of classes could meaningfully be represented as a set of sets like $\{\{A_1, A_2, A_3, A_4\}, \{B_1, B_2, B_3, B_4\}, \{C_1, C_2, C_3, C_4\}, \{D_1, D_2, D_3, D_4\}\}$. We could refer to these four subsets as superclasses.

Definition 6.5 (Superclass)
*A **superclass** represents a set of **classes** and can be treated as **class** itself. Thus a **superclass** could also represent a set of **superclasses**, that is a higher order set of classes.*

*Two **superclasses** are said to be **disjoint**, if the corresponding higher order sets of classes are disjoint.*

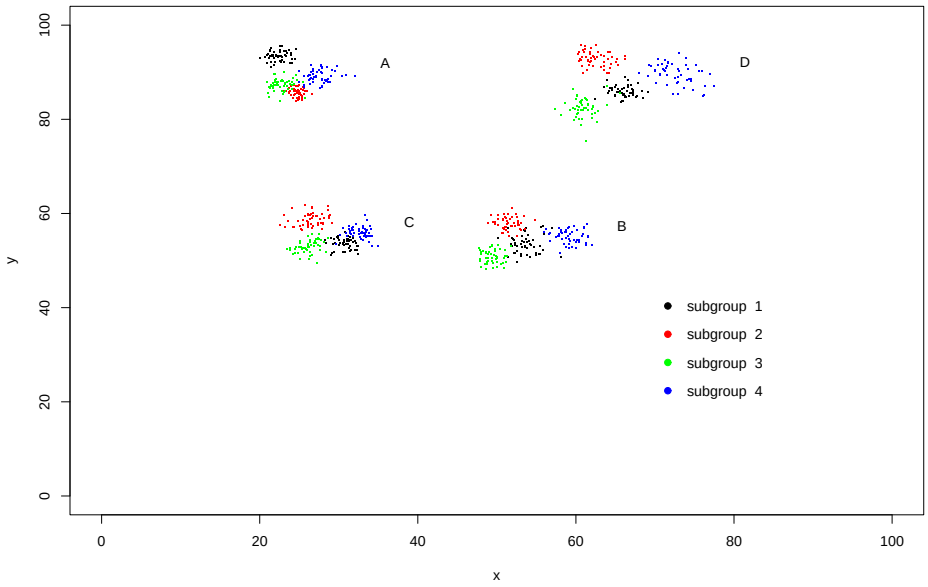


Figure 6.1: Hierarchically structured classes.
These data are composed of 16 classes, that are organized in four superclasses, A, B, C, and D, respectively, each containing 4 base classes.

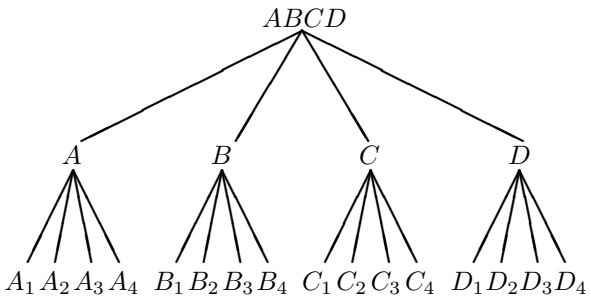


Figure 6.2: A hierarchy of classes as tree.
The hierarchy of the 16 classes of the data presented in Figure 6.1. The root of the tree represents all 16 classes, the root's four children represent the four superclasses, A, B, C, and D, respectively, and the children of those represent the base classes. This representation of the hierarchy as tree is equivalent to the description as set of sets: $\{\{A_1, A_2, A_3, A_4\}, \{B_1, B_2, B_3, B_4\}, \{C_1, C_2, C_3, C_4\}, \{D_1, D_2, D_3, D_4\}\}$, according to Lemma 6.1.

Lemma 6.2
A hierarchy of classes is a set of disjoint superclasses.

Proof.
The claim of Lemma 6.2 follows directly from Definitions 6.1, 6.3, and 6.5.

□

Given this notion of superclass, we can describe the data presented in Figure 6.1 as consisting of 4 superclasses, say A, B, C, and D, each representing four base classes, A_1 to A_4, B_1 to B_4, C_1 to C_4, and D_1 to D_4, respectively. The respective hierarchy of classes is given in Figure 6.2.

Note that in classification tasks performed in the hitherto given context of hierarchies of classes an encountered class could be either a base class or a superclass, that is a set of classes its members being in turn base classes or superclasses.

6.2 Hierarchy-Preserving Binarization of Trees

To employ nested dichotomies as binarization of a polytomous classification problem involving hierarchically structured classes we are now to find a binary tree which preserves the hierarchy. Despite Lemma 6.1, the relation from hierarchies of classes to binary trees is neither injective nor surjective unless the binary tree is assumed to be the exact representation of the hierarchy of classes. If, however, the binary tree were the exact representation of a hierarchy of classes according to Lemma 6.1, each level of the hierarchy would unite two sub-hierarchies. If, otherwise, a hierarchy consists not already of nested dichotomies, it could possibly be represented by several binary trees and any binary tree could possibly be read as representation of several very different such hierarchies.

A system of nested dichotomies as it was introduced so far was usually to represent a flat hierarchy. But it could also be seen as a representation of an ordered system of classes, where the binary tree representing the system of nested dichotomies also represents the order of the classes, that is the hierarchy of classes (recall the '*ordered class binarization*' as defined by Fürnkranz [69, Def. 3] and described in chapter 2, section 2.2.3). But few constraints are sufficient to define the binary trees which are valid representations of a hierarchy. Also binary trees representing a hierarchy are easily to construct given the tree defining the respective hierarchy.

6.2.1 Properties of Valid Representations of Hierarchies of Classes as Binary Trees

If a binary tree is to represent an arbitrary hierarchy of classes $\mathcal{H}$ defined for a set of classes $\mathcal{C}$ that is not binary itself, it will be the tree representation of a more specialized hierarchy of classes $\mathcal{H}'$ for the same set $\mathcal{C}$ according to Lemma 6.1. We will refer to hierarchies of classes that are specialized in this manner as binary hierarchies of classes:

Definition 6.6 (Binary Hierarchy of Classes)
*A **binary hierarchy of classes** is a hierarchy of classes that contains either exactly one class or exactly two **binary hierarchies of classes**. It corresponds to a binary tree according to Lemma 6.1.*

If a hierarchy of classes $\mathcal{H}$ is denoted as $\mathcal{H}_{bin}$, it will be a binary one.

A valid binarization of a hierarchy of classes is therefore defined as follows:

Definition 6.7 (Valid Binarization of a Hierarchy of Classes)
*A binary hierarchy of classes $\mathcal{H}_{bin}$ will be a **valid binarization of a hierarchy of classes** $\mathcal{H}$, if and only if all hierarchies of classes that are members of $\mathcal{H}$ are also members of $\mathcal{H}_{bin}$, that is if and only if it is less general than or equal to $\mathcal{H}$ ($\mathcal{H}_{bin} \succeq \mathcal{H}$) according to Definition 6.4.*

Obviously for any other hierarchy of classes $\mathcal{H}_n$ contained by $\mathcal{H}_{bin}$ the corresponding set of classes will be a subset of at least one set of classes corresponding to a hierarchy of classes $\mathcal{H}_m$ contained by $\mathcal{H}$ as well as a superset of at least one set of classes corresponding to another hierarchy of classes $\mathcal{H}_o$ contained by $\mathcal{H}$. More formally we can state this property as:

$$\begin{aligned} \forall\, \mathcal{H}_n, (\mathcal{H}_n, \mathcal{H}_{bin}) \in \mathcal{M}, (\mathcal{H}_n, \mathcal{H}) \notin \mathcal{M} : \\ \exists\, \mathcal{H}_m, (\mathcal{H}_m, \mathcal{H}) \in \mathcal{M} : S(\mathcal{H}_n) \subset S(\mathcal{H}_m) \\ \wedge \quad \exists\, \mathcal{H}_o, (\mathcal{H}_o, \mathcal{H}) \in \mathcal{M} : S(\mathcal{H}_o) \subset S(\mathcal{H}_n). \end{aligned}$$

Representing a hierarchy of classes that is a valid binarization of a given hierarchy of classes as tree would provide us directly with the representation of a system of nested dichotomies that incorporates the domain knowledge concerning the hierarchical structure of the set of classes – and vice versa, of course. So the question is, how to find those valid binarizations.

6.2.2 Algorithmic Approach to Valid Binarization of Trees

Fortunately, construction of a valid binarization of a hierarchy of classes is straightforward like binarization of a tree: To binarize a hierarchy of classes represented by a tree we take the root of the tree as root of a binary tree and the children of the root as leaves of the binary tree in any order. For any of the children of the root in the original tree having children itself, we take the respective subtree rooted at the respective node and treat it the same way, replacing the respective node in the binary tree created before with the newly created binary tree. Let us state more formally:

Lemma 6.3 (Hierarchy-Preserving Binarization)
Let $\mathcal{H}$ be a hierarchy of classes. Let

$$f_{ND} : \textit{Class list} \rightarrow \textit{Nested Dichotomies}$$

be a procedure to derive a system of nested dichotomies for a given set of classes as defined in Table 4.1. Then the following procedure for binarization of $\mathcal{H}$ will preserve the hierarchy according to Definition 6.7:

1. *Treat the members of the hierarchy of classes as (super-)classes according to Definition 6.5 and Lemma 6.2 and build a binary tree by applying f_{ND} to the set of the members.*
2. *Apply recursively step 1 to each member that contains more than one class. Replace the leaf of the previously created binary tree corresponding to the respective superclass by the newly created binary tree.*

The binarized hierarchy of classes is the hierarchy of classes corresponding to the finally resulting binary tree.

Proof.
The given construction of the binary tree for a hierarchy of classes $\mathcal{H}$ makes sure that the binary hierarchy of classes $\mathcal{H}_{bin}$ corresponding to the resulting binary tree

```
PROCEDURE f_HND(ClassList)
{
    Tree ← f_ND(ClassList);
    Leaves ← list of leaves of Tree;
    WHILE Leaves contains superclasses DO
        FOR i ← 1 TO length(Leaves) DO
            Leaf ← i^th element of Leaves;
            IF Leaf is superclass THEN
                Replace Leaf in Tree with f_ND(Leaf);
        Leaves ← list of leaves of Tree;
    RETURN Tree;
}
```

Table 6.1: Pseudocode for binarization of a hierarchy of classes.
The presented function binarizes a given hierarchy of classes according to Lemma 6.3 and therefore preserves the hierarchy.

will contain any hierarchy of classes $\mathcal{H}_m, (\mathcal{H}_m, \mathcal{H}) \in \mathcal{M}$. Thus it will be a valid binarization of $\mathcal{H}$ according to Definition 6.7.

□

6.2.3 Hierarchically Nested Dichotomies

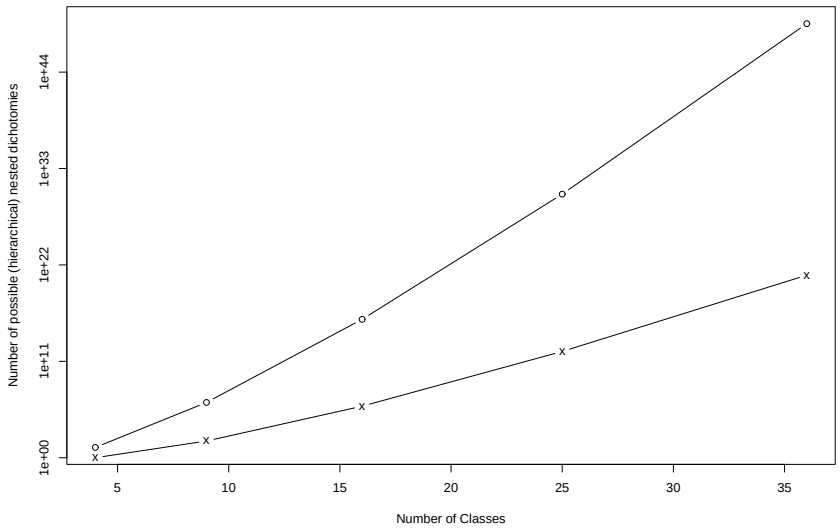
The construction of valid binarizations of hierarchies of classes given in Lemma 6.3 already provides a system of nested dichotomies. Thus the procedure f_{ND} can be employed to derive a system of nested dichotomies reflecting a given hierarchy of classes. This is illustrated in Table 6.1. We refer to a system of nested dichotomies derived from a hierarchy of classes by this procedure as *system of hierarchically nested dichotomies* or simply *hierarchically nested dichotomy* (HND). A hierarchically nested dichotomy has the same properties as a general nested dichotomy (chapter 4). Especially the estimated class probability distribution for hierarchically nested dichotomies and thus the decoding is given by Equation 4.2 as for general nested dichotomies. But furthermore it reflects a hierarchy of classes since a superclass will not be splitted until the superclass is separated from all its siblings. That is, superclasses become nested before their members will do so.

6.2.3.1 Reduction of Error Variance of NDs by Reflecting a Hierarchy

Basically one can think of the constraints for valid binarizations of hierarchies of classes as restrictions of the space of possible nested dichotomies for the respective set of classes. This is illustrated by the different rates of growth for the unrestricted space in comparison to the restricted space of possible nested dichotomies, although we take a simplified type of hierarchies of classes corresponding to completely balanced trees with an equal branching rate for each node. Figure 6.3 illustrates the growth rates for such examples. More formally the examples are restricted as follows: For a hierarchy describing n classes in l levels at each level the respective superclass is divided in c subclasses (thus the number n of classes equals c^l). The growth of valid binarizations for such hierarchies of classes would be described by the following Lemma:

Lemma 6.4 (Growth of Possible Valid Binarizations)
For a completely balanced tree providing a hierarchy of $n = c^l$ classes, that is the number of leaves, where l is the number of levels of the hierarchy, and c is the

(a)



(b)

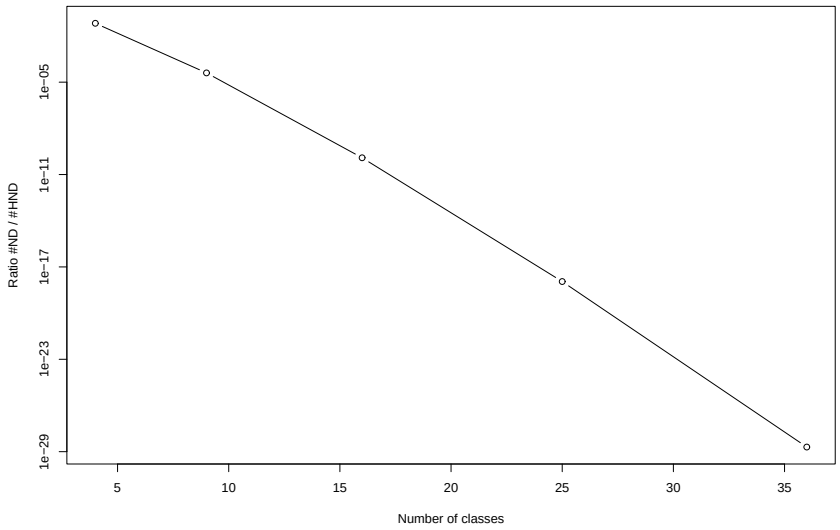


Figure 6.3: Comparison of the growth of spaces of possible nested dichotomies, and possible hierarchically nested dichotomies.
Number of possible nested dichotomies (restricted and unrestricted, respectively) against the number of classes (c^2).

(a) Comparison of $T(c^2)$ (o), and $H(c^2)$ (x) (that is for 2 levels).

(b) Ratio of $\frac{H(c^2)}{T(c^2)}$ (that is for 2 levels).

Both spaces grow more than exponential, but even the ratio decreases at a more than exponential rate, that is $T(n)$ growths considerably faster.

number of children for each internal node, there are

$$\begin{aligned} H(n) &= H(c^l) \\ &= T(c)^{\frac{1-c^l}{1-c}} \end{aligned}$$

***valid binarizations**, where $T(n)$ is the number of possible nested dichotomies for n classes, as given in Lemma 4.1.*

Proof. (Induction on the number of levels)

base case: For $l = 1$, there is no hierarchy and we gain the unrestricted number of possible nested dichotomies for $c^1 = n$ classes, which is $T(c)$.

induction hypothesis: For l levels there are $H(c^l) = T(c)^{\frac{1-c^l}{1-c}}$ valid binarizations.

induction case ($l \rightarrow l+1$): Any valid binarization for level l provides c^l leaves. Incorporating the level $l+1$ adds $T(c)$ possible nested dichotomies consisting of the respective c classes to each of the c^l leaves for each of the $H(c^l)$ valid binarizations of level l, that is, for each of $H(c^l)$ valid binarizations there are c^l combinations of $T(c)$ possibilities. Thus we gain:

$$\begin{aligned} H(c^{l+1}) &= H(c^l) \times T(c)^{c^l} \\ &= T(c)^{\frac{1-c^l}{1-c}} \times T(c)^{c^l} && \text{(induction hypothesis)} \\ &= T(c)^{\sum_{i=0}^{l-1} c^i} \times T(c)^{c^l} && \text{(geometric series)} \\ &= T(c)^{\sum_{i=0}^{l} c^i} \\ &= T(c)^{\frac{1-c^{l+1}}{1-c}} && \text{(geometric series).} \end{aligned}$$

□

The assumption that the hierarchy is completely balanced and that each internal node of the corresponding tree has the same number of children is by no means necessary for our method. But it does suffice to illustrate the restriction of the space of possible nested dichotomies under the given constraints in comparison to the unrestricted space. As Figure 6.3 shows both spaces are growing more than exponentially, but the restricted space of hierarchically nested dichotomies at a distinctly slower rate. Thus the ratio of the size of the restricted space to the unrestricted space decreases also at a more than exponential rate. Note that the restriction will be even stronger for hierarchies of classes corresponding to an unbalanced tree usually.

The reasoning above allows to conclude that considering a proper hierarchy will restrict the hypothesis space of nested dichotomies to a subset of all nested dichotomies exhibiting a decreased variance of error since the space of possible hierarchically nested dichotomies is considerably restricted compared to the space of possible arbitrarily nested dichotomies. On the other hand the error variance will still be high since the space of possible hierarchically nested dichotomies still grows at an over exponential rate.

6.2.3.2 Impact of a Hierarchy on the Classification Task

Since for a class c some sets of classes occurring in the nested dichotomy from the path to the leaf corresponding to c will correspond to a superclass of c the decision of the ND for c is conditioned by the former decisions for all superclasses of c. This makes sense and corresponds to the intuition which leads to the incorporation of domain knowledge concerning hierarchies of classes at first sight. But we ought to

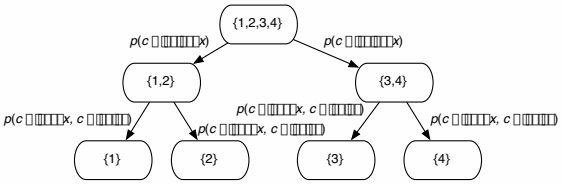


Figure 6.4: High-level classifications can be overruled by low-level classifications. Simple example of an HND. superclasses are $\{1, 2\}$ and $\{3, 4\}$, respectively. A low-confidence assignment to a superclass can be overruled by a high-confidence assignment to a class at lower levels.

note that the separation of some superclasses might be difficult. A misclassification at a high level is unlikely, though possibly, to be compensated for at a lower level.

