Model-based classification for subcellular localization prediction of proteins

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte
Dissertation
von
Alla Bulashevska
aus Odessa

Tag der mündlichen Prüfung: 28 Juni 2005
Erster Gutachter: Prof. Dr. Roland Vollmar
Zweiter Gutachter: Prof. Dr. Roland Eils

Abstract

The present thesis is a result of my work conducted in the Division Theoretical Bioinformatics at the German Cancer Research Center (DKFZ).

The explosive growth in the amount of biological data demands the use of computers for the maintenance and the analysis of these data. This led to the evolution of Bioinformatics, an interdisciplinary field at the intersection of biology, computer science and information technology.

This thesis deals with the task, which comes from Proteomics. One of the concerns of Proteomics is the prediction of protein properties such as active sites, domains, secondary structure, shape, localization and interactions. The goal of my work is to automate the classification of proteins into subcellular location categories. This thesis focuses on the construction of predictors which use only the primary sequence of a protein for this classification task.

As a base learning algorithm I use Bayesian classification procedure. This scheme represents each class with a single probabilistic summary. I examined the use of two models for the description of class density, Markov Chain Model and Multinomial Model. I found that Bayesian classification used with Markov Chain Model was superior in terms of prediction accuracy.

I propose the extension of this base learning procedure that seek to further improve the accuracy. I introduce a new learning algorithm, termed as Bayesian Classification Tree (BCT), which combines the recursive structure of decision trees with the Bayesian classification procedure. The results are encouraging, since for all real-world data sets used for the evaluation of algorithms in this study, the hybrid approach outperforms significantly the original Bayesian classification. BCT also outperform clearly other methods proposed and investigated in this work, which suggests that BCT is a useful tool for subcellular localization prediction of proteins.

Since different learning algorithms employ different knowledge representations and search heuristics, different search spaces are explored and diverse results are obtained. This is the reason why I explore also the use of ensemble learning methods. Ensemble methods build sets of classifiers using a base learning algorithm and classify new examples by combining their predictions. I evaluate different schemes to generate base classifiers: bagging, pairwise classification, different alphabet size for the Markov Chains using the knowledge about structural properties of amino acids, learning classifiers from different regions of the protein sequences. I have empirically evaluated several meta-learning schemes, such as stacking, arbitration, grading and construction of the decision tree of base classifiers.

Further, I propose another divide-and-conquer approach, which constructs a type of probabilistic decision tree with classifiers in the leaves and mixture models on non-terminal nodes. A general training approach based on maximum likelihood, called Expectation Maximization (EM)

algorithm, is used to estimate models on the intermediate and terminal nodes of the tree.

I extend the general framework of Mixtures of Experts, and especially of localized MEs, and propose to use the Bayesian classifier as gate and experts. I outline how this model can be trained by the EM algorithm.

In addition, I investigate the recently proposed paradigm of delegation, evaluate it and offer two extensions, which connect this paradigm with ensemble and meta learning.

All the methods introduced in this thesis present the new way for the analysis of sequence data and contribute to further development of Bioinformatics, Proteomics and Machine Learning.

Key words: Machine Learning, Data Mining, Classification, Bayesian Classification, Decision Trees, Markov Chains, Ensemble methods, Clustering, Mixture Models, EM algorithm, Mixtures of Experts, Delegating Classifiers, Bioinformatics, Proteomics, Protein Subcellular Localization Prediction

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

High throughput sequencing technology has made it possible to determine the complete sequences of a number of genomes. There are more than 1200 genome sequences deposited in public databases [EBI, 2003]. This has created the need for fully automated methods to analyze sequence data and to identify or classify individual genes and proteins.

The most reliable way to determine a biological molecule‘s structure or function is by direct experimentation. The Human Genome Project gave us the raw sequences of an estimated 100000 human genes, only a small fraction of which have been studied experimentally. This provides strong motivation for developing computational methods that can infer biological information from sequence alone.

Demands for sophisticated analyses of biological sequences are driving forward the newly-created and explosively expanding research area of computational molecular biology, or bioinformatics.

Discerning significant similarities between anciently diverged sequences amidst a chaos of random mutation, natural selection and genetic drift presents serious signal to noise problems. Many of the most powerful sequence analysis methods are now based on principles of probabilistic modelling. Examples of such methods include sequence alignments, the use of Hidden Markov Models (HMMs) to identify distant members of sequence families, for prediction of protein secondary structure and modelling of binding motifs, gene finding and the inference of phylogenetic trees using maximum likelihood approaches.

As part of this annotation process a number of systems have been developed that support automated prediction of subcellular localization of proteins. To cooperate towards the execution of a common physiological function (metabolic pathway, signal-transduction cascade, cytoskeleton, etc.), proteins must be localized in the same cellular compartment. The subcellular localization of a protein, the location or compartment it occupies within the cell, is one of its most basic features. There is an involved machinery within the cell for sorting newly synthesized proteins and sending them to their final locations. Identifying the destination of proteins in the cell is key to understanding their function and facilitating their purification. Even if the basic function

of a protein is known, knowledge of the subcellular location may provide insights as to which pathway an enzyme is involved in. A schematic illustration is given in 1.1 to show the different subcellular locations of proteins.

Experimental determination of subcellular location is mainly accomplished by three approaches: cell fractionation, electron microscopy and fluorescence microscopy. However, currently it is still time-consuming and costly to acquire the knowledge solely based on experimental measures.

Automated prediction of protein subcellular localization is an important tool for genome annotation and drug discovery. Many efforts were made in this regard. Actually, a new branch in proteomics, the so-called *Prediction of Protein Cellular Attributes* has emerged [Chou, 2002].

[Nakai and Kanehisa, 1992] developed an integrated expert system called PSORT to sort proteins into different compartments using sequentially applied „if-then“ rules. The rules were based on different signal sequences, cleavage sites, and the amino acid composition of individual proteins. At every node of an „if-then“ tree a protein was classified into a category based on whether it satisfied a certain condition. One advantage of this process was that it could potentially mimic the actual physical decisions in the real sorting process.

Among other existing computational prediction methods there are three basic approaches.

One approach is based on amino acid composition using artificial neural nets (ANN), such as NNSPL [Reinhardt and Hubbard, 1998], or support vector machines (SVM), used in SubLoc [Hua and Sun, 2001].

A second approach such as TargetP of [Emanuelsson et al., 2000] uses the existence of peptide signals, which are short subsequences of approximately 3 to 70 amino acids, to predict specific cell locations. For example, the KDEL, SKL and SV40-like motifs characterize endoplasmic reticulum (ER), peroxisome and nuclear proteins.

ProLoc [Xie et al., 2002] can be classified as a method combining both amino acid composition and sorting signals. ProLoc searches also for compartment-specific domains.

A third approach such as the one used in LOCkey [Nair and Rost, 2002] is to do a similarity search on the sequence, extract text from homologs and use a classifier on the text features. Some tools combine a variety of individual predictors.

Existing predictors have several shortcomings. The performance of the existing programs varies. Most prediction methods achieve high accuracy for the most populated compartments, such as the nucleus and cytosol, but are generally less accurate on the numerous compartments containing fewer individual proteins. The lack of data for certain localization sites contributes to poor performance for specific localizations. Data sets used for training vary in size. Many existing predictors use only three or four different subcellular localizations. Very few predictors deal with the issue of multicompartmental proteins (proteins that may be localized to different organelles). Currently, there is no precise estimate of how many proteins are multicompartmental.

As pointed out by [Reinhardt and Hubbard, 1998], many genes are automatically assigned in large genome analysis projects, and these assignments are often unreliable for the 5‘-regions. This can result in missing or only partially included leader sequences, thereby causing problems for sequence-motif-based localization algorithms. Similar considerations apply to the localization of EST (expressed sequence tags) fragments.

In this work I propose several methods for accurate automated prediction of localization from sequence information alone. Moreover, I use the primary sequence of a protein for the prediction, without employing methods for homology analysis, identification of sorting signals and other motifs. It may enable my methods to better localize proteins for which gene-prediction places the N-terminus incorrectly.

1.2 Thesis layout and original contribution

The aim of my work is to make a good predictor. This means finding a Machine Learning method that generates a predictor based on a set of examples, where the predictor is a good approximation of the function that generated the examples.

This thesis focuses mostly on classification task. A widely applied method in the machine learning and statistical community is Bayesian classification. Bayesian classifiers work well on a wide range of problems. They possess the advantages of learning speed, simpleness, incrementality. The method for implementing the Bayesian classifier is based on obtaining the posterior probabilities of class membership through the estimation of the class prior probabilities and the class conditional densities. This is a generative approach to classification, since a model of the joint distribution of the input data and the class labels is provided.

I apply the idea of Bayesian classification to the problem of prediction of protein subcellular locations. The core part of the design of the Bayesian classifier is the selection of the appropriate model for the generation of data instances of each class. I examined the use of two such models, Markov Chain Model and Multinomial Model. Chapter 2 focuses on the use of Bayesian classification and describes the models for density estimation.

To solve complex classification problems we can use hierarchical architectures, just like linear networks have led to multi-layer perceptrons. I decided to exploit the idea of recursive partitioning, which was widely used in the classification with decision trees, and designed a hybrid algorithm, which I present in Chapter 3. I called it Bayesian Classification Tree (BCT). The main idea is to use Bayesian classification on each non-terminal node of the decision tree. After a short recall of the principles of decision trees, my approach is defined and compared with other related hybrid approaches. The last part of the Chapter 3 shows the results of experiments on all data sets. The results reveal the superiority of Bayesian Classification Trees when compared to the single Bayesian classification.

In the nineties, Meta Machine Learning (MML) methods have been developed for combining predictors. The intuition that different classifiers behave in qualitatively different ways has motivated attempts to build a better meta classifier via some combination of classifiers. The purpose can be to achieve lower generalization error as the combined predictor can be better than the single predictors. Another purpose is to prevent overfitting on the training set.

Two groups of meta machine learning methods exist: Ensemble methods and Mixtures of Experts (ME) methods.

Chapter 4 deals with Ensemble methods. The attraction that this topic exerts on Machine Learning researchers is based on the premise that ensembles are often much more accurate than the individual classifiers that make them up. An overview of various ensemble methods is given.

I employed several base-level classifiers and classifier combination methods in my comparative studies. I generate different classifiers by using different data representation: by manipulating the training set, by using different alphabets of amino acids for Markov Chain Models, by learning classifiers from different regions of the protein sequences. I also investigate the use of pairwise classification for multiclass problem. The base level learning algorithm still remains Bayesian classification. The generated classifiers are then combined. I have empirically evaluated several state-of-the-art methods for ensemble combination: stacking, arbitration, grading and construction of the decision tree of classifiers.

In Chapter 5 I propose another divide-and-conquer approach, which constructs a type of probabilistic decision tree with classifiers in the leaves and mixture models on non-terminal nodes. In order to learn the parameters of the mixture model, a general training approach based on maximum likelihood, called Expectation Maximization (EM) algorithm, is used.