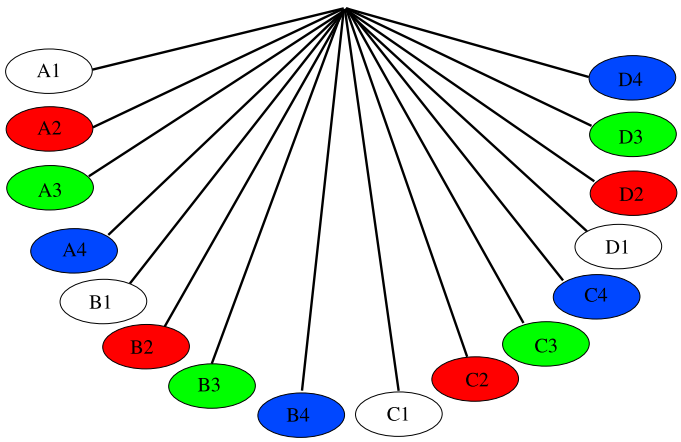
On the other hand a classification at a high level that is correct, but with low confidence, could be overruled by assignments at lower levels that are incorrect, but with high confidence. This could be possible, since all but the correct classifiers at a lower level are not trained with respect to the proper class. Consider following example: For the simple HND given in Figure 6.4, let the probabilities provided by the base learners for an instance x be given by:

$$\begin{aligned} p(c \in \{1,2\}|x) &= 0.4 \\ p(c \in \{3,4\}|x) &= 0.6 \\ p(c = 1|x, c \in \{1,2\}) &= 0.95 \\ p(c = 2|x, c \in \{1,2\}) &= 0.05 \\ p(c = 3|x, c \in \{3,4\}) &= 0.4 \\ p(c = 4|x, c \in \{3,4\}) &= 0.6 \end{aligned}$$

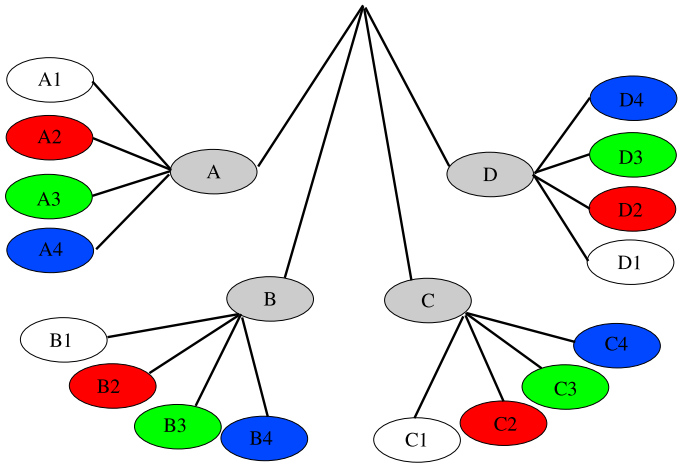
Then the decoded probabilities would be given by:

$$\begin{aligned} p(c = 1|x) &= p(c \in \{1,2\}|x) \times p(c = 1|x, c \in \{1,2\}) \\ &= 0.38 \\ p(c = 2|x) &= p(c \in \{1,2\}|x) \times p(c = 2|x, c \in \{1,2\}) \\ &= 0.02 \\ p(c = 3|x) &= p(c \in \{3,4\}|x) \times p(c = 3|x, c \in \{3,4\}) \\ &= 0.24 \\ p(c = 4|x) &= p(c \in \{3,4\}|x) \times p(c = 4|x, c \in \{3,4\}) \\ &= 0.36 \end{aligned}$$

The class of x would therefore be decoded as 1, even though 4 might be the correct assignment. The example is given in [136] to characterize a drawback of approaches that do not prune subtrees of hierarchically structured classification approaches following only the single path with the respective top-confidence. While this setting could eventually occur as a problem for a single ND or any other hierarchically structured classification approach, it is rather unlikely and we do not follow the argument given in [136]. The setting is even more unlikely for HNDs, given a properly defined hierarchy, since a well defined hierarchy is supposed to allow a clear separation of superclasses. However, note that ensembles of NDs are to reduce that kind of error variance. Therefore, we will consider ensembles of HNDs as well in the next section. A more convincing argument given by Schubert et al. [136] is to consider the pruning of subtrees that are not at top-confidence as a matter of efficiency – but this is not primarily in the scope of our investigation since we are more interested in effectiveness.



(a)



(b)

Figure 6.5: Incorporating domain knowledge can reduce complexity of a classification task.
The classification problem concerning the polytomy given in Figure 6.1.

(a) Not taking into account domain knowledge.

(b) Taking into account domain knowledge concerning the pronounced hierarchy of the classes.

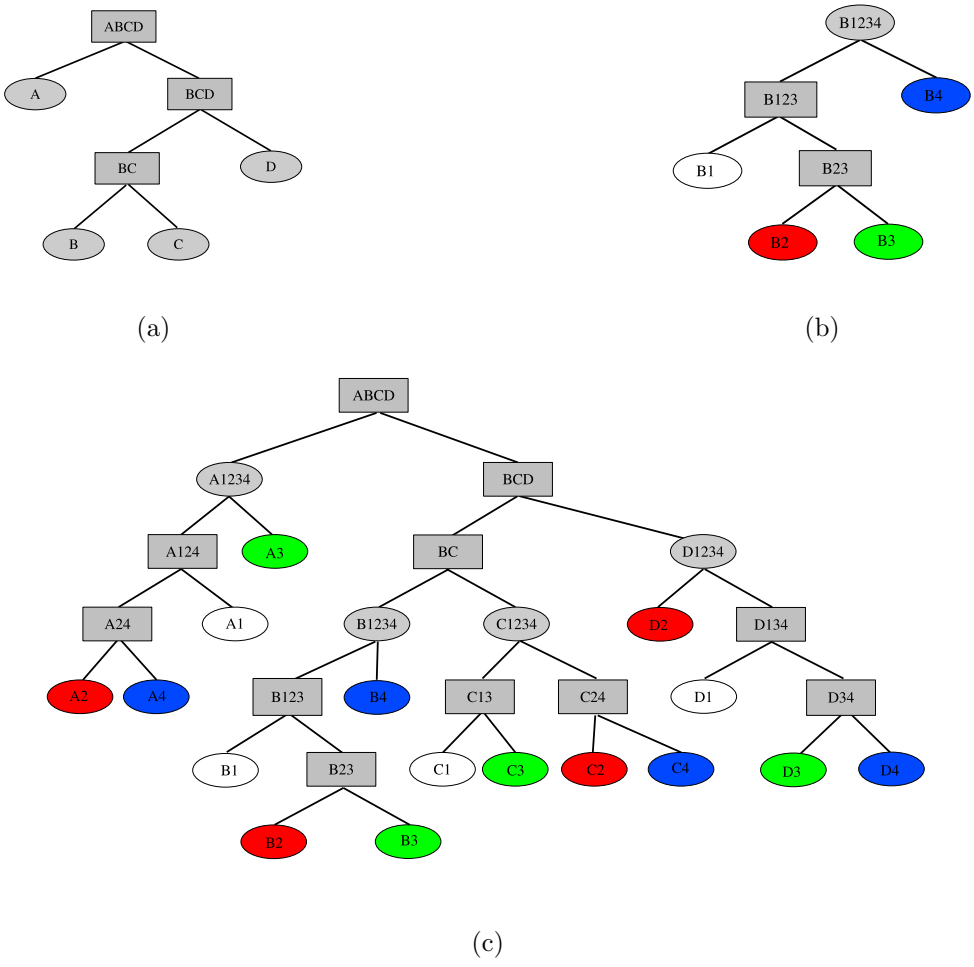


Figure 6.6: Example of the recursive binarization procedure.
The figure shows three partial steps of the binarization procedure for the classification task given in Figure 6.5b, corresponding to the hierarchy of classes given in Figure 6.2:

(a) a possible binarization of the four superclasses (one out of $T(4) = 15$),

(b) a possible binarization of the subclasses of B (one out of $T(4) = 15$), and

(c) the fully binarized problem (one possible binarization out of $T(4)^5 = 759,375$ – unrestricted there would have been $T(16) = 6.19 \times 10^{15}$ possible systems of nested dichotomies).

However, if the hierarchy of classes is merely pronounced, the classification task could generally be expected to become easier by taking the hierarchy into account. Reconsider the example of Figure 6.1. Any approach to the polytomous classification problem not taking into account the pronounced hierarchy would have to discriminate 16 classes directly against each other. Taking into account knowledge concerning the pronounced hierarchy of classes reduces the complexity of the classification task considerably: Five more simple classifiers are each to learn the discrimination of four classes only as illustrated in Figure 6.5.

This is exactly how hierarchically nested dichotomies work. Basically a system of nested dichotomies is built for the classification problem concerning the superclasses and another system of nested dichotomies for each classification problem concerning subclasses of the respective superclass and so on. The level-wise procedure is illustrated in Figure 6.6.

Although the procedure results in an overall system of nested dichotomies, treating the systems of nested dichotomies for different levels of the hierarchy as separate systems (as implied by the illustration in Figure 6.6) would make no difference in theory. However, practically this would provide another benefit since superclasses could separate at different levels by different sets of features as hierarchies of classes could be defined for different levels with respect to different similarity measures (this is already implied in Definition 6.3). Considering the systems of nested dichotomies on different levels as separate classifiers one could also use different sets of features for different levels, or use different sets of parameters to train them. Even using different base classifiers for different hierarchical levels could make sense.

6.3 Ensembles of Hierarchically Nested Dichotomies

Building *ensembles* of several systems of *hierarchically nested dichotomies* (EHNDs) is suggesting for the very same reasons as it is to compose ensembles of several systems of nested dichotomies (chapter 4). Especially reduction of error variance is still an issue.

6.3.1 Reducing Variance of Error

Although a properly defined hierarchy will already reduce the error variance in the set of possible hierarchically nested dichotomies, this set will be still large. Thus the single system of hierarchically nested dichotomies can be expected to be more accurate than an arbitrary system of nested dichotomies, but still biased and diverse from other HNDs. To reduce the remaining error variance is therefore the main motivation for building an ensemble.

A system of hierarchically nested dichotomies is biased due to its correspondence to a binary hierarchy of classes (Definition 6.6) which will generally not be identical to the hierarchy of classes the system of hierarchically nested dichotomies was built for. Thus the hierarchically nested dichotomy will imply a strong bias towards a hierarchy of classes which is basically insufficiently general and therefore essentially wrong. Another HND built for the very same hierarchy of classes will exhibit a bias towards another binary hierarchy of classes. Several HNDs will therefore be diverse, so building an ensemble of them will make sense, and an ensemble of HNDs will usually improve over a single system of HNDs.

6.3.2 Hierarchies as Proper Restrictions of Space of Hypotheses

On the other hand the diversity of HNDs is restricted by the hierarchy of classes in comparison to general nested dichotomies. Thus one could expect ensembles of HNDs to improve upon ensembles of NDs, if and only if the respective hierarchy of classes is a proper description of the relations between the classes in the feature space since the single HNDs can then be expected to be more accurate than an arbitrary ND. Otherwise the restriction will merely cause the errors of the single HNDs to be stronger correlated than the errors of general NDs would be since the diversity of NDs is less restricted than the diversity of HNDs. Thus the ensemble of HNDs could be supposed to predict with worse accuracy compared to an END in that case. The comparison of performance of EHNDs and ENDs therefore could basically provide a measure to assess the ability of the chosen feature space to represent the stated hierarchy of classes given the hierarchy is appropriate to the classification problem at all. This would be especially not the case, if e.g. the respective superclasses were more difficult to be separated from each other than arbitrary mixtures of classes.

6.3.3 Specification of Hierarchies by Sets of Binary Hierarchies

Another property of sets of HNDs seems noteworthy: While one HND is possibly related to many hierarchies of classes, note that a set of HNDs could possibly imply a smaller range of hierarchies of classes or even define a single one as the HNDs differ sufficiently. According to Definition 6.7 one could state a hierarchy of classes to be implied by a set of binary hierarchies of classes, if and only if it is more general than or equal to every of the binary hierarchies of classes (where in case of equality the set would contain only one binary hierarchy of classes) and there is no other hierarchy of classes that fulfills this condition. More formally:

$$\begin{aligned}\{\mathcal{H}_{bin_1}, \ldots, \mathcal{H}_{bin_n}\} \equiv \mathcal{H} :\Longleftrightarrow \\ \forall\, i \in \{1, \ldots, n\} : \mathcal{H}_{bin_i} \succeq \mathcal{H} \\ \wedge \quad \forall \mathcal{K} : \mathcal{H}_{bin_i} \succeq \mathcal{K} \forall\, i \in \{1, \ldots, n\} \Rightarrow \mathcal{K} = \mathcal{H}.\end{aligned}$$

Obviously, any hierarchy of classes could therefore be completely defined by at most two binary hierarchies of classes, both being degenerated to a list by picking different members first out of any commonly occurring sub-hierarchy of classes. However, for a hierarchy of classes being sufficiently general there will be far more than two binary hierarchies of classes representing the original hierarchy of classes and nevertheless differing from each other in a degree that justifies building an ensemble of them.

This reasoning justifies building ensembles of HNDs again and from another point of view: An EHND built of a sufficient number of HNDs will define the hierarchy used for building a single HND more exactly than any single HND (given the hierarchy is not binary itself).

6.3.4 Summary

General reasons why to build ensembles of classifiers and requirements to build good ensembles of classifiers where given in chapter 3. In chapter 4 it was shown that nested dichotomies are a paragon of good ensemble members. This holds still true for hierarchically nested dichotomies: Although space of possible hierarchically nested dichotomies is restricted in comparison to the space of possible general nested dichotomies it is still manifold.

Moreover, while a single HND will usually describe the given hierarchy only poorly, a set of several HNDs can possibly define the given hierarchy completely.

It is, however, essential to define a proper hierarchy of classes. If the hierarchy describes the relation between classes well, the space of possible HNDs will contain a more accurate subset of general NDs. Unless the hierarchy is too restrictive the respective subset of NDs will still contain diverse HNDs. But on the other hand an improper hierarchy will merely correlate the errors of according HNDs instead of improving the accuracy.

6.4 Related Work

Since nested dichotomies are a standard statistical technique for tackling with multi-class problems there unsurprisingly have emerged some ideas in the scientific community similar to HNDs in selecting proper nested dichotomies with respect to the structure of data. However, we are neither aware of approaches using a priori information concerning the structure of classes nor have we found other approaches building ensembles of nested dichotomies – which is recommendable even in case a selection of proper nested dichotomies has been done (unless one single system of nested dichotomies is to represent the structure of classes perfectly).

A simple adaption of nested dichotomies to apply support vector machines to multi-class problems was recently proposed by Vural and Dy [148]. They use a single system of nested dichotomies and compare its performance against *one versus others* and *all versus all*, for the testing of the latter using both the *max wins* strategy [68] and the *decision directed acyclic graph* (DAGSVM) [124]. The system of nested dichotomies is apparently not understood by Vural and Dy as a probabilistic framework, but as a binary decision tree. So no class probability estimates are derived nor used for weighting the decisions in deeper levels of the tree.

However, the point in their work we want to consider is how they propose to derive the successive partitions of the set of classes into nested dichotomies. Three methods are suggested:

1. *k-means based division*: Each class is represented with its corresponding mean (μ_i). The means are grouped into two, using the *k-means* algorithm.

2. *spherical shells*: The classes are grouped into two with respect to the total mean M of data points: The classes with $\mu_i < M$ are grouped as the negative class, and the others as the positive class.

3. *balanced subsets*: The data is divided into two subsets with minimum difference in the number of the instances in the two subsets.

The third method may be useful if the speed of the process of dividing the data into subsets is of importance. Is seems to be of absolutely no use for deriving a hierarchy of classes that makes sense. The second method will not separate more related classes from each other before less related ones are separated – but only if the relations are organized spherically with respect to the origin, which seems to be quite an unnatural assumption. The idea to use a clustering algorithm after all is promising. However, *k-means* [104] (the authors actually cite a precursor of k-means, [64], which is less efficient, but essentially the same) shows certain drawbacks which are obviously not widely acknowledged. Despite the fact it is used in many applications, it is not at all convenient in all those applications.

- First of all the parameter k, that is the number of clusters, is to be set in advance. In our case it will always be 2. If the real number of clusters does not equal k, the results obtained from *k-means* will be rather poor in many

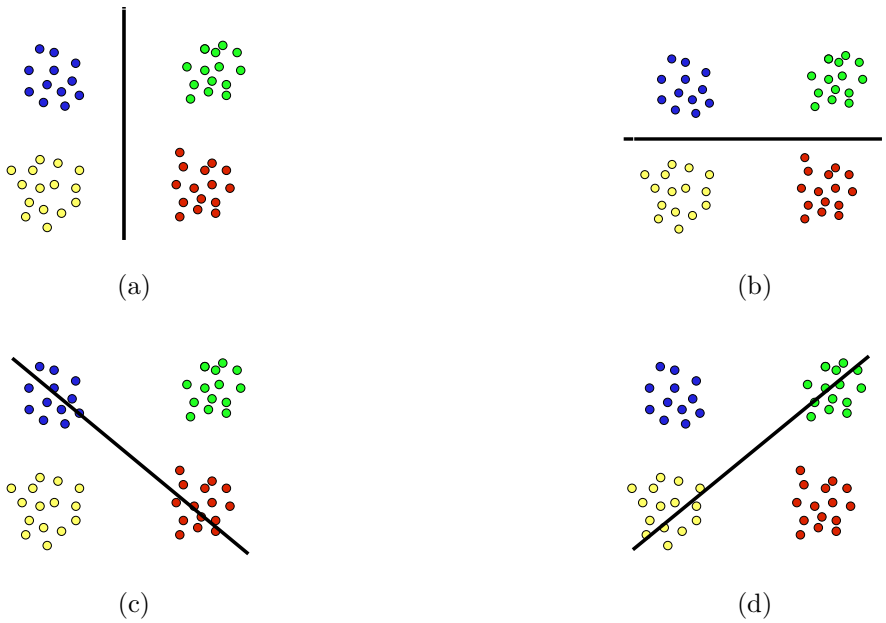


Figure 6.7: *k-means* can find four stable separations of four clusters into two.
For a k smaller than the real number of clusters, the result of *k-means* is rather indeterminate. In theory, there are four stable results for finding two clusters for given four clusters, given the clusters are more or less equally distant from each other (except the diagonals) which is a reasonable assumption for a classification problem. (a) and (b) each resemble a global optimum, (c) and (d) a relatively stable local optimum, respectively, for the clustering.

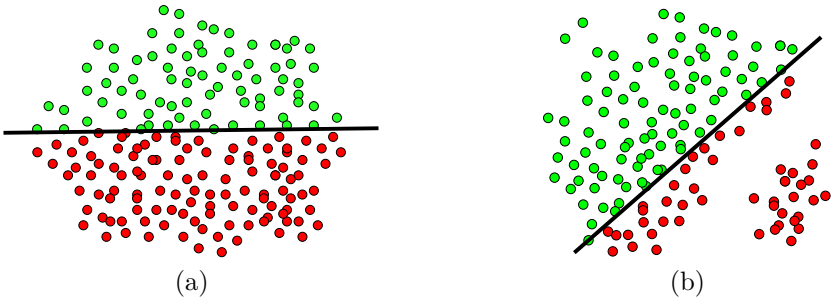


Figure 6.8: *k-means* finds always k clusters preferring equal extensions of clusters.

(a) *k-means* will always find k clusters even if there are none. In this case one big cluster is divided into two. The separation will make absolutely no sense for a classification task.

(b) For $k = 2$ and two clusters of very different size being close to each other, the bigger one will be split asunder by *k-means* and one part will be merged with the smaller cluster in order to find a stable separation. This split will be very unfavorable for any classification task.

cases, dividing several clusters, or at least the result will be indeterminate. (Of course the result of *k-means* is always indeterminate since it is a technique of local optimization. But in the discussed case the global optimum can be also indeterminate. See Figure 6.7 for an illustration.)