The term Mixtures of Experts covers a variety of methods. In Chapter 6 I extend the general framework of Mixtures of Experts, and especially of localized MEs. Thus, we have an architecture consisting of gating Bayesian classifier which partition the data and weight the expert Bayesian classifier predicting the class probabilities. It is also outlined how this model can be trained by the EM algorithm.

Chapter 7 is dedicated to the modern paradigm of delegation. The key idea of a delegating classifier is that it only makes predictions with a maximum level of confidence and delegates the prediction to another classifier otherwise. My contribution in this part of the thesis consists of empirical evaluation of the delegating paradigm on my classification task and developing of two further extensions. One is the use of ensemble of classifiers instead of a single classifier at each stage of the delegation process. The second extension provides some connections between delegating and meta-learning.

Finally, a summary is given in Chapter 8 and proposals for future work.

All the methods proposed in this thesis were succesfully implemented by myself and empirically evaluated on the different biological data sets, which I describe further in this introductory chapter.

1.3 Data

The input data for my algorithms is the amino acid sequences of the proteins, classified into different subcellular location groups. Figure 1.2 shows a cutoff from the file with sequences, assigned to Golgi apparatus.

I applied my methods to the previously published data sets of protein sequences used by other working groups.

1.3.1 Data set of Reinhardt and Hubbard

The first data set was previously used by [Reinhardt and Hubbard, 1998], [Yuan, 1999] and also by [Hua and Sun, 2001]. It included only globular proteins, because transmembrane proteins could be predicted with a much higher accuracy by some known methods (see [Krogh et al., 2001]).

As shown in Table 1.1, there are 2427 protein sequences from eukaryotic species classified into four location groups: cytoplasmic, extracellular, nuclear and mitochondrial. There are also 997 prokaryotic sequences, which were assigned to three location categories: cytoplasm, extracellular and periplasmic, shown in Table 1.2.

Cellular location	Number of proteins
Cytoplasmic	684
Extracellular	325
Mitochondrial	321
Nuclear	1097
Sum	2427

Table 1.1: Eukaryotic sequences within each subcellular location group of the Reinhardt data set (Data_Euk).

Cellular location	Number of proteins
Cytoplasmic	688
Extracellular	107
Periplasmic	202
Sum	997

Table 1.2: Prokaryotic sequences within each subcellular location group of the Reinhardt data set (Data_Prok).

1.3.2 Data set of Huang and Li

The second data set was used as the raw data set in [Huang and Li, 2004]. Sequences were selected from all eukaryotic proteins with annotated subcellular location in SWISS-PROT release 41.0 [Boeckmann et al., 2003]. All proteins with ambiguous words such as PROBABLE, POTENTIAL, POSSIBLE and BY SIMILARITY and also proteins with multiple annotations of locations were excluded. The transmembrane proteins were excluded also. The number of proteins and their distributions in 11 categories are listed in Table 1.3.

1.3.3 Apoptosis proteins

Apoptosis, or programmed cell death, is a fundamental process controlling normal tissue homeostasis by regulating a balance between cell proliferation and death. This process is currently an area of intense investigation. When apoptosis malfunctions, a variety of formidable diseases can ensue: blocking apoptosis is associated with cancer and autoimmune disease, whereas unwanted apoptosis can possibly lead to ishemic damage or neurodegenerative disease.

Cellular location	Number of proteins
Chloroplast	1141
Cytoplasm	2437
Cytoskeleton	24
Endoplasmic	132
Extracellular	4165
Golgi	32
Lysosome	131
Mitochondria	1100
Nuclear	3326
Peroxisome	122
Vacuole	53
Sum	12663

Table 1.3: Protein sequences within each subcellular location group of the data set of Huang and Li (Data_SWISS).

Many efforts in pharmaceutical research have been aimed at understanding the structure and function of apoptosis proteins. To understand the apoptosis mechanism and functions of various apoptosis proteins, it would be helpful to obtain information about their subcellular location.

The authors of [Zhou and Doctor, 2003] constructed a training data set, containing 98 apoptosis proteins classified into four categories (see Table 1.4). The proteins were derived from SWISS-PROT [Bairoch, 2000]. Of the 12 other apoptosis proteins, five are located in nucleus, one in microtubule, and one in lysosome. [Zhou and Doctor, 2003] used covariant discriminant function for their prediction and the results of their study can be seen in Table 1.4. In my study I used the same data set, which I call *Data_Apoptosis*, except one protein (Q9OX1), the sequence of which could not be retrieved.

Cellular location	Number of proteins	Accuracy (%)
Cytoplasmic	43	97.7
Plasma membrane	30	73.3
Mitochondrial	13	30.8
Other	12	25.0
Sum	98	72.5

Table 1.4: Apoptosis proteins within each subcellular location group (Data_Apoptosis) and prediction accuracies achieved in [Zhou and Doctor, 2003].

1.3.4 Gram-negative bacteria data set

A manually curated data set of proteins of experimentally known subcellular localization was constructed by [Gardy et al., 2003]. Gram-negative bacterial sequences with an annotated sub-

cellular localization were extracted from SWISS-PROT release 40.29 [Bairoch, 2000]. All proteins, denoted as fragments and whose annotations were listed as „by similarity“ or „putative“ or with ambiguous annotation, were removed. The proteins were manually checked against the literature for experimental verification of the annotated localization. The final data set consists of 1443 proteins and is available online at http://www.psort.org/dataset. Gram-negative bacteria have five major subcellular localization sites. The data set comprises 1302 proteins resident at a single localization site and 141 proteins resident at multiple localization sites. For my experiments I used the newest version of the data set (see Table 1.5).

Following the method of [Gardy et al., 2003], for the sequences with dual locations, if one of their locations is predicted, I will consider them as correctly predicted.

Cellular location	Number of proteins
Cytoplasmic	278
Cytoplasmic/Cytoplasmic Membrane	16
Cytoplasmic Membrane	309
Cytoplasmic Membrane/ Periplasmic	51
Periplasmic	276
Periplasmic/Outer Membrane	2
Outer Membrane	391
Outer Membrane/Extracellular	78
Extracellular	190
Sum	1591

Table 1.5: Bacterial proteins within each subcellular location group (Data_Gram).

The most widely used predictive tool for Gram-negative bacteria has been PSORT I of [Nakai and Kanehisa, 1991]. However, it does not predict extracellular sequences and its overall prediction accuracy reaches only 61%.

[Gardy et al., 2003] developed a multimodular method PSORT-B, which comprises six modules examining the query sequence for different characteristics such as:

- amino acid composition,
- similarity to proteins of known localization,
- presence of a signal peptide,
- transmembrane $alpha$- helices,
- motifs corresponding to specific localization.

PSORT-B contains an outer membrane motifs module, a classifier, which uses a machine learning approach to identify frequent sequences occurring only in beta-barrel proteins, both integral outer membrane proteins and autotransporter proteins, which possess a beta-barrel transport domain.

This program then constructs a Bayesian Network to generate a final probability value for each localization site. This approach reaches an overall prediction accuracy of 74.8%, significantly improving on the previous results of PSORT I by 14%. However, gives modest prediction for some location categories, such as cytoplasmic (69.4%) and periplasmic (57.6%). The authors argue that it is due in part to the lack of experimental study of periplasmic proteins. PSORT-B predicts outer membrane proteins most accurately of all the localizations (90.3%). This was a particular focus of authors, because outer membrane proteins- as primary cell surface components of Gram-negative bacteria - are attractive potential vaccine targets, diagnostic agents and drug targets of medical, agricultural and environmental interst.

1.4 Accuracy measures

To compare the prediction performance of the classification methods I used standard performance measures.

By *Jack-knife test* (or leave-one-out cross-validation) the learning step is performed with all sequences except the one, for which the location is to be predicted. The prediction quality was evaluated by the overall prediction accuracy and prediction accuracy for each location:

$$\text{overall accuracy} = \frac{\sum_{c=1}^{K} T(c)}{N}$$

$$\text{accuracy(c)} = \frac{T(c)}{N(c)},$$

where N is the total number of sequences, $N(c)$ is the number of sequences observed in location c, K is the number of locations and $T(c)$ is the number of correctly predicted sequences of location c.

In k*-fold cross-validation* the data set is partitioned randomly into k equally-sized partitions, and learning and evaluation is carried out k times, each time using one distinct partition as the testing set and the remaining $k - 1$ partitions as the training set. The choice of $k = 5$ implies that 80% of the sequences are used for training and 20% for testing.

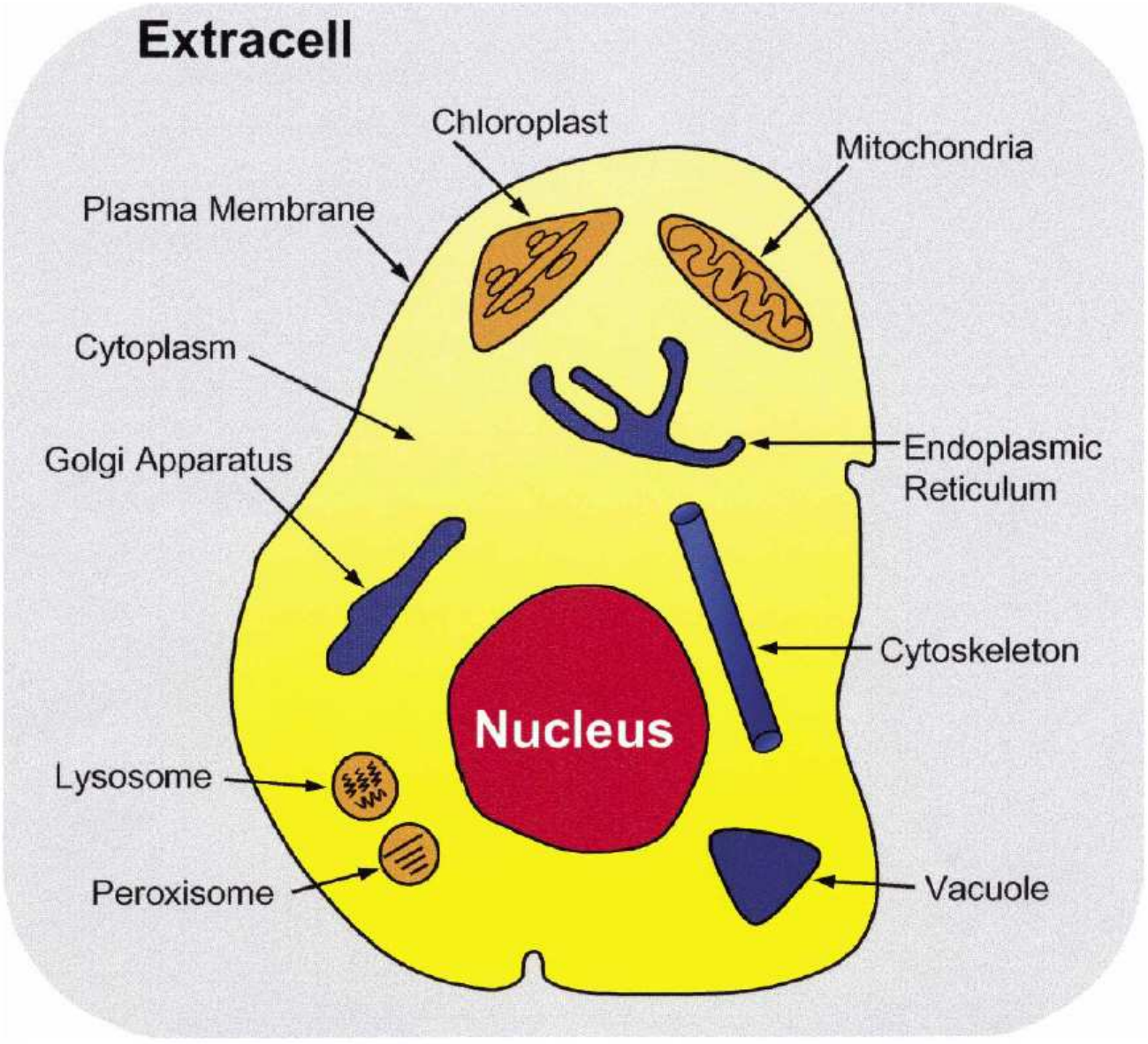

Figure 1.1: Schematic illustration to show the twelve subcellular locations of proteins: chloroplast, cytoplasm, cytoskeleton, endoplasmic reticulum, extracell, Golgi apparatus, lysosome, mitochondria, nucleus, peroxisome, plasma membrane, and vacuole. Note that the vacuole and chloroplast proteins exist only in a plant. Reproduced from [Chou, 2001].

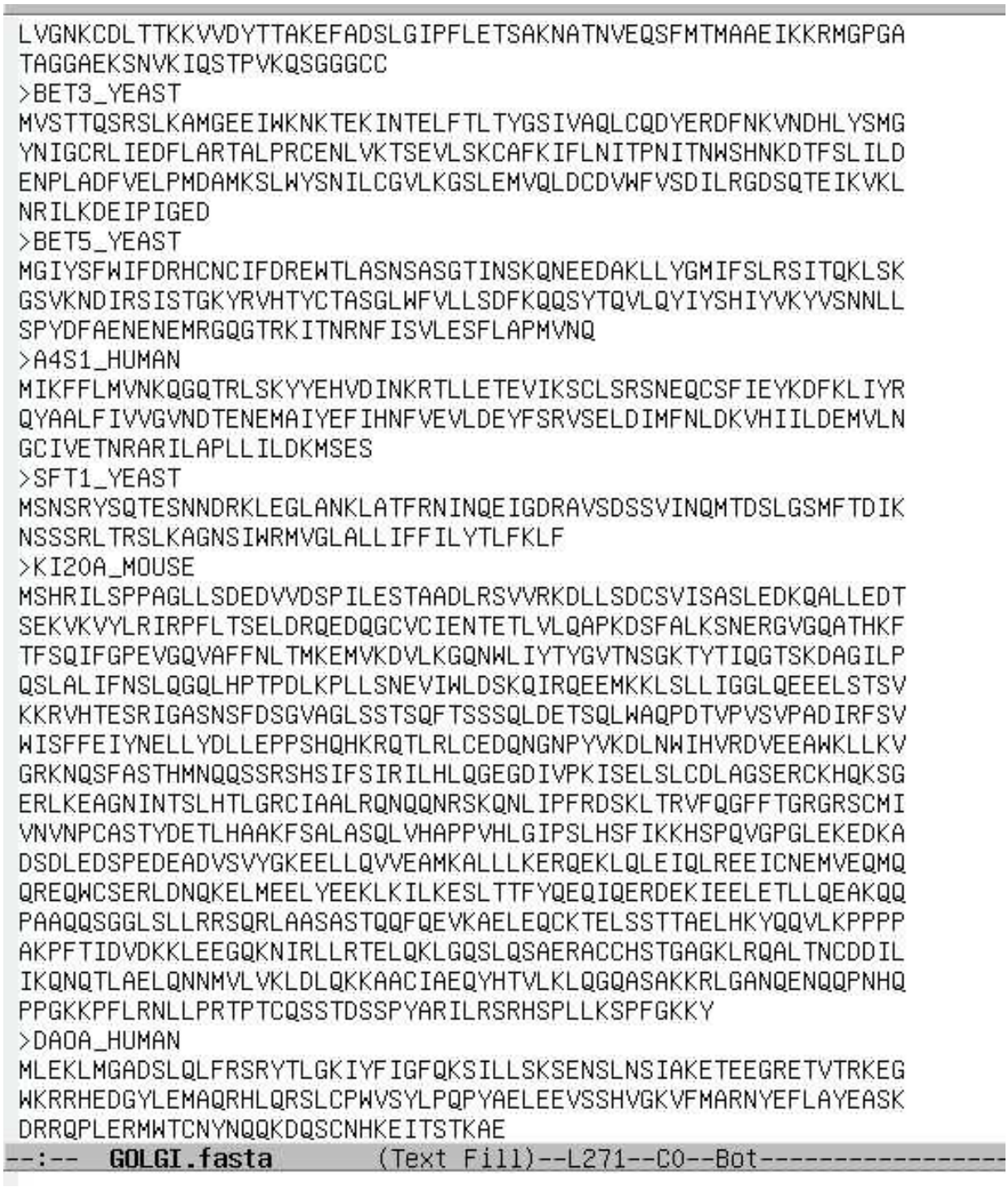

Figure 1.2: Cutoff from the file with amino acid sequences of proteins, assigned to Golgi apparatus.

Chapter 2

Bayesian classifier for subcellular localization prediction of proteins

In this Chapter I review the induction of Bayesian classifiers, propose two models for the class-conditional densities and report the results of application of Bayesian Classification procedure on my real-world data sets.

2.1 Bayesian classifiers

In supervised Machine Learning, a learning algorithm is given a training set D including N training instances $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $y_i \in \{c_1, \ldots, c_K\}$ (a set of K classes). A learning algorithm produces a classifier, which is a hypothesis about the unknown function f for which $y = f(x)$. The classifier is later used to predict the unknown class $y_i \in \{c_1, \ldots, c_K\}$ for a new instance x_i.

One approach to produce a classifier is to use the Bayes' theorem (the Bayes rule). According to it, the class for x_i should be the one which maximizes the probability:

$$P(c_j|x_i) = \frac{P(c_j)P(x_i|c_j)}{P(x_i)} = \frac{P(c_j)P(x_i|c_j)}{\sum_k P(c_k)P(x_i|c_k)} = \frac{prior * class-conditional}{normalization}$$

Estimating the probability $P(x_i)$ is unnecessary because it is the same for all classes. The remaining probabilities can be estimated from the training set D. Priors are often estimated as the proportion of samples of class or using a priori knowledge.

One must specify how to compute the term $P(x_i|c_j)$. In my work I model the class-conditional densities with two parametric methods, which I introduce in the next two Sections. In nominal domains, one typically stores a discrete distribution for each attribute in a description. The *naive Bayesian classifier* assumes independence of attributes within each class.

Note that the estimation of the class-conditional densities involves K-subproblems, in which each of the density is estimated based on the data belonging to the class only. As a consequence it is straightforward to introduce new classes without having to reestimate the whole model.

Advantage of the Bayesian classifier is its ease of training. The basic process can operate either incrementally or nonincrementally, since the order of training instances has no effect on learning. In contrast to many induction methods, which learn only when they make some error, a Bayesian classifier incorporates information from *every* instance that it encounters.

A possible criticism of Bayesian classifiers is that they are modeling too much: for each class many aspects of the data are modeled which may or may not play a role in discriminating between classes. Often Bayesian classifiers also require more parameters and more computation during recall since when a new example is presented, the posterior probabilities of all classes need to be calculated.

Bayesian classifier relies on an important assumption: that the variability of the data set can be summarized by a single probabilistic description, and that this is sufficient to distinguish between classes. From an analysis of *Bias-Variance* this implies that Bayesian classifier uses a reduced set of models to fit the data. The result is low variance, but if the data can not be adequately represented by the set of models, we obtain a large bias.

2.2 Models used

2.2.1 Markov Chain model

Let s be a protein sequence of length n,

$$s = s_1 s_2 \dots s_{i-1} s_i \dots s_n,$$

where s_i is the amino acid residue at sequence position i.

For a first-order Markov model the frequencies of the residues in position i depend on the residue in position $i-1$. The probability of a sequence s to belong to the class c is given by the ordinary Markov chain formula:

$$P^c(s) = P_1^c(s_1) P_2^c(s_2|s_1) P_3^c(s_3|s_2) \dots P_n^c(s_n|s_{n-1})$$

Here $P_n^c(s_n|s_{n-1})$ is the conditional probability (also called transition probability) of observing residue s_n in position n, given that s_{n-1} is in position $n-1$. Because s_n and s_{n-1} can be any of the 20 amino acids, the statistics of consecutive pair-residues will generate a matrix with $20 * 20$ elements, each representing the occurring frequency of amino acid pair (s_{n-1}, s_n), denoted by $F^c(s_{n-1}, s_n)$.

The conditional probability can be calculated as:

$$P_n^c(s_n|s_{n-1}) = \frac{F^c(s_{n-1},s_n)}{\sum_{s_n} F^c(s_{n-1},s_n)}$$

Markov chain models for the prediction of protein subcellular locations were first used by [Yuan, 1999].

2.2.2 Multinomial Model

Introducing class-conditional amino acid distributions $P(a|c)$, the probability of a sequence s to belong to the class c is given by

$$P(s|c) = \prod_a P(a|c)^{n(s,a)},$$

where count variables $n(s, a)$ indicate how often an amino acid a occurred in a sequence s .

The parameters of each class, i.e. class-conditional amino acid distributions, can be estimated from the training set D as following:

$$P(a|c) = \frac{\sum_{s|c_s=c} n(s,a)}{\sum_{s|c_s=c} n(s)}$$

Here $n(s) = \sum_a n(s, a)$ denotes the length of the sequence.

2.3 Results and discussion

The Bayesian Classification approach was validated with *Jack-knife test* and confusion matrices were constructed according to the results of this procedure.

I found that Markov Chain Model works better compared to Multinomial Model concerning the prediction performance. Subsection 2.3.1 presents the results of Bayesian classification based on Multinomial Model for some data sets, Subsection 2.3.2 presents the results of Bayesian classification based on Markov Chain Model for all data sets used in the study.