- Furthermore *k-means* will always find k clusters, even if there are non at all or the data are clustered in one big cluster without reasonable discrimination lines. This is illustrated in Figure 6.8 (a).
- Finally, if there actually are two clusters not too far from each other but one being much bigger than the other, *k-means* will usually divide the big one and merge one part with the small cluster in order to find a stable separation. See Figure 6.8 (b) for an illustration of this problem.

The reasons demonstrated so far suffice to show that using *k-means* for deriving a hierarchy of classes in an arbitrary classification task seems generally not to be a good idea. However, even not being aware of these drawbacks, one will be bound to concede that a clear separation in all dimensions of the feature space is not at all required for an arbitrary classifier. Usually a separation in only one dimension will suffice. Unfortunately such separations will not be detectable for *k-means* in general.

Therefore the method proposed already four years earlier by Kumar et al. [96] seems to be more elaborate although it is a very specialized proposal for a certain dataset. However, the proposed method is not specialized with respect to this dataset. It is based on covariance within and between classes and uses simulated annealing to derive a low entropy of the class partitioning. There may possibly be less complicated methods since at first sight a sort of subspace clustering might suffice to derive appropriate class hierarchies. Several profound algorithms for subspace clustering or feature selection were proposed in the recent years (see e.g. [1], [2], [3], [34], [73], [89], [94], and [17]). Combining advanced clustering methods like those with classifiers like our proposed EHNDs could be a remunerative topic for further research.

Although the method of partitioning proposed by Kumar et al. is quite sophisticated, both of the cited methods choose a single system of nested dichotomies after all. Even if the selection of a single system of nested dichotomies is based on reasonable decisions, this will put a bias on the resulting classifier which often will be stronger than it needs to be – resulting in a high variance component of the classification error. This can be reduced building an ensemble of diverse systems of nested dichotomies.

Note that both methods derive a hierarchy *ad hoc* based on the respective training data. So neither of both is addressing the issue of incorporating domain knowledge.

6.5 Conclusion

We proposed a method to binarize polytomous classification problems based on *systems of nested dichotomies* while incorporating domain knowledge concerning the hierarchical structure of classes. Therefore we introduced constraints to define a subset of binary trees that are valid binarizations of arbitrary hierarchies and developed a method to derive nested dichotomies according to those constraints given a previously defined hierarchy of classes. Nested dichotomies that are built according to this method are called *hierarchically nested dichotomies*.

For nested dichotomies a recently proposed ensemble method, ENDs [66], was reviewed in chapter 4. Since nested dichotomies impose a distinct order on classes

a single system of nested dichotomies exhibits a severe bias. The space of possible nested dichotomies is very large and several systems of nested dichotomies thus will be very diverse. Therefore, by building ensembles the error variance is effectively reduced.

Obviously, the set of hierarchically nested dichotomies is a subset of nested dichotomies. Thus the space of possible hierarchically nested dichotomies is considerably restricted with respect to the space of possible nested dichotomies. Nevertheless it grows by an over-exponential rate for an increasing number of classes. Since the members of an arbitrary set of hierarchically nested dichotomies can therefore still be considered to be diverse, building ensembles of hierarchically nested dichotomies can reasonably be expected to perform well. A set of HNDs will usually describe the given hierarchy more exactly than any single member of the set. Thus the averaged bias of the ensemble members will approximate the bias that was desired by defining the hierarchy.

Constraining nested dichotomies by a hierarchy can be considered to affect directly the accuracy of the nested dichotomy as a single classifier system. If the defined hierarchy describes the real hierarchy well and as it is reflected in feature space, the bias of a hierarchically nested dichotomy will be less erroneous than the bias of an arbitrarily nested dichotomy. Otherwise, however, the errors of several hierarchically nested dichotomies defined according to the identical erroneous hierarchy will be stronger correlated than the errors of several arbitrarily nested dichotomies. This consideration motivates us to expect the trade-off in performance between ENDs and EHNDs *ceteris paribus* to be of value for evaluating the quality of feature spaces. Note, however, that a well defined hierarchy that is completely detectable in the feature space might be of no use, if EHNDs are employing a rather weak learner as base learner which is not able to take advantage of the good representation.

It might occur that classes and superclasses at different levels of a hierarchical classification problem are best separated by different features, different methods, or the same method but in different parameterization. Let us therefore note that our method provides the ability to make use of different features, different methods, and different parameterization for each level of the given hierarchy.

Chapter 7

Evaluation

For evaluation of the proposed method of ensembles of hierarchically nested dichotomies, EHNDs were implemented in `Java` and incorporated in `Weka` [151]. This allows a direct comparison to ENDs which are also available as a classifier within the `Weka`-framework. Furthermore there is a broad range of classification methods available in the same framework. Thus any of them can easily be used as base-learners and competitors.

7.1 Prediction Performance Measures

The performance of any prediction algorithm is completely given in the confusion matrix. The confusion matrix for n classes is a matrix $Z \in \mathbb{N}^{n \times n}$, $Z = (z_{ij})_{i \in N, j \in N}$, where z_{ij} represents the number of times the input is predicted to be in class j while belonging to class i and $N = \{1, \ldots, n\}$ is the set of indices of classes. For two classes, the confusion matrix is easily readable. It counts the occurrences of correct and false predictions for both classes, resulting in four numbers: TP (true positive) is the number of decisions for class 1 for an instance truly belonging to class 1. TN (true negative) is the number of decisions for class -1 for an instance truly belonging to class -1 (i.e. not belonging to class 1). FN (false negative) is the number of decisions for class -1 for an instance that would have been correctly classified as class 1. FP (false positive) is the number of decisions for class 1 for an instance that would have been correctly classified as class -1. The sum $TP + TN + FP + FN$ is the number of instances.

Dealing with polytomous classification problems the situation is more complex since the four numbers defined above will then be defined for each single class, treating all other classes as the negative class. The performance can still be read out of the confusion matrix, but it will be less convenient. Thus one uses other performance measures for multi-class problems.

However, a reasonable measurement of performance should not omit any of the four numbers, TP, TN, FP, or FN. Otherwise information will be lost. On the other hand one wishes to make the performance measure conveniently readable and comparable. Thus one can use percentages and average the values for several classes (e.g. the superclasses of some hierarchically structured problem), or for all of them. Often ratios of the numbers are used. For a profound overview we refer to [13, chapter 6.6 – 6.7], [14], and [147].

For evaluation of EHNDs we choose the prediction performance measures:

- true positive rate,
- false positive rate,

- positive predicted value,
- and F_1-measure.

The remainder of the current section will shortly describe these performance measures.

7.1.1 True Positive Rate

The true positive rate is defined as:

$$TPR = \frac{TP}{TP + FN}. \tag{7.1}$$

The true positive rate is a measure for positive coverage and is also known as *sensitivity* or *recall*. For a single class it equals the number of correctly classified members of this class divided by the overall number of members of this class:

$$TPR_i = \frac{TP_i}{TP_i + FN_i}, \tag{7.2}$$

where

$$TP_i = z_{ii}$$

is the number TP for class i and

$$FN_i = \sum_{\substack{j \in N \\ j \neq i}} z_{ij}$$

is the number FN for class i.

As an average over all classes the true positive rate is generally known as accuracy:

$$Q = \frac{\sum_{i=1}^{n} TPR_i}{n}, \tag{7.3}$$

where n is the number of classes.

We will also use averages over certain subsets $I \subset N$ of classes (e.g. superclasses). These are defined as follows:

$$TPR_I = \frac{\sum_{i \in I} \left(TPR_i \cdot \sum_{j \in N} z_{ij} \right)}{\sum_{(i,j) \in I \times N} z_{ij}}. \tag{7.4}$$

7.1.2 False Positive Rate

The false positive rate is defined as:

$$FPR = \frac{FP}{FP + TN}. \tag{7.5}$$

The false positive rate tells the percentage of instances not belonging to a class but falsely classified as that class. It is also known as *false alarm rate*. With respect to a single class i it equals the number of instances of other classes classified as this class divided by the overall number of instances belonging not to this class:

$$FPR_i = \frac{FP_i}{FP_i + TN_i}, \tag{7.6}$$

where

$$FP_i = \sum_{\substack{j \in N \\ j \neq i}} z_{ji}$$

is the number FP for class i and

$$TN_i = \sum_{\substack{(k,j) \in N \times N \\ k \neq i \wedge j \neq i}} z_{kj}$$

is the number TN for class i.

An average of FPR_I over a certain subset $I \subset N$ of classes is defined by:

$$FPR_I = \frac{\sum_{i \in I} \left(FPR_i \cdot \sum_{j \in N} z_{ij} \right)}{\sum_{(i,j) \in I \times N} z_{ij}}. \tag{7.7}$$

Thereby the overall false positive rate is given by:

$$FPR_N = \frac{\sum_{i \in N} \left(FPR_i \cdot \sum_{j \in N} z_{ij} \right)}{\sum_{(i,j) \in N \times N} z_{ij}}. \tag{7.8}$$

7.1.3 Positive Predicted Value

The positive predicted value is defined as:

$$PPV = \frac{TP}{TP + FP}. \tag{7.9}$$

The positive predicted value is a measure for the rate of instances correctly classified as a certain class out of the overall number of instances classified as that class. It is also known as *precision* or *specificity*. For a single class i it equals the number of instances correctly classified to belong to class i divided by the overall number of instances classified to belong to class i:

$$PPV_i = \frac{TP_i}{TP_i + FP_i}, \tag{7.10}$$

where TP_i and FP_i are defined as above.

The average of PPV_I over a subset $I \subset N$ of classes is defined by:

$$PPV_I = \frac{\sum_{i \in I} \left(PPV_i \cdot \sum_{j \in N} z_{ij} \right)}{\sum_{(i,j) \in I \times N} z_{ij}}. \tag{7.11}$$

Thereby the overall positive predicted value is given by:

$$PPV_N = \frac{\sum_{i \in N} \left(PPV_i \cdot \sum_{j \in N} z_{ij} \right)}{\sum_{(i,j) \in N \times N} z_{ij}}. \tag{7.12}$$

7.1.4 F_1 Measure

The F measure [147] is a combined measure that assesses the tradeoff between precision and recall as a weighted harmonic mean:

$$F_\beta = \frac{(\beta^2 + 1) PPV \cdot TPR}{\beta^2 PPV + TPR}. \tag{7.13}$$

Usually one uses $\beta = 1$ resulting in the balanced F_1 measure:

$$F_1 = \frac{2 \cdot PPV \cdot TPR}{PPV + TPR}. \tag{7.14}$$

The F_1 measure for single classes and subsets of classes is defined straightforward by using the respective values of PPV and TPR. The F_1 measure will often be used, if one needs to optimize a single measure, that balances PPV and TPR. However, we will provide the F_1 measure in addition to these two.

7.1.5 Summary

TPR and FPR together cover all values of TP, TN, FP, and FN. These measures are used to derive the so called receiver operating characteristics (ROC) curves which reflect the tradeoff between true positive rate and false positive rate for different thresholds. Furthermore the specificity or precision of a model is described by the PPV while the F_1 measure yields the tradeoff between PPV and TPR as a single measure which could be optimized.

7.2 Data

A first evaluation of our method was done using the synthetic data given in Figure 7.1. These data were already presented in chapter 6 to illustrate the notion of a hierarchy of classes (see section 6.1). These data consist of 16 different Gaussian distributions clearly separated into four super-clusters, where the means of four Gaussian distributions are nearby each other but far from the means of the respective other 12 Gaussian distributions. Each distribution consists of 50 points in two-dimensional Euclidean space. The subgroups within each group are not clearly separable. Thus here is given a paragon of hierarchically structured classes and one can expect ensembles of hierarchically nested dichotomies to perform very well given the proper hierarchy.

7.3 Results

Not surprisingly, EHNDs performed very well on the 16 class problem with a pronounced hierarchical structure. We compared EHNDs to ENDs and SVMs. SVMs were also the base learners for ENDs and EHNDs. We used the sequential minimal optimization algorithm (SMO) by Platt [122], [123] with improvements by Keerthi et al. [90], as it is implemented in `Weka`. SVMs are fitted to the multi-class problem by pairwise coupling according to [81] (see chapter 2, section 2.2.2.3). The SMO in `Weka` provides class probability estimations on demand by fitting logistic regression models [97] to the outputs of the support vector machine. This was also used to obtain the conditioned class probability estimates in NDs and HNDs. For ENDs and EHNDs we used 20 NDs and HNDs to build ensembles. ENDs performed better than SMO. EHNDs outperformed both of them. The accuracy, that is the percentage of correctly classified instances (Q), in a 10-fold stratified cross-validation was for SMO: 78.25%, for ENDs: 82.375%, and for EHNDs: 94.75%. A detailed comparison with respect to the superclasses is given in Table 7.1.

To investigate the impact of using a hierarchy that is not reflected by the features describing the data we trained another EHND using the hierarchy as defined by: $\{\{A1, B1, C1, D1\}, \{A2, B2, C2, D2\}, \{A3, B3, C3, D3\}, \{A4, B4, C4, D4\}\}$. This imposes a strong bias on the classifier to prefer inconvenient decisions. All classes are to be discerned from each other before the actual superclasses are

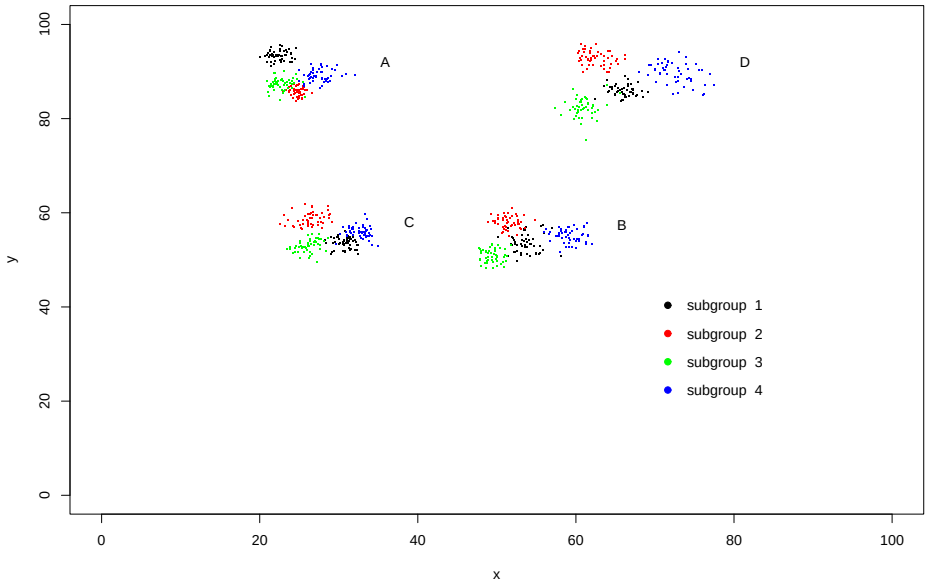
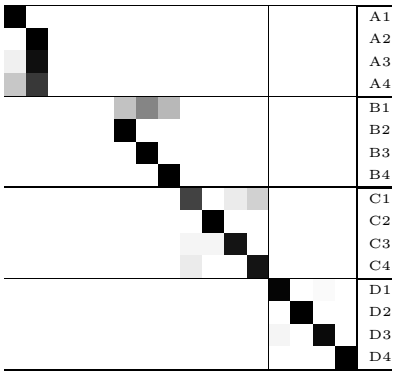


Figure 7.1: Hierarchically structured classes.
These data are composed of 16 classes, that are organized in four superclasses, A, B, C, and D, respectively, each containing 4 base classes.

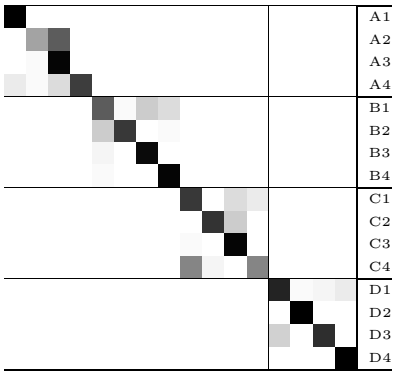
Group	Method	Training				Cross-validation			
		TPR	FPR	PPV	F1	TPR	FPR	PPV	F1
A	SVM	0.835	0.011	0.8744	0.8542	0.5	0.0333	undef	undef
	END	0.78	0.0147	0.8559	0.8162	0.775	0.015	0.8457	0.8088
	EHND	0.92	0.0053	0.9196	0.9198	0.92	0.0053	0.92	0.92
B	SVM	0.92	0.0053	0.9194	0.9197	0.75	0.0167	undef	undef
	END	0.695	0.0203	undef	undef	0.84	0.0107	0.8433	0.8417
	EHND	0.93	0.0047	0.9302	0.9301	0.93	0.0047	0.9294	0.9297
C	SVM	0.955	0.003	0.956	0.9555	0.895	0.007	0.8946	0.8948
	END	0.91	0.006	0.9183	0.9141	0.76	0.016	0.7903	0.7749
	EHND	0.96	0.0027	0.96	0.96	0.955	0.003	0.9553	0.9551
D	SVM	0.985	0.001	0.9851	0.985	0.985	0.001	0.9851	0.985
	END	0.93	0.0047	0.9323	0.9311	0.92	0.0053	0.9217	0.9208
	EHND	0.985	0.001	0.9851	0.985	0.985	0.001	0.9851	0.985
Total	SVM	0.9238	0.0051	0.9337	0.9287	0.7825	0.0145	undef	undef
	END	0.8288	0.0114	undef	undef	0.8238	0.0117	0.8503	0.8368
	EHND	0.9488	0.0034	0.9487	0.9487	0.9475	0.0035	0.9474	0.9475

Table 7.1: Evaluation of SMO, ENDs, and EHNDs on the data given in Figure 7.1.
All methods were implemented in `Java` within the `Weka` framework. SMO was also the base-learner for ENDs and EHNDs, always using the same (standard) parameterization, and fitting logistic regression models to the outputs of the SVM to obtain class probability estimates. ENDs and EHNDs were built using 20 members of the ensembles. The hierarchy was defined as $\{\{A1, A2, A3, A4\}, \{B1, B2, B3, B4\}, \{C1, C2, C3, C4\}, \{D1, D2, D3, D4\}\}$.

(a)



(b)



(c)

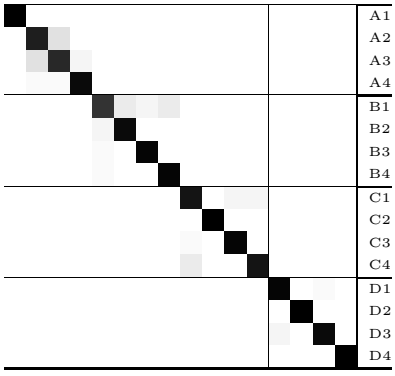


Figure 7.2: Visualization of the confusion matrices for the cross-validation of SVMs, ENDs, and EHNDs over the data given in Figure 7.1.

(a) Confusion of SVMs (pairwise coupling).

(b) Confusion of ENDs.

(c) Confusion of EHNDs.

The darkness is normalized per row by the number of instances for a class.

Group	Training				Cross-validation			
	TPR	FPR	PPV	F1	TPR	FPR	PPV	F1
A	0.25	0.05	undef	undef	0.25	0.05	undef	undef
B	0.705	0.0197	undef	undef	0.465	0.036	undef	undef
C	0.25	0.05	undef	undef	0.25	0.05	undef	undef
D	0.25	0.05	undef	undef	0.275	0.048	undef	undef
Total	0.3638	0.0424	undef	undef	0.31	0.046	undef	undef

Table 7.2: Impact of an improper hierarchy on performance of EHNDs.
EHNDs were used as above (Table 7.1), but the hierarchy was defined as {{$A1$, $B1$, $C1$, $D1$}, {$A2$, $B2$, $C2$, $D2$}, {$A3$, $B3$, $C3$, $D3$}, {$A4$, $B4$, $C4$, $D4$}}.