2.3.1 Results of Bayesian Classifi cation based on Multinomial Model

Tables 2.1 - 2.6 report the corresponding results.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**489**	70	61	64	684
Extracellular	67	**159**	29	70	325
Mitochondrial	132	33	**127**	29	321
Nuclear	176	61	101	**759**	1097
Sum	864	323	318	922	2427

Table 2.1: Confusion matrix of prediction results of Bayesian Classification approach based on Multinomial Model for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	71.5
Extracellular	48.9
Mitochondrial	39.6
Nuclear	69.2
Overall accuracy	**63.2**

Table 2.2: The predictive accuracy for subcellular locations of Bayesian Classification approach based on Multinomial Model for Data_Euk.

	Predicted group			
	Cytoplasmic	Extracellular	Periplasmic	Sum
Cytoplasmic	**631**	1	56	688
Extracellular	6	**84**	17	107
Periplasmic	40	22	**140**	202
Sum	677	107	213	997

Table 2.3: Confusion matrix of prediction results of Bayesian Classification approach based on Multinomial Model for Data_Prok.

Cellular location	Accuracy (%)
Cytoplasmic	91.7
Extracellular	78.5
Periplasmic	69.3
Overall accuracy	**85.8**

Table 2.4: The predictive accuracy for subcellular locations of Bayesian Classification approach based on Multinomial Model for Data_Prok.

	Predicted group				
	Cytoplasmic	Plasma membrane	Mitochondrial	Other	Sum
Cytoplasmic	**34**	2	4	3	43
Plasma membrane	3	**17**	10	0	30
Mitochondrial	1	0	**11**	0	12
Other	6	0	0	**6**	12
Sum	44	19	25	9	97

Table 2.5: Confusion matrix of prediction results of Bayesian Classification approach based on Multinomial Model for Data_Apoptosis.

Cellular location	Accuracy (%)
Cytoplasmic	79.1
Plasma membrane	56.7
Mitochondrial	91.7
Other	50.0
Overall accuracy	**70.1**

Table 2.6: The predictive accuracy for subcellular locations of Bayesian Classification approach based on Multinomial Model for Data_Apoptosis.

2.3.2 Results of Bayesian Classification based on Markov Chain Model

Tables 2.7 - 2.16 report the corresponding results.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**539**	32	60	53	684
Extracellular	60	**200**	37	28	325
Mitochondrial	118	14	**168**	21	321
Nuclear	167	42	76	**812**	1097
Sum	884	288	341	914	2427

Table 2.7: Confusion matrix of prediction results of Bayesian Classification approach based on Markov Chain Model for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	78.8
Extracellular	61.5
Mitochondrial	52.3
Nuclear	74.0
Overall accuracy	**70.8**

Table 2.8: The predictive accuracy for subcellular locations of Bayesian Classification approach based on Markov Chain Model for Data_Euk.

	Predicted group			
	Cytoplasmic	Extracellular	Periplasmic	Sum
Cytoplasmic	**654**	2	32	688
Extracellular	7	**80**	20	107
Periplasmic	32	17	**153**	202
Sum	693	99	205	997

Table 2.9: Confusion matrix of prediction results of Bayesian Classification approach based on Markov Chain Model for Data_Prok.

Cellular location	Accuracy (%)
Cytoplasmic	95.1
Extracellular	74.8
Periplasmic	75.8
Overall accuracy	**89.0**

Table 2.10: The predictive accuracy for subcellular locations of Bayesian Classification approach based on Markov Chain Model for Data_Prok.

	Predicted group											
	Chlor	Cytop	Cytos	End	Ext	Gol	Lys	Mit	Nuc	Per	Vac	Sum
Chloroplast	**604**	159	3	10	77	0	3	213	64	8	0	1141
Cytoplasm	100	**1275**	181	62	190	20	15	320	229	44	1	2437
Cytoskeleton	0	4	**13**	0	1	0	0	0	6	0	0	24
Endoplasmic	4	16	0	**67**	9	0	17	9	8	2	0	132
Extracellular	63	290	0	43	**2971**	28	112	157	435	24	42	4165
Golgi	2	11	0	0	5	**5**	1	1	7	0	0	32
Lysosome	4	5	0	0	12	0	**107**	1	1	1	0	131
Mitochondria	82	182	0	13	56	1	12	**665**	65	21	3	1100
Nuclear	70	471	23	106	262	12	17	150	**2203**	7	5	3326
Peroxisome	10	21	0	1	5	0	1	31	3	**50**	0	122
Vacuole	1	6	0	3	17	0	4	5	2	0	**15**	53
Sum	940	2440	220	305	3605	66	289	1552	3023	157	66	12663

Table 2.11: Confusion matrix of prediction results of Bayesian Classification approach based on Markov Chain Model for Data_SWISS.

Cellular location	Accuracy (%)
Chloroplast	52.9
Cytoplasm	52.3
Cytoskeleton	54.2
Endoplasmic ret	50.8
Extracellular	71.3
Golgi apparatus	15.6
Lysosome	81.7
Mitochondria	60.5
Nuclear	66.2
Peroxisome	41.0
Vacuole	28.3
Overall accuracy	**63.0**

Table 2.12: The predictive accuracy for subcellular locations of Bayesian Classification approach based on Markov Chain Model for Data_SWISS.

	Predicted group				
	Cytoplasmic	Plasma membrane	Mitochondrial	Other	Sum
Cytoplasmic	**39**	2	2	0	43
Plasma membrane	0	**27**	3	0	30
Mitochondrial	0	1	**11**	0	12
Other	5	1	0	**6**	12
Sum	44	31	16	6	97

Table 2.13: Confusion matrix of prediction results of Bayesian Classification approach based on Markov Chain Model for Data_Apoptosis.

Cellular location	Accuracy (%)
Cytoplasmic	90.7
Plasma membrane	90
Mitochondrial	91.7
Other	50.0
Overall accuracy	**85.6**

Table 2.14: The predictive accuracy for subcellular locations of Bayesian Classification approach based on Markov Chain Model for Data_Apoptosis.

	Predicted group					
	Cytoplasm	Inner membrane	Periplasm	Outer membrane	Extracell	Sum
Cytoplasmic	**248**	2	31	4	8	293
Inner membrane	43	**268**	9	8	3	331
Periplasmic	27	9	**233**	23	15	307
Outer membrane	46	6	28	**355**	19	454
Extracellular	11	8	24	39	**124**	206
Sum	375	293	325	429	169	1591

Table 2.15: Confusion matrix of prediction results of Bayesian Classification approach based on Markov Chain Model for Data_Gram.

Cellular location	Accuracy (%)
Cytoplasmic	84.6
Inner membrane	80.7
Periplasmic	75.9
Outer membrane	78.2
Extracellular	60.2
Overall accuracy	**77.2**

Table 2.16: The predictive accuracy for subcellular locations of Bayesian Classification approach based on Markov Chain Model for Data_Gram.

Chapter 3

Bayesian Classification Trees: hybrid approach

I present in this Chapter a new supervised learning procedure, which I call Bayesian Classification Tree (BCT). This is a hybrid representation, learned by a combination of two methods, decision trees and Bayesian classification. The combination of two formalisms into a hybrid makes it possible to draw on the particular strengthes of each of the individual formalisms. In the next Section I give a short introduction to decision trees. In Section 3.2 I define my new algorithm. Section 3.3 relates the proposed algorithm to the previously proposed methods. Section 3.4 demonstrates the results of the empirical study using my data sets.

3.1 Decision trees

Most of Machine Learning algorithms for supervised learning problems use a *divide and conquer* strategy that attacks a complex problem by dividing it into simpler problems and recursively applies the same strategy to the subproblems. Solutions of subproblems can be combined to yield a solution of the complex problems.

A decision tree is a special type of classifier. It uses a standard technique for building classification rules from data, the so called *recursive partitioning* algorithm, which constructs a tree from the training set. The well known decision tree based algorithms are ID3 [Quinlan 1986], CART [Breiman et al., 1984], ASSISTANT [Cestnik 1986] and C4.5 [Quinlan 1993].

A decision tree contains zero or more internal nodes and one or more leaf nodes. A decision (inner) node specifies a test to be carried out on example and each outcome of the test has its own branch leading to the appropriate subtree. All internal nodes have two or more child nodes. Each leaf node has a class label associated with it.

The task of constructing a tree from the training set has been called tree induction or tree growing. Most existing tree induction systems proceed in a greedy top-down fashion. Starting with an empty tree and the entire training set, some variant of the following algorithm is applied until no more splits are possible.

Input: a set of labelled instances.

- If all the training examples at the current node belong to a single class or if some other stopping rule applies, create a leaf node labelled with that class.
- Otherwise, select a test with mutually exclusive outcomes using a splitting rule;
- create as many child nodes as there are outcomes, divide the training set into subsets, each corresponding to one outcome and assign these subsets to the corresponding child node;
- for each child node, call the algorithm recursively.

Stopping rules are used if further growing is unnecessary. Sometimes they employ parameters, which control the complexity of the resulting decision tree. Most recent algorithms stop growing trees when certain conditions are satisfied:

- The node is pure or almost pure, the majority of the examples it contains are of the one class.
- The level of the node is equal to the maximum depth of the tree.
- The size of the node (the number of examples falling at the node) is smaller than a certain size.
- The *utility of a split* of the node is not significantly better as the *utility of the node* [Kohavi, 1996].

A survey of many different splitting rules can be found in [Breiman et al., 1984]. Some common tests are Information Gain, Gini Index of Diversity, χ^2 and G Statistic tests.

One advantage of decision tree classification should be noted. The decision tree segments the data, a task that is an essential part of the Data Mining process in large databases.

The main drawback of the decision tree approach is its instability: small variations of the training set could cause large changes in the resulting predictors. These classifiers have high variance but they can fit any kind of data: the bias of a decision tree is low.

3.2 Bayesian Classification Trees

In this section I introduce my algorithm called Bayesian Classification Tree (BCT).

The BCT builds a decision tree in the well-known top-down manner. The Bayesian Classification algorithm repeatedly presents examples at each node. Trees generated by this procedure are not binary. The number of descendants of each decision node is equal to the number of classes that fall at this node. The intuition behind this procedure is that the majority of the examples, which arrive at one child after the application of Bayesian classifier at the father's node, will have one class label. The outline of the algorithm is as following:

Input: a set of labelled instances.

- If the majority of the training examples at the current node belong to a single class or if the size of the node is smaller than *ThreshSize*, create a leaf node labelled with that class.
- Otherwise, learn Bayesian classifier for the current node from the training examples;
- create as many child nodes as there are classes at the node;
- apply the Bayesian classifier on the examples, e.g. predict for each example its class and assign this example to the corresponding child node;
- for each child node, call the algorithm recursively.

Figure 3.1 shows schematically a Bayesian Classification Tree.

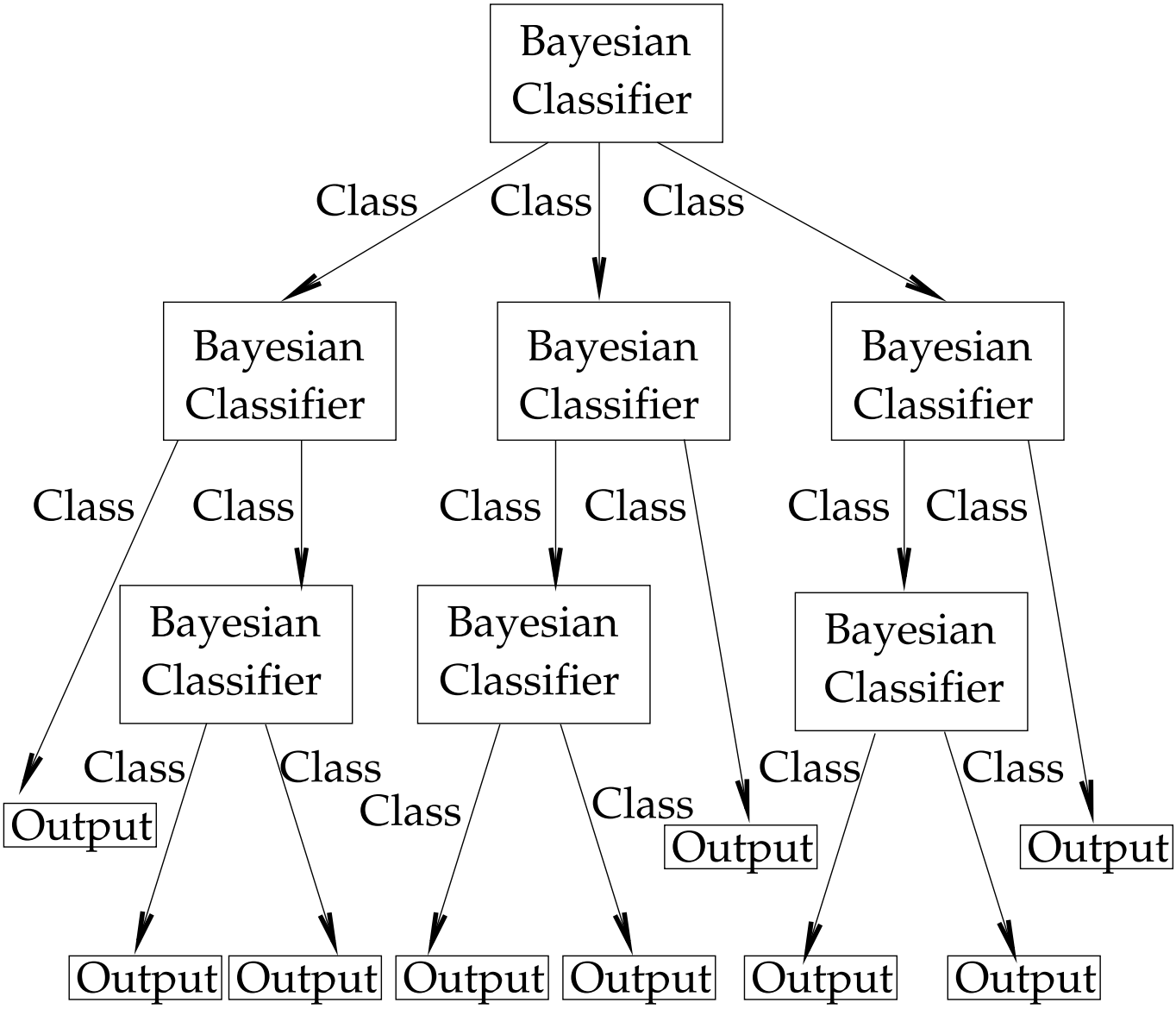

Figure 3.1: Bayesian Classification Tree.

3.3 Related work

The final model of Bayesian Classification Tree approach is closely related with the *recursive naive Bayes* presented in [Langley 1993]. The naive Bayesian classifier was applied to domains with nominal attributes. The author claims that this recursive approach should outperform the naive Bayesian classifier in domains that involve disjunctive concepts, since they violate the independence assumption on which the naive scheme relies. The recursive approach should better classify the cases which occupy noncontiguous regions of the instance space, because one cannot represent such disjunctive situations with a single probabilistic summary for each class. The author managed to show superiority of his method over simple Naive Bayes only on synthetic data specifically generated.

A method for combining decision trees and Linear Threshold Units (LTUs), which are the basic units of Rosenblatt‘s perceptron, has been proposed in [Breiman et al., 1984].

My algorithm resembles also the Linear Machine Decision Tree (LMDT) of Brodley and Utgoff [Brodley and Utgoff, 1995]. Each internal node in LMDT tree is a *linear machine*, which is a set of linear discriminant functions that are used to classify an example. As in LTUs, linear machines specify a set of weights for each class-attribute pair, but they operate competitively. The LMDT is an incremental method for inducing decision trees and requires many passes through the training set.

The FACT system in [Loh, 1988] recursively partition the input space using a linear discriminant function. The number of descendants of each node is equal to the number of classes.

The COBWEB algorithm of [Fisher, 1987] uses an identical organization of probabilistic concepts in a hierarchy.

I should also mention the Probabilistic Linear Tree (*Ltree*) and Probabilistic Bayes Tree (*Btree*) of [Gama 1997]. The method consists of combining a decision tree with a discriminant function by means of *constructive induction* (see [Gama 1998]). Constructive induction discovers new features from the training set and transforms the original instance space into a new one by applying attribute constructor operators. In *Btree* at each decision node a new instance space is defined by the insertion of new attributes, which is derived from the class predictions, given by a naive Bayesian classifier learned at each node.

3.4 Results and discussion

In the experimental study of Bayesian Classification Tree approach I used Bayesian classifiers based on Markov Chain Model.

The results were validated with 10-fold cross-validation procedure.

The results show that my method can significantly improve the accuracy of the predictions when compared to the single global Bayesian classifier. Table 3.1 shows the result of comparison of overall accuracies, achieved with two approaches- Bayesian classification (BC) and Bayesian Classification Trees (BCT)- on all data sets used in this study.

Data set	BC-approach Accuracy (%)	BCT-approach Accuracy (%)
Data_Euk	70.8	78.7
Data_Prok	89.0	89.3
Data_SWISS	63.0	77.4
Data_Apoptosis	85.6	89.7
Data_Gram	77.2	83.2

Table 3.1: Performance comparison of two approaches BC and BCT.

3.4.1 Data set of Reinhardt and Hubbard

Tables 3.2 - 3.5 show the results of BCT procedure with eukaryotic and prokaryotic data. The overall prediction accuracy of 78.7% achieved with BCT for eukaryotic proteins is better than 73.0% of [Yuan, 1999] achieved with fourth-order Markov chains, is comparable with 79.4% achieved with support vector machines in [Hua and Sun, 2001] and is lower than 85.2% of [Huang and Li, 2004]. The overall result of 89.3% for prokaryotic proteins was slightly better than 89.1% of [Yuan, 1999] achieved with fourth-order Markov chains.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**522**	30	46	86	684
Extracellular	28	**256**	9	32	325
Mitochondrial	87	14	**170**	50	321
Nuclear	84	18	33	**962**	1097
Sum	721	318	258	1130	2427

Table 3.2: Confusion matrix of prediction results of Bayesian Classification Tree approach for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	76.3
Extracellular	78.8
Mitochondrial	53.0
Nuclear	87.7
Overall accuracy	**78.7**

Table 3.3: The predictive accuracy for subcellular locations of Bayesian Classification Tree approach for Data_Euk.

	Predicted group			
	Cytoplasmic	Extracellular	Periplasmic	Sum
Cytoplasmic	**657**	1	30	688
Extracellular	8	**78**	21	107
Periplasmic	31	16	**155**	202
Sum	696	95	206	997

Table 3.4: Confusion matrix of prediction results of Bayesian Classification Tree approach for Data_Prok.

Cellular location	Accuracy (%)
Cytoplasmic	95.5
Extracellular	72.9
Periplasmic	76.7
Overall accuracy	**89.3**

Table 3.5: The predictive accuracy for subcellular locations of Bayesian Classification Tree approach for Data_Prok.

3.4.2 Data set of Huang and Li

For the results of experiments with Bayesian Classification Tree approach with this data set see Tables 3.6 and 3.7. It is interesting, that for this big data set the BCT was 14.4% superior than single Bayesian classifier.

	Predicted group											
	Chlor	Cytop	Cytos	End	Ext	Gol	Lys	Mit	Nuc	Per	Vac	Sum
Chloroplast	**840**	90	0	5	41	0	2	95	59	8	1	1141
Cytoplasm	77	**1735**	4	14	147	5	13	141	279	20	2	2437
Cytoskeleton	0	11	**6**	0	1	0	0	0	6	0	0	24
Endoplasmic	6	20	0	**79**	11	1	3	2	9	1	0	132
Extracellular	44	176	0	7	**3513**	15	34	83	256	14	23	4165
Golgi	1	7	0	0	4	**7**	1	2	10	0	0	32
Lysosome	0	8	0	0	23	0	**97**	2	0	1	0	131
Mitochondria	82	145	0	5	73	1	6	**674**	95	19	0	1100
Nuclear	47	232	8	12	156	4	7	100	**2755**	3	2	3326
Peroxisome	8	16	0	1	7	0	1	18	5	**66**	0	122
Vacuole	0	3	0	1	10	0	2	4	6	0	**27**	53
Sum	1105	2443	18	124	3986	33	166	1121	3480	132	55	12663

Table 3.6: Confusion matrix of prediction results of Bayesian Classification Tree approach for Data_SWISS.

Cellular location	Accuracy (%)
Chloroplast	73.6
Cytoplasm	71.2
Cytoskeleton	25.0
Endoplasmic ret	59.8
Extracellular	84.3
Golgi apparatus	21.9
Lysosome	74.0
Mitochondria	61.3
Nuclear	82.8
Peroxisome	54.1
Vacuole	50.9
Overall accuracy	**77.4**

Table 3.7: The predictive accuracy for subcellular locations of Bayesian Classification Tree approach for Data_SWISS.

3.4.3 Apoptosis proteins

For the results of experiments with Bayesian Classification Tree approach with this data set see Tables 3.8 and 3.9. Even with the single Bayesian classifier I reached the overall accuracy of 85.6%, which is 13.1% higher than that reached in [Zhou and Doctor, 2003] with covariant discriminant algorithm (compare with Table 1.4). The BCT approach reaches the overall accuracy of 89.7%.

	Predicted group				
	Cytoplasmic	Plasma membrane	Mitochondrial	Other	Sum
Cytoplasmic	**41**	2	0	0	43
Plasma membrane	1	**27**	2	0	30
Mitochondrial	0	1	**11**	0	12
Other	3	1	0	**8**	12
Sum	45	31	13	8	97

Table 3.8: Confusion matrix of prediction results of Bayesian Classification Tree approach for Data_Apoptosis.