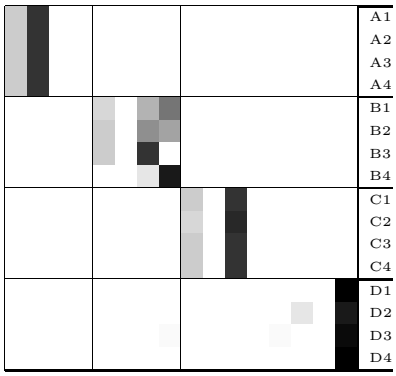


Figure 7.3: Impact of an improper hierarchy on confusion of EHNDs.
Although use of an improper hierarchy has a strong impact on performance of EHNDs, here the actual superclasses are not confused.

separated. The accuracy decreases dramatically (see Table 7.2), although as a matter of fact the actual superclasses are not confused (see Figure 7.3). After all, they are as easy separable as before.

7.4 Discussion

The pronounced hierarchical structure that is completely reflected in the features may lead to the observed strong improvement of EHNDs in comparison to both, ENDs and SVMs. Although neither ENDs nor SVMs do confuse superclasses in this example, as the visualization of the confusion matrices shows (Figure 7.2), they will have to find more difficult decision boundaries: Some NDs could be to separate groups of classes from each other, where the members of one group could be spread through all superclasses. The HNDs are prevented from doing so. Pairwise coupling SVMs on the other hand could collect many false votes for classes trying to classify classes they were not trained for, but which are very similar to the respective class. However, in both scenarios the decision between superclasses remains quite clear.

Our second experiment shows the dramatical impact of an improper hierarchy on HNDs. Here the HNDs are hold to find difficult decision boundaries. Since NDs can be bound to difficult decision boundaries on a high level only accidentally, such an erroneous bias of a single ND can be compensated for by other NDs in an ensemble. EHNDs will not be able to do so, if all ensemble members are bound to the same misleading hierarchy. Thus in this case there will be a high correlation of errors.

7.5 Conclusion

The evaluation of EHNDs on synthetic data exhibiting a hierarchical structure of classes confirmed the claim of their superiority over methods which do not incorporate information concerning the structure of data. However, since the data are synthetic the hierarchical structure was constructed quite pronounced and it was completely reflected by the feature space of the data. For applications of EHNDs to natural data the task remains first to find a feature space representing the hierarchical structure well. But even in case of success for this representation the hierarchy can not be expected to be that pronounced.

It was demonstrated that using improper hierarchies for building HNDs dramatically decreases accuracy of the ensemble of HNDs as well. This observation emphasizes even more the importance of finding a feature space reflecting the assumed hierarchy for use by EHNDs.

Part III

Application

Chapter 8

Application to Protein Classification

We are now going to describe possible biological applications for EHNDs. While hierarchies are ubiquitous in biology, most hierarchical-like structured data are not available as pure hierarchy. This will be discussed in the first section. We chose structural classification of proteins based on primary and secondary sequence (also known as 'fold recognition') as an application to evaluate EHNDs in comparison to other methods. The tasks involved with fold recognition and some related approaches and competing methods will briefly be introduced in the second section, the evaluation is reported in the third one. Finally we will state some conclusions related to this field of application.

8.1 Hierarchically Structured Biological Data

8.1.1 Sequence Similarity

First of all, sequences of DNA, RNA, or amino acids can be ordered by means of sequence comparison. Sequences more similar to one another are treated as being more closely related than sequences showing more dissimilarities since dissimilarities are based on mutations and the accumulation of mutations is mainly a matter of time of divergence. Thus the common ancestor of two sequences is most likely to be the older the more dissimilar the sequences are and trees covering the evolutionary relationships between molecular entities like proteins usually parallel phylogenetic trees of the respective organisms (see Figure 8.1). However, the measure of similarity of sequences is anything but trivial. A great deal of research is currently aiming at suitable definitions of sequence similarity. In recent years there were several redefinitions of sequence similarity proposed e.g. by means of algorithms like PSI-BLAST [8], Hidden Markow Models [16], or profile-profile alignment [133]. See for more general overviews e.g. [78], or [150] regarding the classical approach, or [59] emphasizing the probabilistic approach to sequence analysis which makes perfectly sense since the evolutionary forces acting on biological sequences are also thought of to work stochastically. Partially depending on function and probably on interaction with other proteins [118] proteins and protein families differ markedly in their evolutionary conservation. So stating evolutionary relationships based on sequence similarity will often be quite ambiguous, as we have stated above.

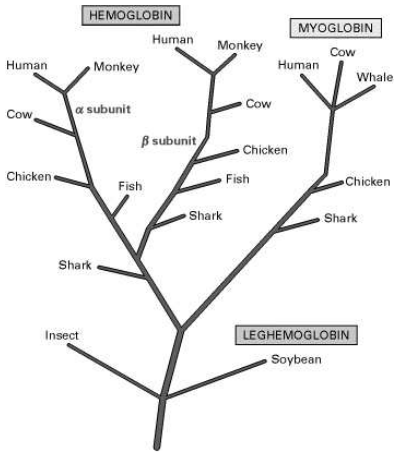


Figure 8.1: Evolutionary tree of globins.
This evolutionary tree is showing relations between several globins throughout several kingdoms, starting from the most primitive oxygen-binding proteins in plants, leghemoglobins. Sequence comparisons have revealed that evolution of the globin proteins parallels the evolution of vertebrates. Major junctions occurred with the divergence of myoglobin from hemoglobin and the later divergence of hemoglobin into the α and β subunits. The Figure is taken from [103, Figure 3-12] which is an adaption from [44].

8.1.2 Functional Classification

Other hierarchies of genes or proteins are based on their function (e.g. Gene Ontology [142], [143], [144], Enzyme Classification [113]). It is commonly accepted that the function of genes and proteins is based on their sequence and also set out to evolutionary forces. Nevertheless, the general relation of biological sequences and biological functions, i.e., a function f : *biological sequences* $\rightarrow$ *biological functions*, that maps a biological sequence to its biological function, is not yet determined. The vision of defining this relation is a main motivation for research in the fields of bioinformatics (see [15] for an overview).

Unfortunately, these hierarchical relationships of protein functions are usually not pure hierarchies in the notion of Definition 6.3. Gene Ontology resembles a directed acyclic graph. Furthermore one protein can have several functions thus belonging to several functional classes. Therefore the respective hierarchy can not be represented as a tree nor can a classification scheme be applied that assumes classes to be mutually exclusive.

8.1.3 Structural Classification of Proteins

Molecules can also be sorted with respect to their structure. By means of such sorting pure hierarchies are provided, even though different aspects of structural properties might lead to different hierarchies. Note that usually sequence similarity is a criterion for building samples at deeper levels in such hierarchies. Furthermore according to common opinion the function of a protein is closely related to its structure as structure is related to sequence. Nevertheless the number of structural motifs seems to be quite limited and certain structures are repeatedly observed among proteins with very different and apparently unrelated sequences of amino acids. So current estimates suggest that there are about 1000 unique protein folds [37], [76], and [152]. Two forces might have played a role in the limitation of actual variety of folds: divergent evolution of protein function (since all folds are derived from a relatively small group of shared common ancestors) and convergent evolution of protein structure (since certain folds are biophysically much more favored

Class	Folds	Superfamilies	Families
All alpha proteins	179	299	480
All beta proteins	126	248	462
Alpha and beta proteins (α/β)	121	199	542
Alpha and beta proteins ($\alpha+\beta$)	234	349	567
Multi-domain proteins	38	38	53
Membrane and cell surface proteins	36	66	73
Small proteins	66	95	150
Total	800	1294	2327

Table 8.1: SCOP Classification Statistics.
SCOP: Structural Classification of Proteins. 1.65 release 20619 PDB Entries (1 August 2003). 54745 Domains. 1 Literature Reference (excluding nucleic acids and theoretical models) (Statistics obtained from `http://scop.berkeley.edu/count.html`.)

and thus may have been created independently in multiple cases [75] – as e.g. morphologously analogous structures often occur in relatively unrelated species). These relations between fold space and protein evolution are also discussed in [135]. (For a gentle introduction to the basics of protein structure see also [22]. Generally we used also [60], [98], [25], and [26].)

One of the first structural classifications of proteins was a flat hierarchy based on the predominant secondary structure element and consisted of four groups: all α (containing proteins based almost entirely on α-helical structure), all β (the members being based on β-sheet), α/β, and $\alpha+\beta$ (both being based on a mixture of α-helices and β-sheets, where α-helix and β-sheet motifs are interwoven in α/β structures, but not in $\alpha+\beta$ structures [99]. This classification scheme has been extended and adjusted later on, resulting in the most prominent hierarchical structural classification schemes for proteins: SCOP and CATH. The third of the major structural classification schemes of proteins, FSSP [83], is based on somewhat different principles not clearly discriminating folds or superfamilies and will not be considered in the remainder. For a systematic comparison of SCOP, CATH, and FSSP see [79].

8.1.3.1 The SCOP-Database

The SCOP-database (Structural Classification Of Proteins [111], [102], [10]) provides a four level hierarchy. The top level classification is the *class*, mainly with respect to the secondary structure composition of the protein. The main classes are the four classes already proposed by Levitt and Chothia in 1976 (all α, all β, α/β, and $\alpha+\beta$, as described above) among other less important classes like *multi-domain* and *small proteins*. Each class consists of several *folds*, the folds of *superfamilies*, these of *families*. Families consist of single domains which exhibit a distinct sequence similarity thus being evolutionarily closely related. E.g. all the proteins given in Figure 8.1 considered as being evolutionarily related belong to the family *globins*, being part of the superfamily *globin-like*, the fold *globin-like*, and are subsumed under the class *all-alpha* at top-level of the SCOP-hierarchy. An overview concerning the most important classes and their content of folds, superfamilies, and families is given in Table 8.1.

Proteins in SCOP are mainly manually classified by visual judgment. While clustering of proteins with similar sequences, or very similar structures and functions into families and superfamilies is more or less indisputable, the choice for classes is somewhat arbitrary. There are other classifications possible. Decision for a SCOP class is made by visual inspection. Superfamilies group families together which are believed to share a common evolutionary origin. Proteins sharing a

common fold exhibit structural similarities due to favorable packing arrangements and chain topologies, although they may sometimes also be distantly evolutionarily linked as structure is much more highly conserved than sequence in distantly related proteins: Proteins that have diverged beyond detectable sequence similarity have often retained the architecture and topology of their ancestral fold nevertheless. For a more detailed discussion of the SCOP database and evolution of protein structures we refer to [127].

Related to SCOP as a connection to PDB (Protein Data Bank [20]) is the ASTRAL compendium ([29], [32], [31]). Basically ASTRAL provides a linkage of protein structures and their associated sequences as this linkage is not always easily to be derived from original PDB files. So ASTRAL provides subsets of proteins featured both by PDB and SCOP as a mapping of PDB contained proteins to SCOP, preferably also based on PDB entries providing high resolution and regularity of crystallographically determined protein structures. The subsets are defined by maximal sequence similarity thus allowing to consider sets of proteins excluding homologues to a certain degree of sequence identity. This is an important issue to properly evaluate methods based on alignments.

8.1.3.2 The CATH-Database

The CATH-database [115], [121] (see also [116]) is based on principles similar to those of SCOP, comprising four major levels: Class, Architecture, Topology, and Homologous Superfamily. The name CATH is an acronym for these levels, and these four levels correspond the three major SCOP levels. On class level CATH does not distinguish α/β from $\alpha + \beta$. A fourth class collects domains exhibiting only few secondary structures. Other classes are less important (see Table 8.2). Topology level corresponds to Fold level of SCOP. Between Class and Topology, on the level of Architecture proteins are clustered manually with respect to similar orientation of secondary structures in space, regardless of their connectivity. Thus proteins sharing the numbers C-A-T exhibit the same fold. See Figure 8.2 for an illustration of the first three levels of CATH.

Homologous superfamilies, the fourth level, groups proteins according to whether there is evidence based on sequence, structure, or function supporting an evolutionary relationship. Within homologous superfamilies, proteins are clustered into sequence families based on several levels of sequence identity. CATH provides therefore three levels which are not separated in family level of SCOP. Statistics regarding the current version is presented in Table 8.2.

In difference from SCOP CATH does incorporate some automation in classifying protein structures. Its hierarchical classification might therefore be more suitable to machine learning methods to be learned.

8.1.3.3 Summary

SCOP and CATH are both well established hierarchies of structural classification of proteins. There are datasets used for evaluation of methods to identify the fold of a protein based on either of SCOP or CATH. We will also refer to a dataset with respect to either of both databases. But beforehand we will shortly survey what is implied in the task of identifying the fold of a protein.

8.2 Fold Recognition

The identification of the fold of a protein is to be seen within the wider horizon of prediction of the three-dimensional structure of a protein from its one-dimensional sequence of amino acids. Ostensible fold recognition is a classification problem and

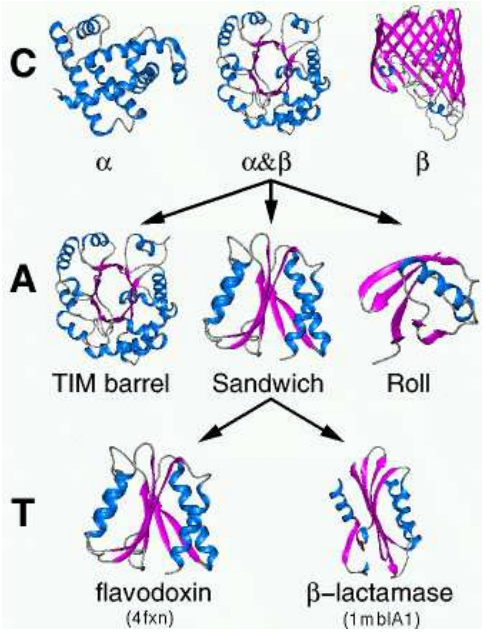


Figure 8.2: Illustration of the CATH hierarchy.
Illustration of the CATH hierarchy as provided by
`http://www.biochem.ucl.ac.uk/bsm/cath_new/images/cathhier.gif`.

C	A	T	H	S	N	I	D
Mainly Alpha	5	227	428	948	1713	3946	10155
Mainly Beta	19	139	292	951	2344	5011	14259
Alpha Beta	12	368	648	2010	3631	8639	23025
Few Secondary Structures	1	86	91	114	225	378	952
Multi-domain chains	1	1053	1057	1071	2186	5801	12471
Preliminary single domain assignments	1	371	374	422	479	789	1663
Multi-domain domains	2	31	31	49	67	139	287
CATH-35 Sequence families	1	997	997	997	1108	2154	3431
Fragments from multi-chain domains	1	28	28	30	33	56	106

Table 8.2: CATH Classification Statistics.
CATH 2.5.1 release (28 January 2004).
(Statistics obtained from `http://www.biochem.ucl.ac.uk/bsm/cath/releases.html`.)

thus it is often treated by classical means of machine learning. But moreover, '*fold recognition*' is also related to a certain method of modeling the tertiary structure of a protein given its primary structure. Generally there are three main methods for assignment of tertiary structure to a protein of unknown tertiary structure, referred to as '*comparative modeling*', '*ab initio*', and '*fold recognition*' (see for an overview also [24]). *Comparative modeling* (also known as '*homology modeling*') is a method to be used preferably whenever there is a clear relationship between the sequence of a protein of unknown structure (also referred to as '*target*') to the sequence of a protein of known structure ('*template*') [93]. *Ab initio* methods (or nowadays '*novel fold*' or '*new fold*', where *ab initio* is a subset of methods relying only on physical principles and not on any existing structure on sequence data) are applied to proteins where no protein fold on sequences of even low similarity is known [36]. *Fold recognition* is the intermediate method to be preferably employed when a template of known fold can be identified in the absence of recognizable sequence similarity. Of course, the threshold between fold recognition and homology modeling based on sequence comparison is constantly changed due to advances in sequence analysis [72].

The template for fold recognition methods is often considered to be distantly related while generally two proteins will be assumed to be related, if a sequence similarity larger than random is found: sequence similarity implies homology, but not vice versa. But taking into account sequence similarity as the only argument for homology can also be misleading, since the so called '*twilight zone*' of sequence similarity corresponds to about 25% of identical amino acids in an optimal alignment between protein pairs. This is why fold recognition earns relevancy even though homology modeling reaches out to regions of even far sequence similarity. Detecting the distant homologies by means beyond of pure sequence comparison relies on the chance that some aspects of function, like the arrangement of active site residues, are conserved although the sequence similarity is undetectable. Thus by means of fold recognition also the prediction of some aspects of protein function might become possible.

However, the first step towards fold recognition in this notion of a modeling method is the correct assignment of a known fold to a protein of unknown structure. This first step is often also referred to as 'fold recognition' itself for suggesting reasons. In the remainder we refer to 'fold recognition' in this restricted notion as a classification problem.

There are basically two approaches towards fold recognition. Godzik [72] discerns these as biological versus physical. Roughly these categories also could be referred to with respect to the methodology as alignment methods versus machine learning methods.

8.2.1 Alignment Methods

The biologist's approach is to explain the nature in terms of patterns of evolution. These patterns are especially found in sequences of nucleic acids or amino acids. Thus comparison of sequences by alignment methods is *prima facie* the preferable choice of means to find similar and therefore related sequences. Methods based on alignments of sequences are also employed to detect the presumably correct fold of a target sequence. Since sequence similarity for typical targets of fold recognition is quite low, the respective methods are enriched by complex scores to take into account e.g. profile-profile alignments or alignments of secondary structure elements. Profile-profile alignments are e.g. used by [86], [133], or [155]. Taking into account structural information to improve fold recognition was already considered in [5], [131], and [62], and recently in [108]. Though profile-profile alignments already are highly correlated with similarities between secondary structure elements [155],

recently methods were proposed combining both approaches explicitly, see [107], [21] (also taking into account enzyme classification), and [71].

Four alignment based methods were taken into account for comparison with our method, despite the fact that alignment based methods do generally perform better in fold recognition than machine learning methods:

1. **GenTHREADER.** GenTHREADER is a simple threading approach introduced by Jones [86]. It is based on an algorithm making use of a sequence profile and analyses of alignments by using energy potentials.

2. **PDB-BLAST.** Rychlewski et al. [133] introduced a two-step PDB-BLAST protocol which uses PSI-BLAST [8] to build a sequence profile for a target based on the non-redundant database of known sequences. The generated profile is then aligned to all target sequences using BLAST [7].

3. **MANIFOLD.** MANIFOLD is based on a combination of secondary structure alignment, PDB-BLAST and enzyme code similarity and was introduced by Bindewald et al. [21]. The three contributions to the method are weighted by training a two-layer neural net.

4. **Alignment combination.** Gewehr et al. [71] combine profile-profile alignments using a log average score in combination with secondary structure element alignment.

As the evaluation of our method in comparison to these methods will show, the gap between alignment based methods and pure machine learning methods is not always that big. It heavily depends on the dataset. Machine learning methods are well able to reach the level of accuracy of alignment based methods in some cases.