Cellular location	Accuracy (%)
Cytoplasmic	95.3
Plasma membrane	90
Mitochondrial	91.7
Other	66.7
Overall accuracy	**89.7**

Table 3.9: The predictive accuracy for subcellular locations of Bayesian Classification Tree approach for Data_Apoptosis.

3.4.4 Gram-negative bacteria data set

The results of experiments with Bayesian Classification Tree approach with this data set are reported in Tables 3.10 and 3.11.

The overall prediction accuracy of my method reaches 83.2%, which is 8.4% higher than that of PSORT-B (74.8%) introduced in [Gardy et al., 2003], though I do not use specialized algorithms or particular input vectors for each localization site. Compared with PSORT-B, my method gives significantly better predictive performances for all the localization sites except outer membrane proteins (OMPs). PSORT-B reaches 90.3% for OMPs, however, it utilizes an extra module for OMPs based on identification of frequent sequences occurring only in beta-barrel proteins. I reach the accuracy of 87.1% for outer membrane proteins, which is also very high. The identification of OMPs is of particular interest, because they are on the surface of the bacteria and so are the most accessible targets to develop new drugs against.

It is remarkable, that for periplasmic proteins my method reaches the accuracy of 79.1%, which is 21.5% higher than that of PSORT-B.

However, the predictive accuracy of my method is 5.7% lower than that of CELLO, reported in [Yu et al., 2004].

	Predicted group					
	Cytoplasm	Inner membrane	Periplasm	Outer membrane	Extracell	Sum
Cytoplasmic	**223**	13	37	14	3	290
Inner membrane	28	**291**	9	9	2	339
Periplasmic	23	11	**239**	18	11	302
Outer membrane	11	4	18	**378**	23	434
Extracellular	7	4	17	31	**167**	226
Sum	292	323	320	450	206	1591

Table 3.10: Confusion matrix of prediction results of Bayesian Classification Tree approach for Data_Gram.

Cellular location	Accuracy (%)
Cytoplasmic	76.9
Inner membrane	85.8
Periplasmic	79.1
Outer membrane	87.1
Extracellular	73.9
Overall accuracy	**83.2**

Table 3.11: The predictive accuracy for subcellular locations of Bayesian Classification Tree approach for Data_Gram.

Chapter 4

Ensemble Modeling

The main motivation for combining models in ensembles is to improve their predictive performance. An ensemble of classifiers is a set of classifiers whose individual predictions are combined in some way to classify new examples. There is a plethora of terms in the literature, such as committee, classifier fusion, aggregation to indicate sets of learning machines that work together to solve a machine learning problem. The variety of terms reflects the absence of a unified theory on ensemble methods and the youngness of this research area.

Empirical studies showed that classification ensembles are often much more accurate than the individual base learner that make them up. Many learning algorithms apply local optimization techniques that may get stuck in local optima (for instance greedy local optimization approach used by decision trees and gradient descent used by neural networks). Building an ensemble using, for instance, different starting points may achieve a better approximation.

The basic framework includes two parts: learning and application. In the *learning part*, an ensemble including base classifiers is generated. In the *application part*, the class predictions of the base classifiers need to be integrated in some way to produce the final classification. There are two main approaches to the integration: *combination approach*, where the final outcome is composed using the predictions of all base classifiers and *selection approach*, where one of the classifiers is selected and the final class predictions is the one produced by it. The two main methods for a *selection* (*competitive combination*) are:

- *Gating*, employed by Mixtures of Experts, which I handle in Chapter 6,
- *Rule-based switching*: In this case the switching between the models can be triggered on the basis of the output of one of the models. For example, in the stydy on the diagnosis of myocardial infarction (heart attack) in [Baxt, 1992], two models were optimized separately by varying the proportion of high risk and low risk patients in the training sets. The first model was trained to make as few positive errors as possible, and the second model trained to make as few negative errors as possible. The output of the first model was used unless it exceeds a threshold, in which case the output of the second model was used.

Sections 4.1, 4.2, 4.3, 4.4 provide background of the field without striving for completeness. The interested reader should see [Dietterich, 2000] for a broad survey of ensemble methodology.

In Section 4.5 I analyze the performance of pairwise classification as a general ensemble technique.

In Sections 4.6 and 4.7 I generate ensemble of localization predictors using different representations of protein sequences.

In the literature can be found the idea of training a *meta-classifier* to utilize the predictions of multiple *base-classifiers*. Voting is then used as a baseline method against which the learned combiners are compared. The rest of the Chapter focuses mostly on *meta-learning*.

One such meta-classification scheme is the family of *stacking* algorithms. The basic idea of stacking is to use the predictions of the original classifiers as attributes in a new training set that keeps the original class labels. Section 4.8 surveys some recent results in stacking.

A kind of meta-learning technique is the *arbiter meta-learning*. I discuss this technique and present the results of my experiments with it in Section 4.9.

In Section 4.10 I apply the idea of *grading*, another ensemble learning scheme, to my classification task.

In the last Section of this Chapter I employ an interesting classifier combination approach, which connects previously learned classifiers in kind of a decision tree using the validation of their performance.

4.1 Generation of ensembles

Ensembles can be generated using a single learning algorithm, such as decision tree learning or neural network training. Different classifiers are generated by

- manipulating the training set, as done in *bagging* and *boosting*. In the first, training sets are fully independent by bootstrapping, in the second they differ as a result of previous classification.

  Training sets may also be different if each of the classes is first split by a cluster analysis and then the classes are separated cluster by cluster.

  Another example is the set of two-class discriminants that may be used to solve a K-class problem. For the approaches of training each classifier on different parts of the data two further distinctions can be made. The first is whether the different parts of the data are chosen randomly or whether they are chosen to be regions that can be described by the attributes (modelled). The second distinction is whether the data partition is absolute (i.e. each classifier is trained on its own mutually exclusive part of the data) or whether they are overlapping).

- manipulating the input features, using different representations of data

- manipulating the output targets or

- injecting randomness in the learning algorithm.

[Zheng, 1998] proposes Naive Bayesian Classifier Committees. Each member of the committee is a naive Bayesian Classifier based on a subset of all the attributes available for the task.

Another approach is to generate base classifiers by applying different learning algorithms (with heterogeneous model representations) to a single data set.

4.2 Voting

The simplest method for combining classifiers is *voting*. Voting counts the number of predictions for each class in the vector of the responses of the base classifiers and predicts the most frequently predicted class. Several variations of voting have been proposed:

- **Unanimity.** The combined classifier decides that an input pattern x comes from class C_j if and only if all the classifiers decide that an x comes from class C_j, otherwise it rejects x.
- **Modified unanimity.** The combined classifier decides that an input pattern x comes from class C_j if some classifiers support that x belongs to C_j and no other classifier supports that x belongs to any other class (i.e. rejects x), otherwise it rejects x.
- **Weighted or unweighted majority.** The combined classifier decides that an observation x belongs to class C_j if more than half of the classifiers support that x belongs to C_j. A modification of this rule is to require a different proportion of classifiers to agree instead of half of them.
- **Thresholded plurality.** The combined classifier decides that an observation x belongs to class C_j if the number of classifiers that support it is considerably bigger than the number of classifiers that support any other class

Combining classifiers with voting is simple, but it has a drawback: it neglects the differences of skills of different base classifiers. The weight of the decision of all the classifiers is equal, even when some of the classifiers are much more accurate than others.

4.3 Popular ensemble methods: boosting, bagging, arcing

Best known ensemble methods are *bagging* (bootstrap aggregating) [Breiman, 1996a] and *boosting* [Freund and Schapire, 1996], which rely on learning a set of diverse base classifiers (typically by using different subsamples of the training set), whose predictions are then combined by simple voting.

The boosting procedure is as follows: a classifier is trained on a randomly chosen subset D_1 of the available training data. This classifier is then used to filter the remaining training data to produce a second training set D_2 for a second classifier. Flip a fair coin, if the coin comes up heads, examples are passed through the first classifier until it misclassifies a pattern. This pattern is then added to the second training set D_2. If the coin comes up tails, examples are passed through the first classifier until it correctly classifies a pattern. This pattern is then added to the

second training set D_2. This process is continued until enough patterns have been collected to train the second classifier.

After training the second classifier both classifiers are used to produce a third training set D_3 for a third classifier. The remaining training data is passed through the first two classifiers. If they disagree on the classification of a pattern it is added to a third training set D_3. If they agree the pattern is discarded. This process is continued until enough patterns have been collected to train the third classifier. The boosted classifiers are combined using either averaging or a voting scheme. The voting scheme works as follows: if the first two classifiers agree, their answer is used as the prediction, if they disagree, the answer of the third classifier is used as the prediction.

AdaBoost (Adaptive Boosting) creates a sequence of training sets and determines weights of the training instances, with higher weights for those that are incorrectly classified.

AdaBoost combined with probabilistic neural network algorithm was applied for protein subcellular localization in the recent paper of [Guo et al., 2004].

Boosting and Bagging applied to C4.5 were used in [Melville and Kokku] for the prediction of functional classification of genes using expression data and phylogenetic profiles.

The *arcing method* by [Breiman, 1996b] uses a simplified procedure for weighting of the training instances.

The *bagging* algorithm uses classifiers trained on bootstrap samples, created by randomly drawing a fixed number of training data instances from the pool which always contain all training instances. Results are aggregated by voting. [Breiman, 1996a] has shown that bagging nearest neighbor classifiers in general will not be effective, because they are *stable*: small perturbations in the training data will not change the hypothesis very much.

4.4 Classifier diversity and coverage

Important questions of ensemble modeling are: how many classifiers are required in an ensemble to obtain a desired accuracy and how many are needed for an improvement over the accuracy of the best single classifier?

Clearly, fewer classifiers is prefered, since training and application costs will be lower.

Experts which are very similar tend to provide redundant information, and the high level of their dependence means not only minimal gains from aggregation but also difficulties during the integration process. Thus, heterogeneity among experts is highly desirable.

Ideally, the classifiers should be *diverse*: each should work well on different parts of the given data set as no benefit arises from combining the predictions of a set of classifiers that all classify the same portion of the data correctly. A related objective is to maximize *coverage* of the data, which is the percentage of the data that at least one classifier can classify correctly. A proper training of base classifiers, it means here training avoiding overfitting, is important, as the performance of the base classifiers is not of primary importance, instead, sacrificing a small amount of accuracy in each classifier of the ensemble may result in increased coverage. See for the ilustration of this idea an example in [Brodley and Lane, 1996].

Achieving coverage greater than the accuracy of the best single classifier requires diversity among the classifiers.

[Wolpert, 1992] has suggested, that „one should try to find generalizers which behave very differently from one another, which are in some sense 'orthogonal', so that their guesses (predictions) are not synchronized".

How different the resulting classifiers are and especially how this should be measured is an open, but heavily studied topic [Kuncheva, 2003].