8.2.2 Machine Learning Methods

As the 'biological' methods of fold recognition – based on homology recognition – assume that structural similarity results from the distant relation between two considered proteins, they are tackling the question whether or not a protein sequence belongs to a given family of proteins with respect to a specific set of rules of mutations. As Godzik [72, page 533] states concerning these methods:

> *The structure was not used directly and entered the picture only by restricting accepted mutations in different ways at different positions. At the same time, most proteins fold on their own [...], without checking what the structure of their homologs is in databases but following physical laws governing their behavior.*

This declares perfectly the motivation to pursue other than alignment based methods that hopefully allow detecting the implied *physical* laws. According to common opinion the sequence of amino acids perfectly determines the three dimensional structure [11], a principle that inspires *ab initio* methods in protein fold prediction. In the field of protein fold recognition some methods take also into account the energy of a fold. Such methods are referred to as threading ([87], [117] – compare other sources cited by Godzik [72]). Of course, this approach is often combined with alignment based methods, also in some of the methods cited above.

Energy functions generally must be simplified to be computable. Another possibility to simplify properties of sequences is extraction of feature vectors. This is where machine learning can tackle the problem. The physicist seeks a function from sequence space to (continuous) fold space, while machine learning seeks a function from feature space to (discretized) fold space, given a function from sequence space

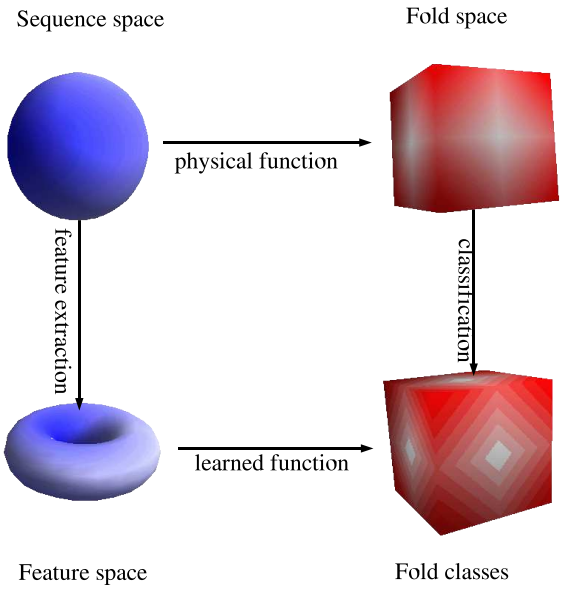


Figure 8.3: The approximation of the assumed physical function by means of machine learning.
Machine learning tackles part of the problem of the physical approach: The physicist seeks a function from sequence space to (continuous) fold space, while machine learning seeks a function from feature space to (discretized) fold space, given a function from sequence space to feature space. Machine learning thus performs part of the physical task and hopefully helps approximating the physical function.

to feature space (Figure 8.3). (There are other machine learning approaches, e.g. Hidden Markov models, operating directly on sequences, of course. So for example Raval et al. [126] make use of HMMs for fold recognition.)

The task in applying machine learning to protein fold recognition is on the one hand to detect feature spaces that describe proteins well with respect to structure. On the other hand a broad variance of machine learning methods is available, so the question is which one to use. We will therefore apply our method of EHNDs in comparison to other methods to test its usefulness. This is suggesting since hierarchical structures are well defined for proteins, as was described above.

8.2.3 Feature Spaces for Proteins

As Figure 8.3 shows an important step to prepare problems for machine learning approaches is the representation of the problem in a feature space. Since the fold depends according to common opinion [11] on the sequence, a promising feature space should provide as much as possible information concerning the sequence. In the literature we found basically three approaches to this task which will now be shortly presented:

8.2.3.1 Composition – Distribution – Transition

The by far best known set of features for protein fold recognition was proposed by Dubchak et al. [56] and [57] and often used since then not only by Dubchak herself [53] but also by many other machine learning approaches to the problem. Ding and Dubchak [53] provided also a training and a test set of proteins and the prepared features according to [56] and [57]. Although the thereby provided SCOP-classification

Property	Group 1	Group 2	Group 3
Hydrophobicity	Polar: {R,K,E,D,Q,N}	Neutral: {G,A,S,T,P,H,Y}	Hydrophobic: {C,V,L,I,M,F,W}
Normalized van der Waals volume	0–2.78: {G,A,S,C,T,P,D}	2.95–4.0: {N,V,E,Q,I,L}	4.43–8.08: {M,H,K,F,R,Y,W}
Polarity	4.9–6.2: {L,I,F,W,C,M,V,Y}	8.0–9.2: {P,A,T,G,S}	10.4–13.0: {H,Q,R,K,N,E,D}
Polarizability	0–0.108: {G,A,S,D,T}	0.128–0.186: {C,P,N,V,E,Q,I,L}	0.219–0.409: {K,M,H,F,R,Y,W}

Table 8.3: Amino acid attributes and the mapping of amino acids onto sets of three groups.
For each property the set of amino acids is mapped onto a set of three groups as defined above. The mapping is based on previous publications as stated by [57]: Hydrophobicity: [38], Normalized van der Waals volume: [61], Polarity: [77], Polarizability: [33].
Table originally provided by [57, Table I].

is not up to date anymore these data are used even by recent publications and provide therefore good means for comparison of different approaches.

The idea of Dubchak et al. was to describe the properties of a sequence in three different descriptors called composition, distribution, and transition, based on groups of amino acids.

Groups For application of descriptors the set of amino acids is mapped onto a set of groups. The groups used by Ding and Dubchak [53] were hydrophobicity, normalized van der Waals volume, polarity, and polarizability. The mapping for these sets of groups is defined by Table 8.3. Of course, the set of amino acids is also a set of groups (of size 20) as well as the predicted states of secondary structure (as helix, sheet, and coil).

Descriptors The descriptors can be applied to each of the sets of groups. This works as follows:

- **Composition.** The composition describes the percentage of amino acids of the sequence per group. This results in a number between 0 and 1 for each group, where the sum is 1.
- **Distribution.** The distribution consists of five numbers for each group: the fractions of the entire sequence where the first residue of the respective group occurred, and where 25%, 50%, 75%, and 100% respectively of those are contained.
- **Transition.** The transition provides a number for each pair of groups out of a given set of groups which is the count of transitions from one group to the other or vice versa within a given sequence.

Summary For the sets of groups of three members like secondary states or the groups defined in Table 8.3, the composition descriptor and the transition descriptor each provide three numbers, the distribution descriptor five times three. Thus for each of these groups a feature vector of dimensionality 21 is provided. For the single amino acids only the composition is computed which results in combination with the length of a sequence also in a feature vector of dimensionality 21. Distribution and transition for the single amino acids would result in very high dimensional feature spaces. The combination of the feature vectors provides a feature space of $6 \times 21 = 126$ dimensions.

It is often reported and conforms also to our experience that best separation of structural classes and folds is allowed by the 20 features based on amino acid composition and the 21 features based on secondary structure.

8.2.3.2 Auto-Correlation Function

For separation of structural classes another feature space is proposed and reported to perform well by Bu et al. [30]. They propose to represent an amino acid sequence as feature vector consisting of the values of a so called auto-correlation function. Therefore the sequence of amino acids is firstly to be transformed into a sequence of numbers dependent on an arbitrary amino acid index [145]. An amino acid index provides a number for any amino acid with respect to a certain physicochemical property. Tomii and Kanehisa [145] collected 402 sets of amino acid indices which all could possibly be employed for usage in such a transformation of an amino acid sequence to a sequence of numbers. Resulting is for a sequence $(a_1, a_2, \ldots, a_N), a_i \in$ *aminoacids*, a sequence $(h_1, h_2, \ldots, h_N), h_i \in \mathbb{R}$.

The auto-correlation function proposed by Bu et al. [30] is based on [41] and [156] and defined for a number n and a sequence of numbers h_i, $i = 1, \ldots, N$ by:

$$r_n = \frac{1}{N-n} \sum_{i=1}^{N-n} h_i h_{i+n}. \tag{8.1}$$

Choosing for n the numbers $1, \ldots, m$ results in m values $r_1, \ldots, r_m$. These numbers are used as features describing the original sequence of amino acids.

Bu et al. tested all 402 amino acid indices collected by Tommi and Kanehisa for prediction of structural class for amino acid sequences. They found best prediction accuracy for the index proposed by Oobatake and Ooi [114] which describes the average non-bonded energy per residue. Other well performing amino acid indices were found to be strongly correlated to the Oobatake-Ooi index.

The optimal value for m depends on the chosen amino acid index. For Oobatake-Ooi Bu et al. suggest to set $m = 30$. However, the optimal value is also dependent on the data. Note that m directly destines the number of features, so m should generally be a smaller value for smaller databases in order to avoid overfitting. Bu et al. used a database consisting of 359 proteins, so a value around 30 for m seems perfectly reasonable.

The feature space based on auto-correlation of amino acid indices promises to be an interesting complement to the feature space describing composition, transition, and distribution, since it provides more information concerning the sequence of amino acids.

8.2.3.3 Occurrence of Motifs

A third approach to transform a sequence to a feature space is to count the occurrence of certain motifs within a sequence. The question arises which motifs are to be counted. This question can be tackled coming from two directions: Derive motifs that show a high ability to discriminate classes within the training data, or to use motifs generally found in bigger databases of sequences.

An example for the first approach is the work of Luo et al. [154]. By stepwise discriminant analysis they chose 296 peptides whose frequencies are used for final prediction of structural classes, among them 12 single amino acids, 62 dipeptides, 130 tripeptides, 23 tetrapeptides, 66 pentapeptides, and three hexapeptides. It could be an interesting question, whether better motifs could be derived using an *apriori* [4] like approach. Note that Luo et al. derive only continuous patterns of amino acids, whereas motifs generally would allow wildcards within a pattern.

Thus this approach to create a feature space could be a rewarding topic of further research in order to derive more general patterns in a more efficient manner. As it is it was only shown by Luo et al. to perform well on level of structural class for the four main classes (α, β, $\alpha + \beta$, and α/β) for a dataset allowing sequence similarity of 40% (PDB40-B [28] and PDB40-J [119]). Similar approaches are inspired by text-mining and create features based on n-grams, like [95], [149], and [43]. We did not make further use of feature spaces of this kind. Nevertheless, this approach seems to be well worth further examination.

The other direction is proposed by Ben-Hur and Brutlag [19]: Based on any established database of motifs the occurrence of motifs in a sequence is counted. So the dimensionality of the feature space is the number of known motifs and will be quite high. The idea is to use a simple linear kernel as dot-product of two feature vectors:

$$K(x, x') \quad = \quad \Phi(x) \cdot \Phi(x'), \tag{8.2}$$

where $\Phi(x) = (\phi_m(x))_{m \in \mathcal{M}}$, and $\phi_m(x)$ is the number of occurrences of the motif m in sequence x for the given motif database $\mathcal{M}$. Since a motif will appear usually only once in a sequence, this kernel basically counts the number of motifs that are common to both sequences thus providing a similarity measure for two sequences generally emphasizing frequent patterns as provided by a database of motifs.

This approach has some intrinsic drawbacks: the evaluation is very difficult to be performed reliably, since the motifs are derived from a generally unknown set of proteins that could contain also proteins of the test set. Furthermore many motif databases are constructed in a supervised way from known protein families. Ben-Hur and Brutlag therefore propose to use the eBLOCKS database [140]. This database of motifs is based on groups of proteins with varying levels of similarity, i.e., in an un-supervised way. Motifs are made out of aligned blocks of sequences using the eMOTIF method [112]. Nevertheless, the evaluation of prediction performance based on such feature spaces remains somewhat unreliable.

This approach theoretically seems to fetch some of the benefits of profile-profile alignments. But unfortunately, we found this feature space not to be helpful in our tests on different datasets.

8.2.3.4 Summary

The feature spaces for representation of proteins proposed so far seem far from perfect. We found the extension of the feature space on composition, distribution, and transition of predefined groups by values derived for amino acid indices by means of an auto-correlation function to be sometimes helpful. A problem here could be that the number of features in comparison to the number of instances becomes to big. The motif based feature space on the other hand did not turn out to be supportive to our tests, but it seems nevertheless promising and well worth further studies. As noted in other studies [22], in machine learning approaches to bioinformatics many new methods are developed or applied, but in most cases essentially the same feature spaces are employed. It might therefore be rewarding to put more effort in development of more sophisticated feature space representations of proteins with respect to a certain problem.

8.2.4 Related Approaches

8.2.4.1 Voting of Coupled Binary Classifiers

Ding and Dubchak [53] tested neural nets and several combinations (*all-versus-all* and *one-versus-others*) of support vector machines on the task of protein fold prediction. They also provided data based on feature sets of composition, distribution,

and transition as described above which was often used afterwards for evaluating machine learning approaches. Each protein was represented in several feature spaces. The respective learners for each feature space were combined by a voting procedure.

Similar and often also simpler approaches were used by other groups. Ding and Dubchak are well known mainly for the provided data (see below), while other groups evaluated similar approaches on different data.

8.2.4.2 Classification with Respect to the Hierarchy

Chung et al. [40] propose a hierarchical classifier architecture for separating first structural classes of proteins, then separately classifying the folds belonging to the respective structural class. The hierarchical architecture allows different kinds of base classifiers. Chung et al. use neural networks and support vector machines directly as a multi-class classifier as proposed in [100]. Thus their method is somewhat similar to ours but does not further take into consideration reduction of polytomous to dichotomous classification problems. Also their approach seems a proposal especially adapted to the problem of hierarchically protein fold classification, although it might be easily to be generalized. Their method is evaluated on the feature space and data proposed by Dubchak et al. [57].

Huang et al. [85] furthermore propose a method to select the most promising features, based on a similar architecture as [40] but now combining the classifiers for structural class level (level 1) and fold level (level 2) in a parallel manner by an AND-gate since level 2 could gain advantage from features which were not selected for level 1 (thus level 2 gets unfiltered features for classification). Their method is also evaluated on the feature space and data proposed by Dubchak et al. [57].

8.2.4.3 Bayesian Classification

Chinnasamy et al. [35] proposed a method based on a Bayesian classifier, called BAYESPROT. Obviously the method is specialized to protein fold classification and uses the feature space proposed by Dubchak et al. (see section 8.2.3.1). The extracted features are discretized to four discrete states. For the discretized features Tree-Augmented Bayesian Networks were built. By Mean Probability Voting (MPV) then either the structural class or the fold is classified. These two predictions are connected.

BAYESPROT is evaluated on data also used by many other methods (see section 8.3.1.1), but it seems more successful than the other methods based on neural networks and SVMs (see section 8.3.2.1).

8.2.5 Summary

While machine learning probably will not detect the pure physical laws of folding of proteins it will hopefully help to reveal rules that help detecting and defining these laws. Although alignment based methods usually will gain better results than machine learning methods do, they usually will explain nothing concerning the process of protein folding. Thus applying machine learning methods to fold recognition remains an interesting enterprise from a scientific point of view.

Clearly, support vector machines on the other hand do not lend themselves to human inspection. But other algorithms can actually be more accurate than support vector machines on certain datasets. The here proposed method of EHNDs is compatible with most base learners.

Nevertheless, finding promising feature spaces is a task of its own. It first of all requires biochemical expertise. But also machine learning methods might help once

again, e.g. to discern feature spaces that are able to represent a protein well from others that do not.

Note again that the line between alignment based methods and machine learning methods is not always clear. Another possibility would be to address alignment based methods as examples of lazy learners. This would once more point out that they are not able to provide a model. However, we would like to emphasize the importance of feature spaces to the methods we address as 'machine learning' methods in a somewhat restricted notion. This dependence makes a comparison of alignment based methods and feature space based methods difficult.

8.3 Evaluation

8.3.1 Data

8.3.1.1 Ding and Dubchak

The training and test set provided by Ding and Dubchak [53] are available under `http://www.nersc.gov/~cding/protein`. Both sets are also appended in `.arff` format in the appendix. For training Ding and Dubchak selected 27 folds which have at least seven proteins in the database. These 27 folds in the data represent all major structural classes: α, β, α/β, and $\alpha+\beta$. Note that this criterion leaves only 320 proteins of the originally presented 605 in the training set. For an independent test set Ding and Dubchak selected 386 representatives from the PDB-40D [101]. This set contains the SCOP sequences of less than 40% sequence identity with each other. The representatives were to present the same 27 folds as the training set. Proteins were excluded when they had a sequence identity of more than 35% with any of the proteins in the training set. For fold recognition this level of sequence identity of 35% is still quite high. Consequently the test set of Ding and Dubchak is known to contain some homologous proteins with respect to the training set [21]. Note that the test set at this point in time available under the cited URI contains only 385 proteins (and so does our test set).

Obviously the proportion between size of training set (320) and size of test set (385) is less than optimal. Also note that the SCOP classification provided by Ding and Dubchak for their dataset is partially obsolete. Nevertheless we used these data as they were (including the original feature space – see section 8.2.3.1) for comparing our method with several other machine learning approaches that used the same data.

8.3.1.2 Ding-Dubchak-Bindewald

Bindewald et al. [21] and following them Gewehr et al. [71] used a modification of the datasets of Ding and Dubchak. Bindewald et al. report to make use of a more recent SCOP version (1.53) and to have removed homologous proteins from the training set which result in a BLAST [7] e-value of less than 10^{-3} with any protein of the testset. According to Jan Gewehr [70] this reduced dataset was also used in [71].

The mapping of the proteins to a more recent version of SCOP than the original mapping by Ding and Dubchak (version 1.53, as used by Bindewald et al.) results in further reduction of both training and test set. The procedure leaves 240 proteins for the training set and 373 proteins for the test set. Both sets cover 27 folds, as the original sets, but some folds are represented within the training set by less than seven instances.

We used for the reduced sets the original feature space of Ding and Dubchak (section 8.2.3.1).

The sets as used here are provided in `.arff` format in the appendix.

8.3.1.3 McGuffin-Bindewald

McGuffin and Jones [108] provide another dataset of 542 nonredundant domains based on CATH version 1.7. The set is divided into a 'unique' and a 'known' domain set. The unique set comprises 290 domains with folds unique within the data set. The known set comprises 252 domains each having at least one other matching fold in the data set. These data are available from `http://www.cs.ucl.ac.uk/staff/L.McGuffin/targets.html`.

Bindewald et al. [21] adapted the data to CATH version 2.4. Thus they had to remove four instances from the unique set because their domain definition has changed or was not part of CATH version 2.4. The resulting data consist of the known set of 252 instances covering 55 topologies and the unique set of 286 instances covering 274 topologies. Note that 16 of the 55 topologies covered by the known set are represented by only one instance. This fact makes the use of several machine learning approaches (including EHNDs using SVMs as base learner) impossible, since the fitting of a class probability estimation requires each class to have at least two members. Of course, this architecture of data sets is anything but favorable for machine learning approaches since one can not reasonably expect to find an appropriate generalization for classes represented by only one or two proteins.

For features based on secondary structure elements we made use of the DSSP [88] annotation as provided by McGuffin. The mapping of a DSSP state to a secondary structure class was defined according to [9] by:

$$\begin{aligned} [G|H|I] &\mapsto h, \\ [E|B] &\mapsto e, \\ [T|S|L] &\mapsto c, \end{aligned}$$

where L is given in DSSP output as space or in some cases as underscore. No further interpretation as e.g. removing short sequences of secondary structure elements was done.

All sets are provided in `.arff` format in the appendix as we used them.

8.3.2 Results

8.3.2.1 Ding and Dubchak

SVMs were fitted on the training set of Ding and Dubchak. We then trained an ensemble of 20 HNDs on the training set of Ding and Dubchak using SVMs as base learners with the parameter setting obtained by the fitting procedure. SVMs were used as described in section 7.3.

Furthermore we employed Bagging (40 iterations) with PART as base learners. These algorithms were used as implemented in `Weka`, Bagging according to [27], PART according to [67]. PART is a rule learning algorithm based on partial decision trees. Bagged PART was already found to perform better than SVMs in some cases [132]. Employing a rule learner may provide the additional benefit of possibly obtaining explicit knowledge.

We reached on the respective test set an accuracy of more than 58% for the EHNDs. We built also Ensembles of NDs that allow evaluation of the quality of hierarchy representation in the feature space in direct comparison with EHNDs. The results achieved with both, ENDs and EHNDs, are not worse than the results reported for other machine learning approaches. A comparison with other approaches using the same data – as we are aware of – is given in Table 8.4.

Approach	Prediction accuracy (Q)	
Ding and Dubchak		
NN (OvO)	41.8	%
SVM (OvO)	45.2	%
SVM (uOvO)	51.1	%
SVM (AvA)	56.0	%
Chung et al.		
RBFN	49.4	%
Hierarchical Structure (MLP)	44.7	%
Hierarchical Structure (RBFN)	56.4	%
Hierarchical Structure (GRNN)	45.2	%
Hierarchical Structure (SVM)	53.8	%
Huang et al.	56.36	%
Chinnasamy et al.	58.18	%
ENDs		
(SVM)	58.96	%
(Bagged PART)	57.64	%
EHNDs		
four structural classes (SVM)	58.18	%
five structural classes (SVM)	58.44	%
five structural classes (Bagged PART)	58.7	%

Table 8.4: Comparison of prediction accuracy for several machine learning approaches to fold recognition on the data of Ding and Dubchak.
The accuracy for other approaches is reported according to Ding and Dubchak: [53], Chung et al.: [40], Huang et al.: [85], and Chinnasamy et al.: [35]. These approaches used the training set and the test set as provided by [53]. We used the same data for training and testing ensembles of 20 NDs (ENDs) and ensembles of 20 HNDs (EHNDs). Both performed as well as the best of the other approaches, ENDs slightly better than EHNDs. Both ensembles employed SVMs as base learners using a radial basis function as kernel with parameters $C = 4$ and $\gamma = 0.125$. Finally we trained ENDs and EHNDs (20 ensemble members) employing Bagging (40 iterations) with PART (Bagged PART) as base learners. Here EHNDs performed slightly better than ENDs.
For EHNDs two different hierarchies were employed: a distinction of four structural classes (as by Chung et al. and Huang et al.), and the more appropriate distinction of five structural classes (as by Chinnasamy et al.).
Reported are results on the complete feature set (126 features) for the independent test set.

Group	Method	Training				Test			
		TPR	FPR	PPV	F1	TPR	FPR	PPV	F1
α	SMO	1	0.0014	0.9688	0.9842	0.6393	0.0091	0.6818	0.6599
	ENDs	1	0	1	1	0.6557	0.0049	0.7888	0.7161
	EHNDs (4)	1	0	1	1	0.7049	0.0094	0.7104	0.7077
	EHNDs (5)	0.9818	0.0004	0.9841	0.983	0.7377	0.009	0.7187	0.7281
β	SMO	0.7909	0.0134	undef	undef	0.453	0.0416	undef	undef
	ENDs	0.9818	0	1	0.9908	0.5726	0.0264	0.6801	0.6218
	EHNDs (4)	0.9909	0.001	0.9912	0.9911	0.5128	0.0316	0.644	0.571
	EHNDs (5)	0.9909	0.002	0.9829	0.9869	0.5128	0.0331	0.6339	0.567
α/β	SMO	0.9487	0.0055	0.9305	0.9395	0.5655	0.0379	0.5272	0.5457
	ENDs	0.9573	0.0054	0.9471	0.9521	0.5862	0.0401	0.517	0.5495
	EHNDs (4)	1	0.0017	0.984	0.9919	0.6	0.0335	0.5637	0.5813
	EHNDs (5)	0.9915	0.0009	0.9917	0.9916	0.5931	0.0339	0.5597	0.5759
$\alpha+\beta$	SMO	0.9211	0.0065	0.8737	0.8967	0.5806	0.0127	0.671	0.6225
	ENDs	1	0	1	1	0.5645	0.0108	0.7048	0.6269
	EHNDs (4)	0.9474	0	1	0.973	0.5484	0.0182	0.5959	0.5712
	EHNDs (5)	0.65	0.0127	undef	undef	0.2571	0.0345	undef	undef
small proteins	EHNDs (5)	1	0	1	1	0.9259	0.0028	0.9615	0.9434
Total	SMO	0.9	0.0076	undef	undef	0.5455	0.0304	undef	undef
	ENDs	0.9781	0.002	0.9806	0.9794	0.5896	0.0256	0.6399	0.6137
	EHNDs (4)	0.9906	0.001	0.9911	0.9909	0.5818	0.0266	0.6166	0.5987
	EHNDs (5)	0.9688	0.0018	undef	undef	0.5844	0.0276	undef	undef

Table 8.5: Evaluation of SVMs, ENDs, and EHNDs for Ding and Dubchak data.
The table provides more detailed evaluation of classification performance of an all-versus-all SVM (SMO) classification, ENDs (SVM), and EHNDs (SVM) (same as in Table 8.4). EHNDs were trained with two different hierarchies, discerning four structural classes (4), and five structural classes (5), respectively. The group 'small proteins' is given separate only for EHNDs trained on the respective hierarchy (5).

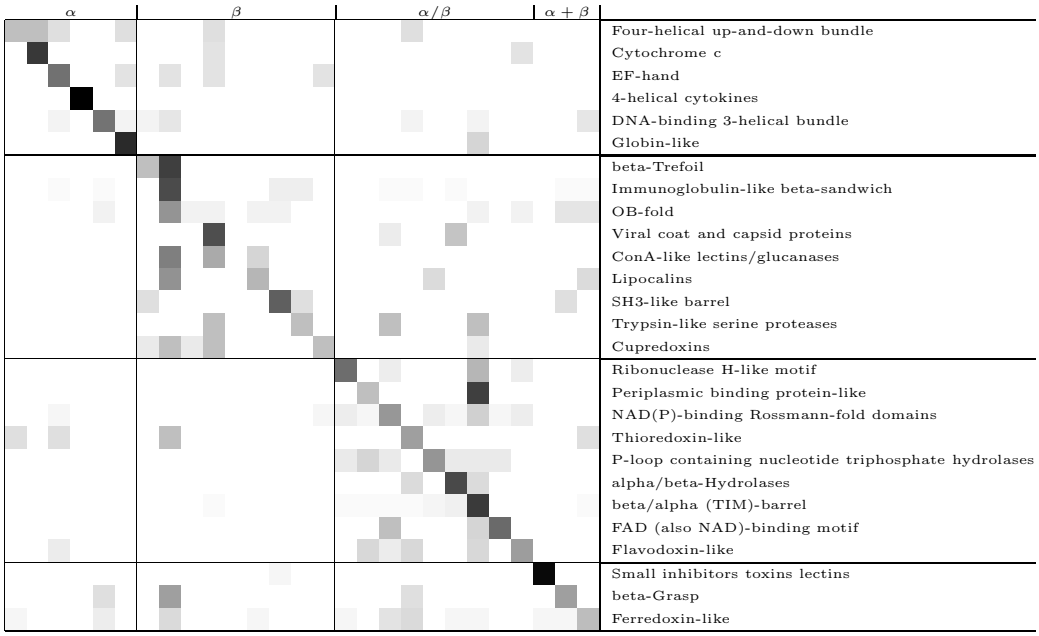


Figure 8.4: Visualization of the confusion matrix for the test of SVMs over the data set of Ding and Dubchak.
Confusion matrix of an all-versus-all classification using SVMs as base learners.

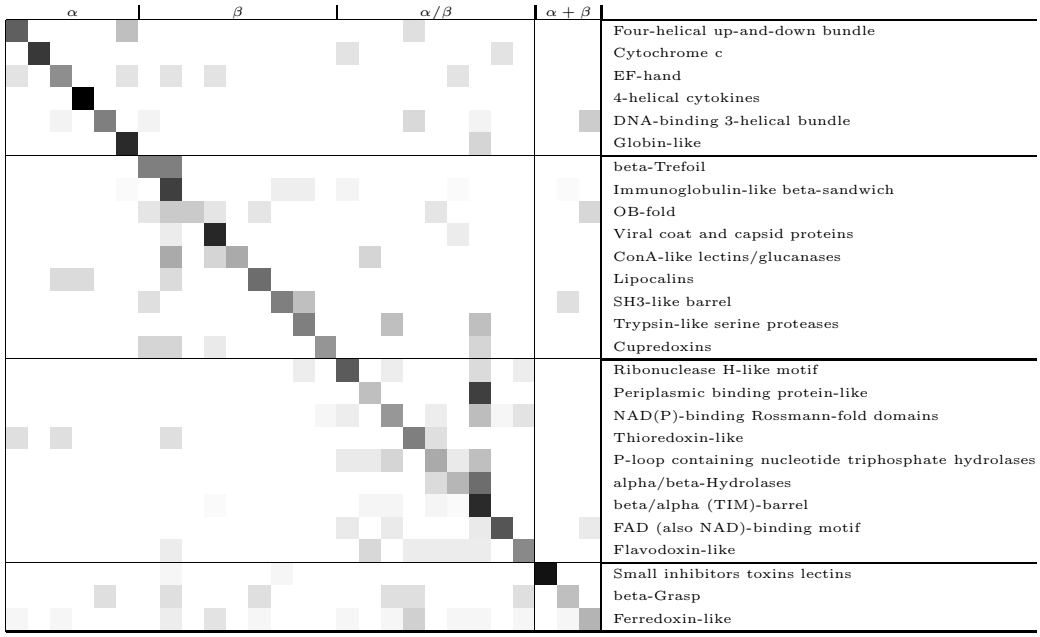


Figure 8.5: Visualization of the confusion matrix for the test of ENDs over the data set of Ding and Dubchak.
Confusion matrix of ENDs employing SVMs as base learners as presented in Table 8.4.

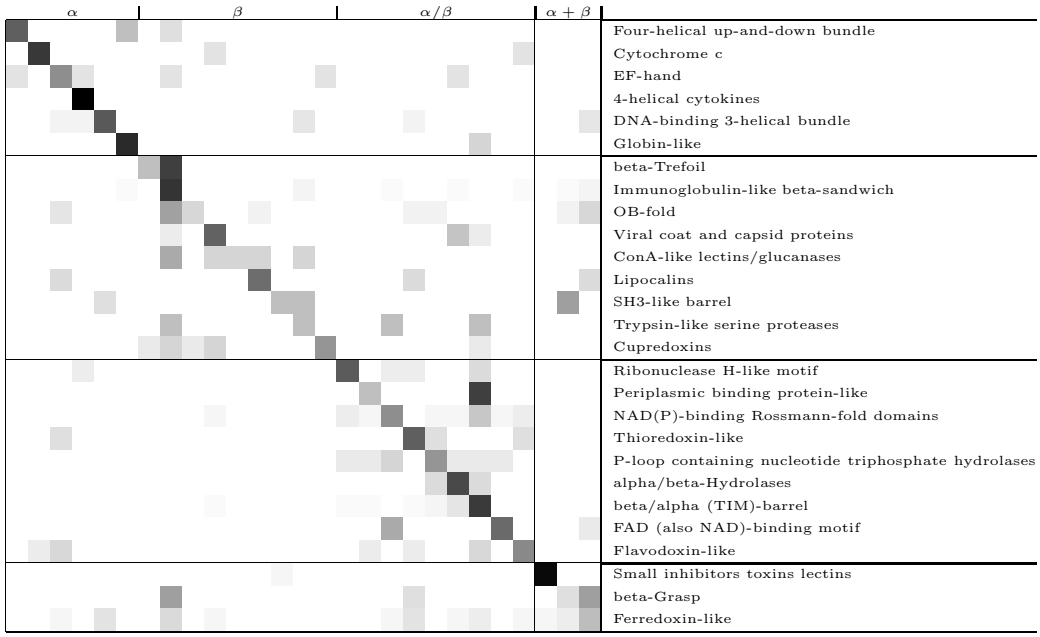


Figure 8.6: Visualization of the confusion matrix for the test of EHNDs over the data set of Ding and Dubchak (four class hierarchy).
Confusion matrix of EHNDs for a four class hierarchy employing SVMs as base learners as presented in Table 8.4.

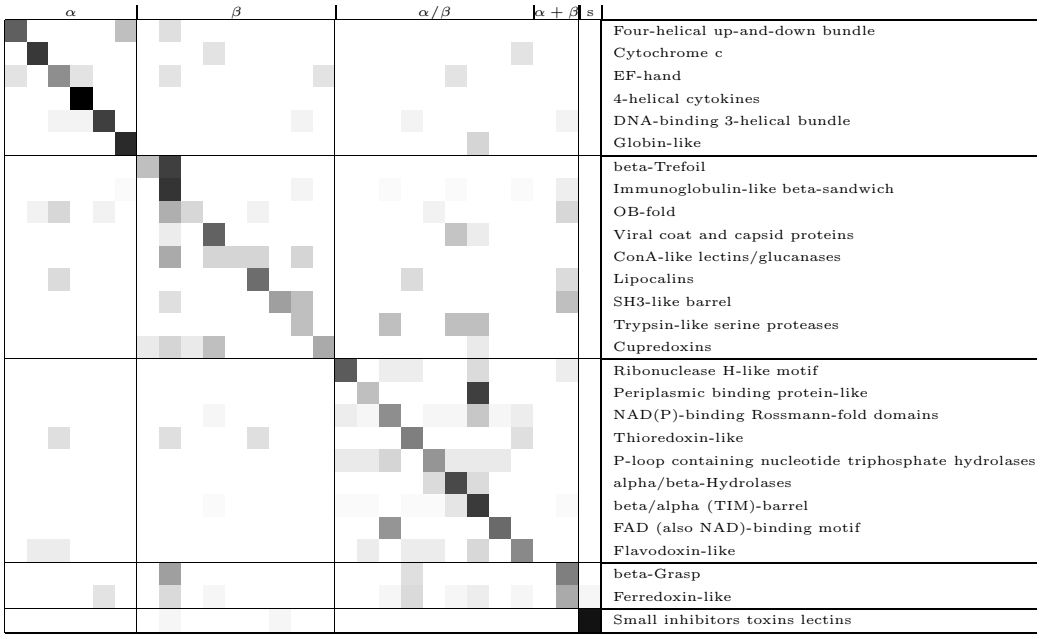


Figure 8.7: Visualization of the confusion matrix for the test of EHNDs over the data set of Ding and Dubchak (five class hierarchy).
Confusion matrix of EHNDs for a five class hierarchy employing SVMs as base learners as presented in Table 8.4.

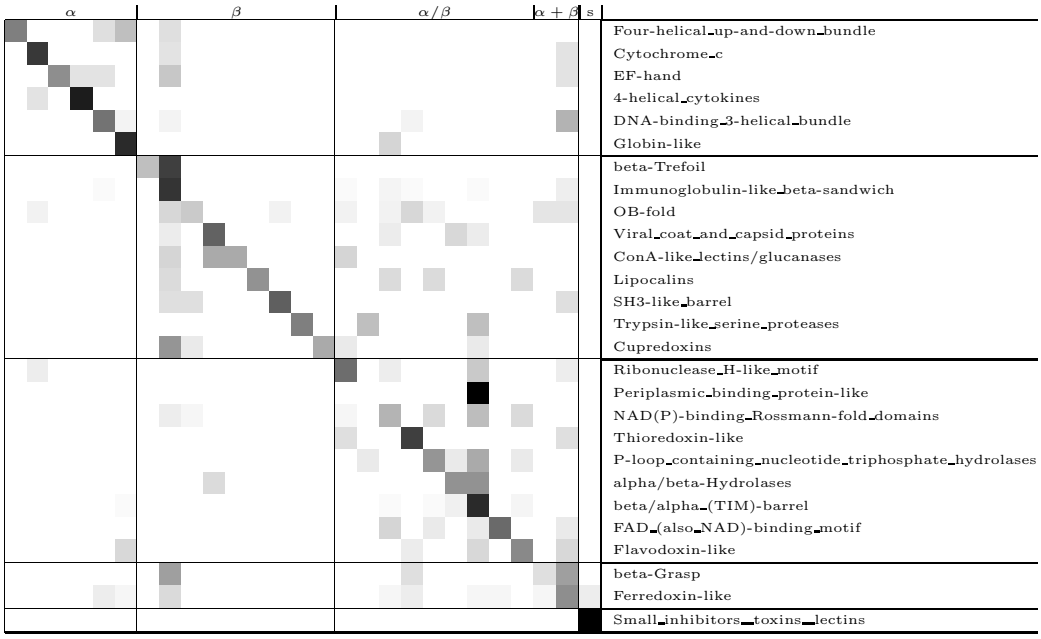


Figure 8.8: Visualization of the confusion matrix for the test of EHNDs over the data set of Ding and Dubchak (five class hierarchy).
Confusion matrix of EHNDs for a five class hierarchy employing Bagging (40 iterations) with PART as base learners as presented in Table 8.4.

Approach	Prediction accuracy (Q)	
PDB-BLAST	58.08	%
MANIFOLD	74.93	%
Alignment Combination	80	%
EHNDs		
(Bagged PART)	52.01	%
(SVM)	50.67	%

Table 8.6: Comparison of prediction accuracy for several alignment based approaches to fold recognition on the data of Ding and Dubchak as reduced by Bindewald et al.
The accuracy for other approaches is reported for PDB-BLAST and MANIFOLD according to [21], and for Alignment Combination according to [71]. These approaches used the training set and the test set (respectively only the test set for parameter free methods) as provided by [53] and reduced by Bindewald et al. [21].
We used the same data mapped to the feature space introduced by Ding and Dubchak for training ensembles of 20 HNDs (EHNDs) with Bagging (40 iterations) of PART (Bagged PART) and SVMs, respectively, as base learners. For SVMs a radial basis function was used as kernel with parameters $C = 1024$ and $\gamma = 0.044$.

Chinnasamy et al. [35] seem to discern five stuctural classes for the dataset. This is appropriate, since the fold 'Small inhibitors, toxins, lectins' belongs to the structural class 'Small proteins'. This is given in the classlist provided by Ding and Dubchak, but the fold is subsumed under the structural class $\alpha + \beta$ in [53]. Other hierarchy based approaches used a hierarchy based on four structural classes. We therefore tested EHNDs using both hierarchies. Using the more appropriate hierarchy of five structural classes improves the accuracy slightly.

We provide also further evaluation of the results for a classification by SVMs (pairwise classification), ENDs (SVM), and EHNDs (SVM) (Table 8.5, and Figures 8.4, 8.5, 8.6, 8.7, and 8.8).

8.3.2.2 Ding-Dubchak-Bindewald

For comparison with alignment methods that use a modified version of the sets of Ding and Dubchak we trained also Ensembles of 20 HNDs with the respective training set using Bagging (40 iterations) with PART and SVMs, respectively, as base learners. Parameters for SVMs (using a radial basis function as kernel) were fitted on the training set. The obtained results on the respective test set were worse than for the original sets. Obviously, for these data EHNDs can not keep pace with the best alignment based methods. An overview is presented in Table 8.6.

8.3.2.3 McGuffin-Bindewald

For the McGuffin known set as prepared by Bindewald et al. mapped to the feature space introduced by Ding and Dubchak we trained an Ensemble of 20 HNDs using Bagging (40 iterations) with PART as base learners. SVMs could not be used as base learners for these data for reasons explained above. By tenfold cross-validation we obtained an accuracy of 45.63%. This set is referred to as the 'most difficult set' by Gewehr et al. [71]. Indeed all methods reach accuracy values well below the respective values for the Ding-Dubchak-Bindewald set.