One metric for determining the similarity of the classification decisions of a set of learning algorithms is *classification overlap*, which requires counting the number of instances, that were classified the same way by each of the classifiers. A set of classifiers S_1 is more diverse than another set S_2 if:

$$overlap(S_1) < overlap(S_2).$$

[Ali and Pazzani, 1996] define diversity as the percent of test instances for which the classifiers make different predictions, but for which one of them is correct.

In [Tsymbal et al., 2002] another measure was used. A diversity of the classifier Cl_i and the whole ensemble Div_i is the average difference in the predictions on test instances of all the pairs of classifiers including Cl_i:

$$Div_i = \frac{\sum_{j=1}^{N} \sum_{k=1,k\neq i}^{K} Dif(Cl_i(x_j),Cl_k(x_j))}{N*(K-1)},$$

where $Cl_i(x_j)$ denotes the classification of the instance x_j by the classifier Cl_i, $Dif(a,b)$ is the difference in two classifications a and b, which is zero if classifications are the same and one if they are different, K denotes the number of classifiers and N is the number of data instances.

[Skalak, 1995] proposes to use the term *complementary* component classifiers rather than *dissimilar*. The author warns of choosing dissimilar classifiers without regard to whether they will actually work well together in a composite classifier. Consider an analogy to a basketball team. Team members may be selected according to some criterion of dissimilarity. They may be from different states or of different races. But if they are chosen so that together they should make a winning team, then a good play-maker, a good outside shooter and a good rebounder will be chosen as part of the team.

4.5 Pairwise classification as an ensemble technique

Pairwise classification is a technique for turning multi-class problems into two-class problems. There are two most popular approaches: *one-against-one* and *one-against-rest* approach.

One-against-rest approach uses the examples of the corresponding class as positive examples and all the others as negative examples.

One-against-one (or *round robin*) approach (see [F¨urnkranz, 2002a] and [F¨urnkranz, 2002b]) learns one classifier for each pair of classes, using only training examples for these two classes and ignoring all others. There are total $K*(K-1)/2$ classifiers for the K-class problem. For example we have 55 classifiers for the 11-class problem of location prediction for DATA_SWISS.

A new example is classified by submitting it to each of these classifiers and combining their predictions via simple voting. This will often result in poor classification performance. Assume pattern x comes from class C_j. There will be $(K-1)*(K-2)/2$ classifiers which have never seen objects from this class. Combining this *ignorant classifiers* will therefore result in almost random classification. This becomes even more prominent for larger number of classes. For a classification problem with more than 4 classes the majority of the classifiers are ignorant of class C_j, and for a 10-class problem even 80%.

Nevertheless, in [F¨urnkranz, 2002b] it was shown that the use of *round robin* ensembles can increase the classification performance of decision tree learners, even though they can directly handle multi-class problems.

I learned 6 pairwise Bayesian classifiers with eukaryotic data and combined their results with plurality voting scheme. Tables 4.1 and 4.2 report the results. Compared to the results of single global Bayesian Classification procedure (see Table 2.8), there is no improvements on overall accuracy and also on accuracies for single locations.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**536**	29	63	56	684
Extracellular	62	**199**	35	29	325
Mitochondrial	120	13	**167**	21	321
Nuclear	171	43	73	**810**	1097
Sum	889	284	338	916	2427

Table 4.1: Confusion matrix of prediction results of *round-robin* procedure for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	78.4
Extracellular	61.2
Mitochondrial	52
Nuclear	73.8
Overall accuracy	**70.5**

Table 4.2: The predictive accuracy for subcellular locations of *round-robin* procedure for Data_Euk.

In [F¨urnkranz, 2002b] the author argues that the most pressing issue for further research in pairwise classification is an investigation of the effects of different voting schemes.

I have conducted the following experiment. I allowed a classifier only to vote for a class if it has a certain minimum confidence in its prediction. As I work with probabilistic Bayesian classifiers and I have the posterior probabilities for each class, I allow a classifier only to vote for a class if the posterior probability of this class is bigger than a certain threshold. Otherwise, it rejects an example (gives out „unknown“ class). We become so the *cautious classifiers*, which I treat also in Chapter 7.

Furthermore, I ordered the classes so that the classes, which are underrepresented, are ranked higher, and I give the preference to the higher ranked classes in case of a tie.

Tables 4.3 and 4.4 show the results of this experiment. Note from Table 4.3, that 5 sequences were classified as „unknown“. Although the drop in overall accuracy is observed, we can state the increase of the prediction accuracy for two underrepresented classes (Mitochondrial and Extracellular).

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**456**	44	133	49	684
Extracellular	46	**201**	50	25	325
Mitochondrial	79	11	**219**	12	321
Nuclear	151	55	112	**779**	1097
Sum	732	311	514	865	2422 $\backslash 2427$

Table 4.3: Confusion matrix of prediction results of *round-robin* procedure with *cautious* pairwise classifiers and ordered classes for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	66.7
Extracellular	61.8
Mitochondrial	68.2
Nuclear	71
Overall accuracy	**68.2**

Table 4.4: The predictive accuracy for subcellular locations of *round-robin* procedure with *cautious* pairwise classifiers and ordered classes for Data_Euk.

4.6 Combining localization predictors learned with reduced amino acid alphabet

Since there are some similarities between amino acids, the amino acid alphabet can be simplified. The amino acids can be regrouped into smaller groups of classes according to their physicochemical or structural properties.

[Yu et al., 2004] uses the reduced 3-peptide amino acid composition in which 20 amino acids are classified into four groups (charged, polar, aromatic, nonpolar). This helps to overcome the size problem for n-peptide composition when n gets larger.

The work of [Yu et al., 2004] is an example of using the idea of ensemble of classifiers for predicting subcellular localization of proteins. The authors used Support Vector Machines

(SVMs) and because the SVM separates two classes, for multiclass classification the combination of one-against-one classifiers should be used. For 5 classes of Gram-negative bacteria, $5(5-1)/2 = 10$ SVM classifiers should be constructed. Also the four sequence coding schemes: A_1, A_2, X_4 and F_3X_5 were used. The notation A_n denotes the n-peptide composition of amino acids, F_n denotes the n-peptide composition with the alphabet reduced to the four letters (each letter for the group of amino acid), X_k denotes the partitioned composition of amino acids in which the sequence is partitioned into k regions of equal length. Therefore, $10*4 = 40$ classifiers were constructed. The votes from these classifiers were combined with the jury voting scheme to determine the final assignment. In the case of identical votes, more weights were given to the votes from A_1. The same method was applied for predicting protein three-dimensional folds in [Yu et al., 2003].

I used the reduced alphabet of amino acids taken from [Li et al., 2003]. These groupings of residues are biologically relevant in various aspects and similar to some other results. Figure 4.1 shows the groupings that are organized in a hierarchical manner.

2	CMLIVWFY								AGTSPNQDEHRK											
3	CMLIV					WFY			AGTSPNQDEHRK											
4	CMLIV					WFY			AGTSP					NQDEHRK						
5	C	MLIV				WFY			AGTSP					NQDEHRK						
6	C	MLIV				W	FY		AGTSP					NQDEHRK						
7	C	MLIV				W	FY		AGTS				P	NQDEHRK						
8	C	MLIV				W	FY		AGTS				P	H	NQDERK					
9	C	MLIV				W	FY		AGTS				P	H	ND		QERK			
10	C	MLIV				W	FY		G	ATS			P	H	ND		QERK			
11	C	MLIV				W	FY		G	ATS			P	H	ND		QE		RK	
12	C	MLIV				W	FY		G	ATS			P	H	N	D	QE		RK	
13	C	ML		IV		W	FY		G	ATS			P	H	N	D	QE		RK	
14	C	ML		IV		W	FY		G	AS		T	P	H	N	D	QE		RK	
15	C	ML		IV		W	F	Y	G	AS		T	P	H	N	D	QE		RK	
16	C	ML		IV		W	F	Y	G	A	S	T	P	H	N	D	QE		RK	
17	C	ML		IV		W	F	Y	G	A	S	T	P	H	N	D	Q	E	RK	
18	C	ML		IV		W	F	Y	G	A	S	T	P	H	N	D	Q	E	R	K
19	C	M	L	IV		W	F	Y	G	A	S	T	P	H	N	D	Q	E	R	K
20	C	M	L	I	V	W	F	Y	G	A	S	T	P	H	N	D	Q	E	R	K

Figure 4.1: The groupings of 20 kinds of amino acids from $n = 2$ to $n = 20$, where n is the number of groups.

I explore the effect of reduction of alphabet size for the Markov Chains and found that it does not hurt Bayesian Classification and classification with Bayesian Classification Trees so much, if the alphabet will be reduced to the length of 12. Table 4.5 shows the results of BC and BCT procedures used with Markov Chains learned with reduced alphabet of 12 amino acid groups for eukaryotic data. Compare them with the results shown in Table 2.8 and Table 3.3.

Cellular location	BC Accuracy (%)	BCT Accuracy (%)
Cytoplasmic	75.6	73.8
Extracellular	52.0	79.7
Mitochondrial	51.7	54.8
Nuclear	73.5	85.3
Overall accuracy	**68.3**	**77.3**

Table 4.5: The predictive accuracy for subcellular locations of BC and BCT approaches used with reduced alphabet of 12 amino acid groups for Data_Euk.

Tables 4.6 and 4.7 report the results of voting combination of 3 Bayesian classifiers: one learned with Markov Chain Model, one learned with Markov Chain Model with the reduced alphabet and one learned with Multinomial Model. No improvement on the overall accuracy of single Bayesian classifier can be observed.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**521**	42	64	51	684
Extracellular	58	**189**	33	43	325
Mitochondrial	111	18	**167**	17	321
Nuclear	147	55	83	**797**	1097
Sum	837	304	347	908	2427

Table 4.6: Confusion matrix of prediction results of voting combination of 3 Bayesian classifiers for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	76.2
Extracellular	58.2
Mitochondrial	52.0
Nuclear	72.7
Overall accuracy	**69**

Table 4.7: The predictive accuracy for subcellular locations of voting combination of 3 Bayesian classifiers for Data_Euk.

4.7 Combining localization predictors learned from different sequence regions

In this Section I report the results of the following experiment: the sequences were partitioned into 3 regions of equal length. 3 Bayesian classifiers were learned from the corresponding sequence regions and then combined with simple voting. The results for eukaryotic data are summarized in Table 4.8 and Table 4.9. The reached overall accuracy of 74.3% compared with the 70.8% of single Bayesian classifier learned from the whole sequence length (see Table 2.8) demonstrates the potential usefulness of this approach.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**530**	20	44	90	684
Extracellular	49	**206**	21	49	325
Mitochondrial	99	12	**158**	52	321
Nuclear	138	20	30	**909**	1097
Sum	816	258	253	1100	2427

Table 4.8: Confusion matrix of prediction results of combining classifiers learned from 3 sequence regions for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	77.5
Extracellular	63.4
Mitochondrial	49.2
Nuclear	82.9
Overall accuracy	**74.3**

Table 4.9: The predictive accuracy for subcellular locations of combining classifiers learned from 3 sequence regions for Data_Euk.