Using only cross-validation on the known set, EHNDs perform relatively well and increase the accuracy compared to its base learner. However, the value achieved by cross-validation on the known set only is not directly comparable to the results of the other methods. Bindewald et al. performed a rather complicated evaluation on these data. The complete set (known and unique) was used as templates, the

Approach	Prediction accuracy (Q)	
PDB-BLAST	13.25	%
GenTHREADER	14	%
MANIFOLD	33.96	%
Alignment Combination	42	%
J48 *leave-one-out* – known-vs-complete*	21.43	%
EHNDs		
leave-one-out – known-vs-complete* (J48)	32.94	%
leave-one-out – known-vs-complete* (Bagged PART)	42.06	%
Bagged PART *10-fold cross-validation – known*	43.25	%
ENDs *10-fold cross-validation – known* (Bagged PART)	46.03	%
EHNDs		
10-fold cross-validation – known (Bagged PART)	45.63	%
20-fold cross-validation – known (Bagged PART)	45.24	%
leave-one-out – known (Bagged PART)	44.44	%

Table 8.7: Comparison of prediction accuracy for several alignment based approaches to fold recognition on the data of McGuffin as adapted by Bindewald et al.

The accuracy for other approaches is reported for PDB-BLAST and MANIFOLD according to [21], for GenTHREADER and Alignment Combination according to [71]. These approaches used the known set and the unique set as provided by [108] and adapted by Bindewald et al. [21] in a specialized leave-one-out procedure as proposed by Bindewald et al.

The same procedure was performed with `J48` (`C4.5`) and EHNDs employing `J48` and Bagged PART, respectively, as base learners. We used the same data mapped into the feature space introduced by Ding and Dubchak.

Furthermore we performed tests by cross-validation on the known set only for ensembles of 20 HNDs using Bagging (40 iterations) with PART as base learners. The thereby reached accuracy is not directly comparable with the values of the alignment based methods but with the respective base learner and with ENDs employing the same base learner.

known set only as targets. Gewehr et al. followed the procedure of Bindewald et al.

This evaluation resembles a leave-one-out cross-validation on the complete set using only instances from the known set for testing. This testing procedure is highly unfavorable for a machine learning approach since the complete set consists of 317 classes represented by 538 instances. 268 classes (85%) are therefore represented by only one instance (consuming 50% of all instances for classes that allow no generalization at all). Only 15 classes are represented by more than 2 instances within the known set (27%), 20 within the complete set (6%). This allows practically no appropriate generalizations. Nevertheless we assessed the general ability of machine learning to cope with this procedure. Considering the large number of classes we decided to use a decision tree algorithm, C4.5 [125], implemented in `Weka` as `J48`. `J48` reached an accuracy level of 21%.

Compared to the accuracy achieved with `J48` EHNDs using `J48` as base learner improve considerably. Using Bagging (40 iterations) with PART as base learner EHNDs even reach the level of accuracy of the so far best known alignment based method.

We also trained an ensemble of NDs for comparison with EHND on equal terms, employing Bagging (40 Iterations) with PART as base learner and running a tenfold stratified cross-validation on the known set only. Here ENDs were slightly better than EHNDs.

An overview over the obtained results is provided by Table 8.7.

8.3.3 Discussion

The comparison with other machine learning approaches on the same data and feature space shows EHNDs to perform well. The obtained results are not worse than the results reported for the best of the other methods. On the other hand EHNDs do not improve very much in comparison to e.g. *all-versus-all* support vector machines. One may wonder whether the reachable accuracy on the given data set is limited by the chosen feature space.

This question becomes more important regarding the other experiments: The comparison with respect to alignment based methods does not evaluate the pure performance of EHNDs versus alignment based methods. The performance heavily depends on the feature space. Were in the first experiment all methods used the same feature space, here are the same proteins but different representations used. This is unavoidable in comparison to alignment based methods, that do not at all make directly use of extracted features.

The second experiment shows alignment based methods to perform much better than our method. This may occur due to considerable sequence similarity of several instances in the datasets. Also some classes are represented by very few members since some instances were excluded from the original set. Thus deriving a appropriate generalization becomes more difficult.

The third experiment is not easy to be interpreted. The testing procedure used in the compared methods is not at all suitable for any machine learning approach. But the data are prepared especially to evaluate alignment based methods and thus thought to be 'more difficult' [71]. Evaluating by cross-validation on the known set only a remarkable result can be achieved. These test are also showing EHNDs to improve well upon their respective base learner. But even following the more exhaustive specialized leave-one-out approach on the complete data set that is very unsuitable to machine learning methods, EHNDs reach the level of the so far best known alignment based method using Bagged PART as base learner.

The presented results raise another question. The difference in performance between ENDs and EHNDs *ceteris paribus* is ambiguous. On the same dataset for one base learner ENDs are better than EHNDs, but vice versa employing another

base learner and also alternating assessing different superclasses separately (Table 8.5). However, the differences are mostly not very pronounced. This leads us to the assumption that the defined hierarchy is at least not delusive. This assumption is also found to be supported by the confusion matrices showing less confusion between different superclasses than within the same superclass, for EHNDs as well as for other classifiers.

But evidently the hierarchy is not completely reflected in the chosen feature space or if so, it cannot be taken advantage of it. This is not surprising since the used hierarchies, SCOP and CATH, respectively, are constructed for structural features in three-dimensional fold while the feature space is based on primary and secondary structure only. Of course the three-dimensional structure is supposed to depend on the sequence. But the relation between sequence and structure may not be completely detectable in the features that represent the sequence to the learning algorithm. Furthermore, it should not be neglected that the used hierarchies, SCOP and CATH, and also another well known hierarchy of protein folds, FSSP, differ in principle, in architecture, and of course also in actual assignments of folds – mainly due to differences in opinion between the curators of these databases, as other investigations have already pointed out [107], [79].

Based on our results we would therefore like to encourage further research concerned with the application of machine learning algorithms to the problem of protein fold classification to make more efforts in developing feature spaces that describe proteins well than in developing new machine learning algorithms that again hinge on the same somewhat insufficient feature space. Supported by theoretical considerations (chapter 6) as well as by experiments on data with a hierarchical structure being either completely or not at all reflected in the feature space (chapter 7) we believe that the tradeoff in accuracy between ENDs and EHNDs would be a helpful instrument to evaluate the quality of a feature space with respect to a given hierarchy.

8.4 Conclusion

We applied EHNDs to protein fold classification which is a task often coped with as 'fold recognition' as well by pure machine learning approaches as by alignment based approaches. The line between those approaches is not always clear. We call pure 'machine learning approaches' algorithms that hinge on a feature space representation of proteins.

Compared to the best known other machine learning approaches EHNDs performed not worse than these. The comparison to alignment based methods is ambiguous and strongly depends on the data set. Some data sets exhibiting a relative high sequence similarity between some instances will usually be more easy for alignment methods. Furthermore the comparison of performance of machine learning methods versus alignment based approaches does not only assess the quality of a pure machine learning algorithm but also the quality of the selected feature space. Nevertheless, the accuracy of EHNDs on data considered to be 'more difficult' for alignment based methods improves to the level of the currently best known alignment based method.

Interesting is the comparison between EHNDs and ENDs. On the considered data there is not only not much of a gap between them but they are also alternating in improving over one another. This might indicate that the used hierarchy is not misleading but also not quite evident in the selected feature space.

So far introduced feature spaces for proteins all hinge on sequences and secondary structure elements. Since the structure of a protein is commonly supposed to depend completely on its sequence the effort to find feature spaces that repre-

sent as much as possible information concerning the sequence is understandable, although this might not surpass alignment methods that naturally use all the information a sequence can provide – or even more by combination with profiles or secondary structure alignment. But fold recognition is finally concerned with three-dimensional structures. So we would like to point out as a possibly promising direction to find valuable feature representations for this task an investigation whether there are well discriminating features in known structures that are highly correlated with well extractable properties of sequences.

We believe that our method taken together with ENDs is of special interest in evaluating the quality of a feature space with respect to a given hierarchy. Since the widely used hierarchical structural classifications of proteins, SCOP, CATH, and also FSSP, exhibiting a less pronounced hierarchy, differ quite considerably the suggestion of McGuffin et al. [107] seems reasonable, to benchmark automatic fold recognition methods on a consensus of structural classifications only.

Part IV

Conclusion

Chapter 9

Conclusion

9.1 Summary

Many classifiers work best on dichotomous data. At the same time many real world applications require to consider many classes. Therefore several possibilities to decompose polytomies to dichotomies are established in machine learning. Another task in machine learning research is to build ensembles of classifiers that improve over their single members. Some ensemble techniques are also able to decompose polytomies to dichotomies. We reviewed especially ensembles of nested dichotomies (ENDs) that have been recently shown to work well with respect to both of these [66]. A third task is to incorporate domain knowledge in learning algorithms in order to improve accuracy or efficiency.

Our method tackles those tasks in combination: EHNDs are ensembles built of nested dichotomies that are reflecting a given hierarchy of classes. Compared to ENDs the hierarchy is incorporated to improve accuracy. The success, however, depends mainly on two conditions:

1. The given hierarchy must reflect the relations between classes according to an arbitrary similarity measure.

2. The respective similarity of classes must be detectable in the feature space representing instances of classes.

Both conditions being fulfilled, EHNDs were shown to improve over their base learner and over ENDs not considering the hierarchy. The data used for this evaluation were created synthetically and exhibit a quite pronounced hierarchy. Constraining EHNDs by an improper hierarchy, however, leads to a dramatic decrease of accuracy. This observation, confirming theoretical considerations, motivates the idea to use the tradeoff in accuracy between ENDs and EHNDs for evaluation of different feature spaces with respect to a given hierarchy.

EHNDs were applied to fold recognition using features based on primary and secondary structure. The established hierarchies in structural classification of proteins (SCOP and CATH) might be quite less pronounced than the hierarchy in synthetic data. Furthermore the respective hierarchies might not be completely reflected in the feature space. On the other hand neither the hierarchies nor the feature spaces were delusive since ENDs and EHNDs were close to one another in terms of accuracy. EHND performed at the level of best performing machine learning approaches we are aware of. For difficult data EHNDs achieved also an accuracy level equal to the best performing alignment based approach known to us.

9.2 Outlook

9.2.1 Biological Applications

9.2.1.1 Fold Recognition

EHNDs were shown to work well on fold recognition compared to other machine learning algorithms. They were also more accurate than some alignment based methods on certain data sets and reached the level of accuracy of the best performing alignment based method on difficult data. Perhaps EHNDs might therefore be worthwhile to be considered to be used alongside alignment methods. For fold recognition often not only the prediction with highest confidence is of interest but the predictions at rank one to five or even one to ten. In further investigations we would like to assess the performance of EHNDs under these conditions.

9.2.1.2 Using Structural Features

We have considered several different feature representations of proteins. They have all in common to hinge on sequences and secondary structure elements. It is a reasonable goal to find feature spaces that represent as much information as possible concerning the underlying sequence since it is commonly accepted that the structure of a protein is completely determined by the protein's primary structure. On the other hand one might not surpass alignment methods based on these features since alignment methods naturally use all information that can be provided by a sequence. In combination with profiles or secondary structure alignments the latter methods can make use of even more information than the information provided by a single sequence. On the other hand fold recognition is finally concerned with three-dimensional structures. Thus a rewarding idea to find valuable feature representations for fold recognition might possibly be to search for well discriminating features in known structures that are highly correlated with well extractable properties of sequences.

9.2.1.3 Automated Classification

Since the hierarchy of structural classifications like SCOP and CATH is defined with respect to three-dimensional structure using EHNDs might be more prolific in automated classification, that is based on features derived from a known structure instead of sequence based features only. Unfortunately deriving such features is not trivial at all. But given a good structure-based feature representation we consider EHNDs to be a very suitable method for hierarchical classification of protein structure that is easily adaptive to either SCOP or CATH or any other hierarchical database by simply constraining the learner with the proper hierarchy of classes.

9.2.1.4 Evaluating Feature Spaces

We have already motivated to search for new and better feature spaces for proteins. Based on theoretical considerations as well as on experimentally derived results we suggest to use the tradeoff in accuracy between ENDs and EHNDs for evaluation of different feature spaces reflecting the same hierarchy. This is, however, a preliminary idea that demands further investigation in order to be successfully employed.

9.2.2 Generalizations

9.2.2.1 Hierarchy-Dependent Cost-Matrices

Using cost-matrices dependent on a class-hierarchy could also be a method to introduce domain-knowledge concerning a hierarchical structure in data *implicitly*, without building a classifier *explicitly* hierarchically. Thus any classifier could use hierarchical information on data not being hierarchically structured itself. Further studies on improvement of classification-accuracy for different hierarchically structured data would be desirable.

9.2.2.2 Hierarchical Structure as Meta-Reduction

Whilst the method we proposed in this thesis is very suggesting in combination with Ensembles of Nested Dichotomies it is not at all restricted to them. It could easily be generalized to reduce polytomous classification problems to several oligotomous ones, which could improve performance for any classification method with respect to a hierarchical structure in data. Consider, e.g., the *all-versus-all* coupling of support vector machines, which is of quadratic complexity over the number of classes. This is a very frequently encountered application of support vector machines. Using hierarchical structure could reduce the overall required number of support vector machines considerably. In case of the simple 16-class problem presented in Figure 6.1, one would require $5 \times \binom{4}{2}$ instead of $\binom{16}{2}$ base-classifiers. Besides, the prediction accuracy should also improve, since irrelevant classes are ignored while building some classifiers. But this topic also demands further investigation.

One could also expect to improve accuracy as well as speed for the third binarization considered in chapter 2, *one-versus-others*: All classes are to be considered by all base learners only for the top-level superclasses while for subclasses only the related classes and the respective instances are to be considered.

Let us note that the implementation we provided for our method can easily be adapted to work with arbitrary classifiers instead of using NDs as level-wise classifier. Furthermore it shows potential that was not taken advantage of in our investigation although we performed preliminary experiments with respect to this property: The separation of classes might be performed best on different levels by different base learners, on different subsets of feature space, or by same base learners but using different parameters. Exploring these possibilities, perhaps including different combinations of feature spaces, would require to conduct many additional time consuming experiments. Perhaps some future research can carry on these ideas.

9.2.2.3 Hierarchical ECOC-Schemes

We referred ECOC as means of generalization of binarization techniques for *all-versus-all*, *one-versus-others*, and some less frequent used techniques in chapter 3. Also NDs are representable in the namely framework (chapter 4) even though it will demand some further efforts to define the decoding procedure properly in this case. Defining constraints concerning hierarchically structured classes that are applicable to build proper ECOC schemes seems a promising idea since that would also provide a general framework to apply hierarchical classification to arbitrary decomposition procedures. We are aware of a possible drawback in setting to much constraints for building ECOC schemes since this could compromise the two main requirements for well working ECOC: row separation and column separation. However, the topic might be interesting and well worth further consideration.

9.2.2.4 Allowing Multiple Classes for One Instance

In chapter 8, section 8.1.2, we considered other fields of biological tasks of hierarchical classification, where applying machine learning approaches might be rewarding. However, our method would require some adaptations to be applied to those tasks. We considered the classification of protein functions. Since a single protein can have several functions this task implies learning a function that maps an instance to a subset of the set of classes. Functional classification of proteins exhibits a pronounced hierarchical structure of classes. Thus it might be worthwhile to extend EHNDs for covering classification tasks concerning classes that are not mutually exclusive.

9.2.2.5 Generalizing Class-Relationships

The subsets considered for the definition of hierarchies of classes (Definition 6.3) were disjoint. It may be possible that the similarity of classes differs for different subspaces. Consider that the members of a subset of some classes are more similar to each other than they are to other classes not contained in the respective subset, but only with respect to certain features, and that they would cluster to other subsets with respect to other features. Thus the possible subsets defining related classes would not be disjoint any more and defining a respective tree, that is a hierarchy of classes, would not be possible.

Also in chapter 8, section 8.1.2 we encountered the well established hierarchy of biological entities, Gene Ontology. As we already have noted there, Gene Ontology resembles not a pure hierarchy but a graph that could be considered to be a directed acyclic graph (DAG). A possible and interesting extension of EHNDs therefore could be to use DAGs instead of hierarchies. However, the impact on the method would not be a small one since the decoding procedure of NDs ought to be adapted because the class probability estimates are based on the assumption of independence of dichotomies.

Perhaps an alternative would be to employ an ensemble of EHNDs where the ensemble members, single ensembles of HNDs, are built with respect to different hierarchies that taken together define the respective DAG.

9.3 Conclusion

The method EHNDs proposed in this thesis may be considered for further studies in possible applications using proper feature spaces or to assess feature spaces with respect to a given hierarchy. Several directions of generalizations were suggested with respect to theoretical or practical enhancements. We therefore believe to have shown that EHNDs may remain an interesting topic of scientific research in the future and hope that further developments can be based on the considerations presented in this thesis.

Part V

Appendix

Appendix A

Data and Implementations

The appended compact disc provides the data and software used for evaluation of our method. EHNDs were implemented in `Java` within the `Weka`-framework. Further information is provided within the documentation to the software.

Bibliography

[1] C. C. Aggarwal and Cecilia M. Procopiuc. Fast algorithms for projected clustering. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA*, 1999.

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional space. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'00), Dallas, TX*, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'98), Seattle, WA*, 1998.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94), Minneapolis, MN*, pages 94–105, 1994.

[5] Nickolai N. Alexandrov, Ruth Nussinov, and Ralf Zimmer. Fast protein fold recognition via sequence to structure alignment and contact capacity potentials. In L. Hunter and T. E. Klein, editors, *Pacic Symposium on Biocomputing*, pages 53–72, 1995.

[6] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[7] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[8] S. F. Altschul, T. L. Madden, A. A. Schaeffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acid Research*, 25:3389–3402, 1997.

[9] Claus A. F. Andersen and Burkhard Rost. Secondary structure assignment. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 17, pages 341–363. John Wiley & Sons, Inc., 2003.

[10] Antonina Andreeva, Dave Howorth, Steven E. Brenner, Tim J. P. Hubbard, Cyrus Chothia, and Alexey G. Murzin. SCOP database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Research*, 32(Database issue):D226–D229, 2004.

[11] Christian B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[12] Aristotle. De historia animalium.

[13] Pierre Baldi and Søren Brunak. *Bioinformatics. The Machine Learning Approach*. Adaptive Computation and Machine Learning. MIT Press, second edition, 2001.

[14] Pierre Baldi, Søren Brunak, Yves Chauvin, Claus A. F. Andersen, and Henrik Nielsen. Assessing the accuracy of prediction algorithms for classification: An overview. *Bioinformatics*, 16(5):412–424, 2000.

[15] Gail J. Bartlett, Annabel E. Todd, and Janet M. Thornton. Inferring protein function from structure. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 19, pages 387–407. John Wiley & Sons, Inc., 2003.

[16] A. Bateman, E. Birney, R. Durbin, S. R. Eddy, K. L. Howe, and E. L. Sonnhammer. The Pfam protein families database. *Nucleic Acids Research*, 28(1):263–266, 2000.

[17] Christian Baumgartner, Karin Kailing, Hans-Peter Kriegel, Peer Kröger, and Claudia Plant. Subspace selection for clustering high-dimensional data. In *Proc. 4th IEEE International Conference on Data Mining (ICDM'04), Brighton, UK*, pages 11–18, 2004.

[18] Richard Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[19] Asa Ben-Hur and Douglas Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 19(Suppl. 1):i26–i33, 2003.

[20] F. C. Bernstein, T. F. Koetzle, G. J. Williams, E. E. Meyer, M. D. Brice, J. R. Rogers, O. Kennard, T. Shimanouchi, and M. Tasumi. Protein Data Bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

[21] Eckart Bindewald, Alessandro Cestaro, Jürgen Hesser, Matthias Heiler, and Silvio C. E. Tosatto. MANIFOLD: protein fold recognition based on secondary structure, sequence similarity and enzyme classification. *Protein Engineering*, 16(11):785–789, 2003.

[22] Fabian Birzele. Data mining for protein secondary structure prediction. Master's thesis, TU Munich, Institute for Computer Science 12, 2005.

[23] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, 1992.

[24] Philip E. Bourne. CASP and CAFASP experiments and their findings. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 24, pages 501–507. John Wiley & Sons, Inc., 2003.

[25] Philip E. Bourne and Helge Weissig, editors. *Structural Bioinformatics*. John Wiley & Sons, Inc., 2003.

[26] Carl-Ivar Branden and John Tooze. *Introduction to Protein Structure*. Garland Publishing, second edition, 1999.

[27] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[28] Steven E. Brenner, Cyrus Chothia, and Tim J. P. Hubbard. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proc. Natl. Acad. Sci. USA*, 95:6073–6078, 1998.

[29] Steven E. Brenner, Patrice Koehl, and Michael Levitt. The ASTRAL compendium for protein structure and sequence analysis. *Nucleic Acids Research*, 28(1):254–256, 2000.

[30] Wei-Shu Bu, Zhi-Ping Feng, Ziding Zhang, and Chun-Ting Zhang. Prediction of protein (domain) structural classes based on amino-acid index. *Eur. J. Biochem.*, 266:1043–1049, 1999.

[31] John-Marc Chandonia, Gary Hon, Nigel S. Walker, Loredana Lo Conte, Patrice Koehl, Michael Levitt, and Steven Brenner. The ASTRAL compendium in 2004. *Nucleic Acids Research*, 32(Database issue):D189–D192, 2004.

[32] John-Marc Chandonia, Nigel S. Walker, Loredana Lo Conte, Patrice Koehl, Michael Levitt, and Steven E. Brenner. ASTRAL compendium enhancements. *Nucleic Acids Research*, 30(1):260–263, 2002.

[33] M. Charton and B. I. Charton. The structural dependence of amino acid hydrophobicity parameters. *J. Theor. Biol.*, 99:629–644, 1982.

[34] C.-H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery in Databases (SIGKDD'99), San Diego, FL*, 1999.

[35] A. Chinnasamy, W. K. Sung, and A. Mittal. Protein structure and fold prediction using tree-augmented naïve bayesian classifier. In R. Altman, A. Keith, L. Hunter, T. Jung, and T. Klein, editors, *Pacific Symposium on Biocomputing 2003*, volume 9, pages 387–398, 2004.

[36] Dylan Chivian, Timothy Robertson, Richard Bonneau, and David Baker. Ab initio methods. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 27, pages 547–557. John Wiley & Sons, Inc., 2003.

[37] Cyrus Chothia. One thousand families for the molecular biologist. *Nature*, 357:543–544, 1992.

[38] Cyrus Chothia and A. V: Finkelstein. The classification and origins of protein folding patterns. *Annu. Rev. Biochem.*, 59:1007–1039, 1990.

[39] Nello Christianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[40] I-Fang Chung, Chuen-Der Huang, Ya-Hsin Shen, and Chin-Teng Lin. Recognition of structure classification of protein folding by NN and SVM hierarchical learning architecture. In Okyay Kaynak, Ethem Alpaydin, Erkki Oja, and Lei Xu, editors, *Artificial Neural Networks and Neural Information Processing – ICANN/ICONIP 2003*, volume 2714 of *Lecture Notes in Computer Science (LNCS)*, pages 1159–1167. Joint International Conference ICANN/ICONIP 2003, Istanbul, Turkey, June 26-29, 2003, Springer-Verlag, 2003.

[41] J. L. Cornette, K. B. Cease, H. Margali, J. L. Spouge, J. A. Berzofsky, and C. DeLisi. Hydrophobicity scales and computational techniques for detecting amphipathic structures in proteins. *J. Mol. Biol.*, 195:659–685, 1987.

[42] Charles R. Darwin. *On the Origin of Species*. John Murray, 1859.

[43] Mukund Deshpade and George Karypis. Evaluation of techniques for classifying biological sequences. In *Proc. of Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'02)*, pages 417–431, 2002.

[44] R. E. Dickerson and I. Geis. *Hemoglobin: Structure, Function, Evolution, and Pathology.* Benjamin-Cummings, 1983.

[45] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. Technical report, Oregon State University, Department of Computer Science, Corvallis, OR 97331, 1997.

[46] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2000.

[47] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[48] Thomas G. Dietterich. An overview of MAXQ hierarchical reinforcement learning. In B. Y. Choueiry and T. Walsh, editors, *Proceedings of the Symposium on Abstraction, Reformulation and Approximation SARA 2000*, Lecture Notes in Artificial Intelligence, pages 26–44. Springer Verlag, 2000.

[49] Thomas G. Dietterich. Ensemble learning. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 405–408. MIT Press, second edition, 2003.

[50] Thomas G. Dietterich and Ghulum Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of AAAI-91*, pages 572–577. AAAI Press / MIT Press, 1991.

[51] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[52] Thomas G. Dietterich and N. Flann. Explanation-based learning and reinforcement laerning: A unified view. In *Proceedings of Machine Learning Conference*, 1995.

[53] Chris H. Q. Ding and Inna Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

[54] Theodosius Grigorevich Dobzhansky. Nothing in biology makes sense except in the light of evolution. *The American Biology Teacher*, 35(3):125–129, 1973.

[55] Andreas Dress. Phylogenetic combinatorics. Lecture in the Munich Bioinformatics Colloquium, November 2004.

[56] Inna Dubchak, Ilya Muchnik, Stephen R. Holbrook, and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proc. Natl. Acad. Sci. USA*, 92:8700–8704, September 1995.

[57] Inna Dubchak, Ilya Muchnik, Christopher Mayor, Igor Dralyuk, and Sung-Hou Kim. Recognition of a protein fold in the context of the SCOP classification. *PROTEINS: Structure, Function, and Genetics*, 35:401–407, 1999.

[58] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM Press, 2000.

[59] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis. Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, seventh edition, 2002.

[60] Gerald D. Fasman, editor. *Prediction of Protein Structure and the Principles of Protein Conformation*. Plenum Press, 1989.

[61] J. L. Fauchere, M. Charton, L. B. Kier, A. Verloop, and V. Pliska. Amino acid side chain parameters for correlation studies in biology and pharmacology. *Int. J. Pept. Protein Res.*, 3:269–278, 1988.

[62] Daniel Fischer and David Eisenberg. Protein fold recognition using sequence-derived predictions. *Protein Science*, 5:947–955, 1996.

[63] R. A. Fisher. The use of multiple measurements on taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[64] E. W. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classification (abstract). *Biometrics*, 21:768–769, 1965.

[65] John Fox. *Applied Regression Analysis, Linear Models, and Related Methods*. Sage, 1997.

[66] Eibe Frank and Stefan Kramer. Ensembles of nested dichotomies for multi-class problems. In *Twenty-first international conference on Machine learning*. ACM Press, 2004.

[67] Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In J. Shavlik, editor, *Machine Learning: Proceedings of the Fifteenth International Conference*. Morgan Kaufmann, 1998.

[68] Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics and Stanford Linear Accelerator Center, Stanford University, 1996.

[69] Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[70] Jan Gewehr. personal communication, 2005.

[71] Jan E. Gewehr, Niklas von Öhsen, and Ralf Zimmer. Combining secondary structure element alignment and profile-profile alignment for fold recognition. In R. Giegerich and Jens Stoye, editors, *German Conference on Bioinformatics (GCB) 2004, GI*, volume P-53 of *Lecture Notes in Informatics*, pages 141–148, 2004. Revised version: `http://www.bio.ifi.lmu.de/mitarbeiter/gewehr/GCB04final.pdf`.

[72] Adam Godzik. Fold recognition methods. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 26, pages 525–546. John Wiley & Sons, Inc., 2003.

[73] S. Goil, H. S. Nagesh, and A. Choudhary. MAFIA: Efficiant and scalable subspace clustering for very large data sets. Tech. Report No. CPDC-TR-9906-010, Center for Parallel and Distributed Computing, Dept. of Electrical and Computer Engineering, Northwestern University, 1999.

[74] Stephen J. Gould. The first unmasking of nature. *Natural History*, 14(4):16–21, 1993.

[75] Sridhar Govindarajan and Richard A. Goldstein. Why are some protein structures so common? *Proc. Natl. Acad. Sci. USA*, 93:3341–3345, April 1996.

[76] Sridhar Govindarajan, R. Recabarren, and Richard A. Goldstein. Estimating the total number of protein folds. *Proteins*, 35:408–414, 1999.

[77] R. Grantham. Amino acid difference formula to help explain protein evolution. *Science*, 185:862–864, 1974.

[78] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology.* Cambidge University Press, 1997.

[79] Caroline Hadley and David T. Jones. A systematic comparison of protein structure classifications: SCOP, CATH and FSSP. *Structure*, 7:1099–1112, September 1999.

[80] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining.* Adaptive Computation and Machine Learning. MIT Press, 2001.

[81] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.

[82] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* Springer-Verlag, 2001.

[83] Liisa Holm and Chris Sander. The FSSP database of structurally aligned protein fold families. *Nucleic Acids Res.*, 22:3600–3609, 1994.

[84] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan, 2001.

[85] Chuen-Der Huang, I-Fang Chung, Nikhil Ranjan Pal, and Chin-Teng Lin. Machine learning for multi-class protein fold classification based on neural networks with feature gating. In Okyay Kaynak, Ethem Alpaydin, Erkki Oja, and Lei Xu, editors, *Artificial Neural Networks and Neural Information Processing – ICANN/ICONIP 2003*, volume 2714 of *Lecture Notes in Computer Science (LNCS)*, pages 1168–1175. Joint International Conference ICANN/ICONIP 2003, Istanbul, Turkey, June 26-29, 2003, Springer-Verlag, 2003.

[86] David T. Jones. GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. *J. Mol. Biol.*, 287:797–815, 1999.

[87] David T. Jones, W. R. Taylor, and Janet M. Thornton. A new approach to protein fold recognition. *Nature*, 358:86–89, 1992.

[88] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[89] Karin Kailing, Hans-Peter Kriegel, Peer Kröger, and Stefanie Wanka. Ranking interesting subspaces for clustering high dimensional data. In *Proc. 7th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'03), Cavtat-Dubrovnic, Croatia*, volume 2838 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 241–252, 2003.

[90] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001.

[91] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In Françoise Fogelman Soulié and Jeanny Hérault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, volume F68 of *NATO ASI Series*, pages 41–50. Springer-Verlag, 1990.

[92] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *The XII International Conference on Machine Learning, San Francisco, CA*, pages 313–321. Morgan Kaufmann, 1995.

[93] Elmar Krieger, Sander B. Nabuurs, and Gert Vriend. Homology modeling. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 25, pages 509–523. John Wiley & Sons, Inc., 2003.

[94] Peer Kröger, Hans-Peter Kriegel, and Karin Kailing. Density-connected subspace clustering for high-dimensional data. In *Proc. SIAM Int. Conf. on Data Mining (SDM'04), Lake Buena Vista, FL*, pages 246–257, 2004.

[95] Daniel Kudenko and Haym Hirsh. Feature generation for sequence categorization. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, pages 733–738, 1998.

[96] Shailesh Kumar, Joydeep Ghosh, and Melba Crawford. A hierarchical multiclassifier system for hyperspectral data analysis. In *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000*, volume 1857 of *Lecture Notes in Computer Science*, pages 270–279. Springer-Verlag, 2000.

[97] S. le Cessie and J. C. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[98] Arthur M. Lesk. *Introduction to Protein Architecture. The Structural Biology of Proteins.* Oxford University Press, 2001.

[99] Michael Levitt and Cyrus Chothia. Structural patterns in globular proteins. *Nature*, 261:552–558, 1976.

[100] Chih-Jen Lin and Chih-Wei Hsu. A comparison of methods for multi-class support vector machines. In *IEEE Transactions on Neural Networks*, volume 13, pages 415–425, 2002.

[101] Loredana Lo Conte, B. Ailey, Tim J. P. Hubbard, Steven E. Brenner, Alexey G. Murzin, and Cyrus Chothia. SCOP: a structural classification of proteins database. *Nucleic Acids Research*, 28:257–259, 2000.

[102] Loredana Lo Conte, Steven E. Brenner, Tim J. P. Hubbard, Cyrus Chothia, and Alexey G. Murzin. SCOP database in 2002: refinements accomodate structural genomics. *Nucleic Acids Research*, 30(1):264–267, 2002.

[103] Harvey Lodish, Arnold Berk, Lawrence Zipursky, Paul Matsudaira, David Baltimore, and James Darnell. *Molecular Cell Biology*. Freeman, fourth edition, 2000.

[104] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Prob.*, volume 1, pages 281–297, 1967.

[105] Eddy Mayoraz and Miguel Moreira. On the decomposition of polychotomies into dichotomies. In Douglas H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 219–226. Morgan Kaufmann Publishers Inc., 1997.

[106] Ernst Mayr. *The Growth of Biological Thought*. Belknap Press, 1982.

[107] Liam J. McGuffin, Kevin Bryson, and David T. Jones. What are the baselines for protein fold recognition? *Bioinformatics*, 17(1):63–72, 2001.

[108] Liam J. McGuffin and David T. Jones. Targeting novel folds for structural genomics. *Proteins: Structure, Function, and Genetics*, 48:44–52, 2002.

[109] Tom M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, Department of Computer Science, 1980.

[110] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[111] Alexey G. Murzin, Steven E. Brenner, Tim Hubbard, and Cyrus Chothia. SCOP a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

[112] Craig G. Nevill-Manning, Thomas D. Wu, and Douglas Brutlag. Highly specific protein sequence motifs for genome analysis. *Proc. Natl. Acad. Sci. USA*, 95(11):5865–5871, 1998.

[113] Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB). *Enzyme Nomenclature*. Academic Press, San Diego, 1992.

[114] M. Oobatake and T. Ooi. An analysis of non-bonded energy of proteins. *J. Theor. Biol.*, 67:567–584, 1977.

[115] Christine A. Orengo, A.D. Michie, S. Jones, David T. Jones, M. B. Swindells, and Janet M. Thornton. CATH – a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

[116] Christine A. Orengo, Frances M. G. Pearl, and Janet M. Thornton. The CATH domain structure database. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 13, pages 249–271. John Wiley & Sons, Inc., 2003.

[117] C. Ouzounis, C. Sander, M. Scharf, and R. Schenider. Prediction of protein structure by evaluation of sequence-structure fitness: aligning sequences to contact profiles derived from 3D structures. *J. Mol. Biol.*, 232:805–825, 1993.

[118] Philipp Pagel, Hans-Werner Mewes, and Dmitrij Frishman. Conservation of protein-protein interactions – lessons from ascomycota. *Trends in Genetics*, 20(2):72–76, 2004.

[119] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, Tim J. P. Hubbard, and Cyrus Chothia. Sequence cmparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J. Mol. Biol.*, 284:1201–1210, 1998.

[120] Bambang Parmanto, Paul W. Munro, and Howard R. Doyle. Improving committee diagnosis with resampling techniques. In David S. Touretzky, Michael Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, November 27-30, 1995*. MIT Press, 1996.

[121] Frances M. G. Pearl, D. Lee, J. E. Bray, I. Sillitoe, Annabel E. Todd, Janet M. Thornton, and Christine A. Orengo. Assigning genomic sequences to CATH. *Nucleic Acids Res.*, 28(1):277–282, 2000.

[122] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1998.

[123] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, April 1998.

[124] John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553. MIT Press, 2000.

[125] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[126] A. Raval, Z. Ghahramani, and D. L. Wild. A Bayesian network model for protein fold and remote homologue recognition. *Bioinformatics*, 18(6):788–801, 2002.

[127] Boojala V. B. Reddy and Philip E. Bourne. Protein structure evolution and the SCOP database. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 12, pages 239–248. John Wiley & Sons, Inc., 2003.

[128] Ryan Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2002.

[129] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[130] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[131] B. Rost. Topits: Threading one-dimensional predictions into three-dimensional structures. In *Proc. Conf. Intelligent Systems in Molecular Biology*, pages 314–321. ISMB-95, 1995.

[132] Ulrich Rückert and Stefan Kramer. Towards tight bounds for rule learning. In Russel Greiner and Dale Schuurmans, editors, *Proc. of the 21st International Conference on Machine Learning (ACM, 2004)*, 2004.

[133] L. Rychlewski, L. Jaroszewski, W. Li, and A. Godzik. Comparison of sequence profiles: strategies for structural predictions using sequence information. *Protein Science*, 9:232–241, 2000.

[134] Robert E. Schapire, Marie Rochery, Mazin Rahim, and Narendra Gupta. Incorporating prior knowledge into boosting. In *Machine Learning: Proceedings of the Nineteenth International Conference*, 2002.

[135] Eric D. Scheef and J. Lynn Fink. Fundamentals of protein structure. In Philip E. Bourne and Helge Weissig, editors, *Structural Bioinformatics*, chapter 2, pages 15–39. John Wiley & Sons, Inc., 2003.

[136] Matthias Schubert, Alexey Pryakhin, Peer Kröger, and Hans-Peter Kriegel. Using support vector machines for classifying large sets of multi-represented objects. In *SIAM International Conference on Data Mining (SIAM DM'04)*, 2004.

[137] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Syst.*, 1:145–168, 1987.

[138] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[139] Daniel Shapiro, Pat Langley, and Ross Shachter. Using background knowledge to speed reinforcement learning in physical agents. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 254–261. ACM Press, 2001.

[140] Q. Su, S. Saxonov, and Douglas Brutlag. eBLOCKS: an automated database of protein conserved regions maximizing sensitivity and specificity. `http://motif.stanford.edu/eblocks`.

[141] Prasad Tadepalli and Thomas G. Dietterich. Hierarchical explanation-based reinforcement learning. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 358–366, San Francisco, CA, 1997. Morgan Kaufmann.

[142] The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 725(5):25–29, 2000.

[143] The Gene Ontology Consortium. Creating the gene ontology resource: design and implementation. *Genome Res.*, 11:1425–1433, 2001.

[144] The Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Res.*, 32:D258–D261, 2004.

[145] K. Tomii and M. Kanehisa. Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins. *Protein Engineering*, 9:27–36, 1996.

[146] Giorgio Valentini and Francesco Masulli. Ensembles of learning machines. In R. Tagliaferri and M. Marinaro, editors, *Neural Nets WIRN Vietri–2002*, volume 2486 of *Lecture Notes in Computer Sciences*, pages 3–19. Springer-Verlag, 2002.

[147] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[148] Volkan Vural and Jennifer G. Dy. A hierarchical method for multi-class support vector machines. In *Twenty-first international conference on Machine learning*. ACM Press, 2004.

[149] J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu. New techniques for extracting features from protein sequences. *IBM Systems Journal*, 40(2):426–441, 2001.

[150] Michael S. Waterman. *Introduction to Computational Biology. Maps, Sequences, and Genomes. Interdisciplinary Statistics*. Chapman & Hall, 1995.

[151] Ian H. Witten and Eibe Frank. *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman, 2000.

[152] Y. I. Wolf, N. V. Grishin, and E. V. Koonin. Estimating the number of protein folds and families from complete genome data. *J. Mol. Biol.*, 299:897–905, 2000.

[153] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[154] Rui yan Luo, Zhi ping Feng, and Jia kun Liu. Prediction of protein structural class by amino acid and polypeptide composition. *Eur. J. Biochem.*, 269:4219–4225, 2002.

[155] Golan Yona and Michael Levitt. Within the twilight zone: A sensitive profile-profile comparison tool based on information theory. *J. Mol. Biol.*, 315:1257–1275, 2002.

[156] C. T. Zhang, Z. S. Lin, Z. Zhang, and M. Yan. Prediction of the helix/strand content of globular proteins based on their primary sequences. *Protein Engineering*, 11:971–979, 1998.