4.8 Stacking

Learning ensembles is a useful method for reducing error on a test set. Simple cross validation schemes choose only the best model on the validation set, which corresponds to a winner-take all strategy. An alternative approach is to use a combination of models trained on different validation sets.

[Wolpert, 1992] suggested to use cross-validation to combine models rather than choose between them and extended this idea under the name of *stacked generalization*.

The basic idea of stacking is to train I models on a training set D, leaving aside a section of the training set for validation. Let us assume that only one pattern was leaved aside for validation.

Each model is evaluated on this pattern and each produces an output. The set of I outputs of the models and the target label of the pattern is then added as a new training example to a new training set D_2. This process is repeated for all patterns in the training set D, so that a new training set D_2 is generated. This set is used to train *level 1* model, which learns to map from the outputs of the models trained on different partition of the training set, to the target output. At the end of this process the *level 0* models are retrained using the full data set D. Predictions are made by feeding query data to the *level 0* models and feeding their outputs to the *level 1* model.

The general framework of combining the predictions of multiple classifiers via a separate, trainable classifier is commonly referred to as *stacking*. Since for the training set we have both the predictions of the based learners (*base-level classifiers*) and the true class, we can train a *meta-level classifier*. Figure 4.2 depicts a stacked classifier architecture.

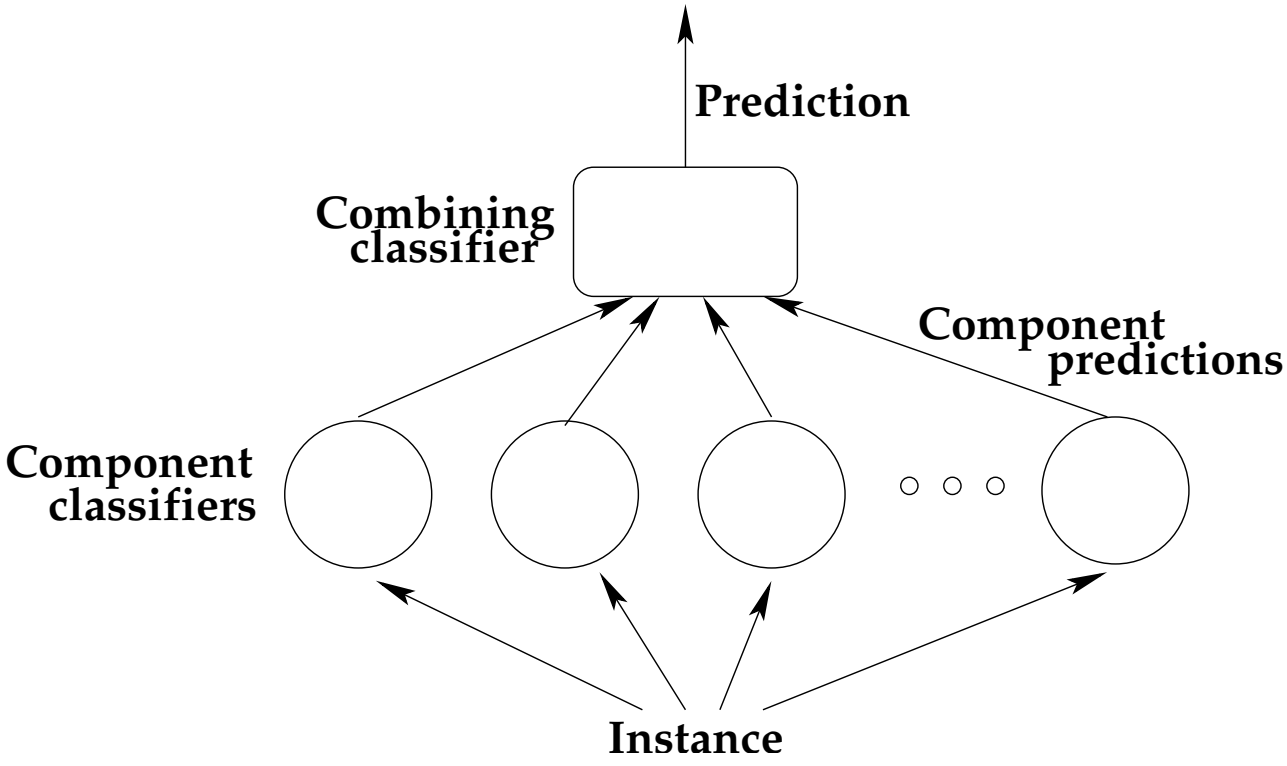

Figure 4.2: Stacked classifier architecture.

The most important issues in stacking are the choice of the features and the algorithm for learning at the meta-level.

[Ting and Witten, 1999] use base-level classifiers whose predictions are probability distributions over the set of class values, rather than single class values. The meta-level attributes are thus the probabilities of each of the class values returned by each of the base-level classifiers. Multi-response linear regression (MLR) is recommended by the authors for meta-level learning.

[Merz, 1999] proposes a stacking method called SCANN that uses correspondence analysis to detect correlations between the predictions of base-level classifiers. The original meta-level feature space (the class value predictions) is transformed to remove the dependencies and a nearest-neighbor method is used as the meta-level classifier on this new feature space.

[Todorovski and Dzeroski, 2000] introduce a new meta-level learning method called *meta decision trees* (MDTs). Properties of the probability distributions predicted by the base-level classifiers (such as entropy and maximum probability) are used as meta-level attributes, rather than the distributions themselves. These properties reflect the confidence of the base-level classifiers and give rise to very small MDTs, which can be inspected and interpreted. In a leaf node, a MDT predicts which classifier is to be used for classification of an example, instead of predicting the class value of the example directly.

Cascading [Gama and Brazdil, 2000] is another related variant where the classifiers are applied in chain and there is no level 1 classifier. Each base classifier, when applied to the data, adds his class probability distribution to the data and returns this augmented data set, which is to be used by the next classifier. Thus, the order in which the classifiers are executed becomes important. Furthermore, cascading increases the dimensionality of the data set with each step.

I experimented with combining the results of different Bayesian classifiers (learned on a randomly chosen subsets of data, learned with different models and alphabets) with Naive Bayes. But this stacked generalizer worked not better as the single Bayesian classifier.

4.9 Arbitration

An *arbiter* is a classifier that is trained to resolve disagreements between the base classifiers (see e.g. [Chan and Stolfo, 1995] and [Tsymbal et al., 1999]). An arbiter is generated using the same learning algorithm that is used to train the base classifiers. In the arbiter technique, the training set for the arbiter is subset of the union of the training sets for the base classifiers. The choice of examples picked for the training set for the arbiter is dictated by a *selection rule*. One version of the selection rule is as follows:

An instance is selected if none of the classes in the K base predictions gathers a majority vote ($> K/2$ votes).

The purpose of this rule is to choose examples that are confusing, i.e. the majority of classifiers do not agree.

In the prediction phase an *arbitration rule* decides a final classification outcome based upon the base predictions and the classification predicted by the arbiter. One arbitration rule is as follows:

Return the class with a plurality of votes with preference given to the arbiter‘s choice in case of a tie.

The generation of an arbiter has much in common with the boosting technique that also filters training instances to train the base classifiers. The approach can be considered as a particular case of the stacked generalization framework that integrates the results of the base classifiers by a trained meta-level classifier.

I evaluated empirically the arbiter meta-learning technique. Combining of classifiers, trained on two training data subsets with Bayesian classification procedure, and arbiter, works slightly better as the single Bayesian classifier, learned on the whole training data. Tables 4.10, 4.11 and 4.12, 4.13 show the results for Data_Euk and Data_Gram data sets.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**520**	26	71	67	684
Extracellular	53	**208**	29	35	325
Mitochondrial	109	11	**160**	41	321
Nuclear	147	29	65	**856**	1097
Sum	829	274	325	999	2427

Table 4.10: Confusion matrix of prediction results of *arbiter meta-learning* procedure for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	76.0
Extracellular	64.0
Mitochondrial	49.8
Nuclear	78.0
Overall accuracy	**71.9**

Table 4.11: The predictive accuracy for subcellular locations of *arbiter meta-learning* procedure for Data_Euk.

4.10 Grading

One can assume that each classifier has a particular subdomain for which it is most reliable, and so a description of this area of expertise can be learned.

[Seewald and F¨urnkranz, 2001] propose a method called *grading* that learns a meta-level classifier for each base-level classifier. The meta-level classifier predicts whether the base-level classifier is to be trusted, i.e. whether its prediction will be correct. The base-level attributes are used also as meta-level attributes, while the meta-level outputs are + (correct) and - (incorrect). In the prediction phase only the predictions of base-level classifiers that are predicted to be correct are taken and combined by summing up the probability distributions.

	Predicted group					
	Cytoplasm	Inner membrane	Periplasm	Outer membrane	Extracell	Sum
Cytoplasmic	**244**	8	32	3	6	293
Inner membrane	34	**272**	14	10	1	331
Periplasmic	30	12	**222**	28	15	307
Outer membrane	37	19	31	**305**	62	454
Extracellular	10	9	18	39	**130**	206
Sum	355	320	317	385	214	1591

Table 4.12: Confusion matrix of prediction results of *arbiter meta-learning* procedure for Data_Gram.

Cellular location	Accuracy (%)
Cytoplasmic	83.3
Inner membrane	82.2
Periplasmic	72.3
Outer membrane	67.2
Extracellular	63.1
Overall accuracy	**79.0**

Table 4.13: The predictive accuracy for subcellular locations of *arbiter meta-learning* procedure for Data_Gram.

Grading is very similar to the approach introduced by [Bay, 2000]. They also train a classifier to learn whether a classification is reliable or not. However, they did not use this approach for decision making, but instead aimed at providing insight about the domain regions in which a learner is not able to discriminate well. The negative feedback- when the meta classifier predicts that the base classifier is wrong- only rules out the class predicted by the base classifier, but does not help to choose among the remaining classes (except, for two-class problems).

The main differences between grading and stacking (or combiners) is that grading does not change the original input data- by replacing it with class predictions or class probabilities (or adding them to it)- but instead modifies the class values.

The approach of [Ortega 2001] builds a *referee* predictor for each of the component classifiers. The final classification is that returned by the component classifier whose correctness can be trusted the most, according to a confidence level provided by the *referees*. It may not be the case that those regions of the example space where base classifier is reliable can be simply described by the input data. [Ortega 2001] proposes to use intermediate subconcepts of the base classifier as features for the induction of *referees*. This is the point where approach of [Ortega 2001] differs from grading. The modular architecture of the approach is depicted in Figure 4.3.

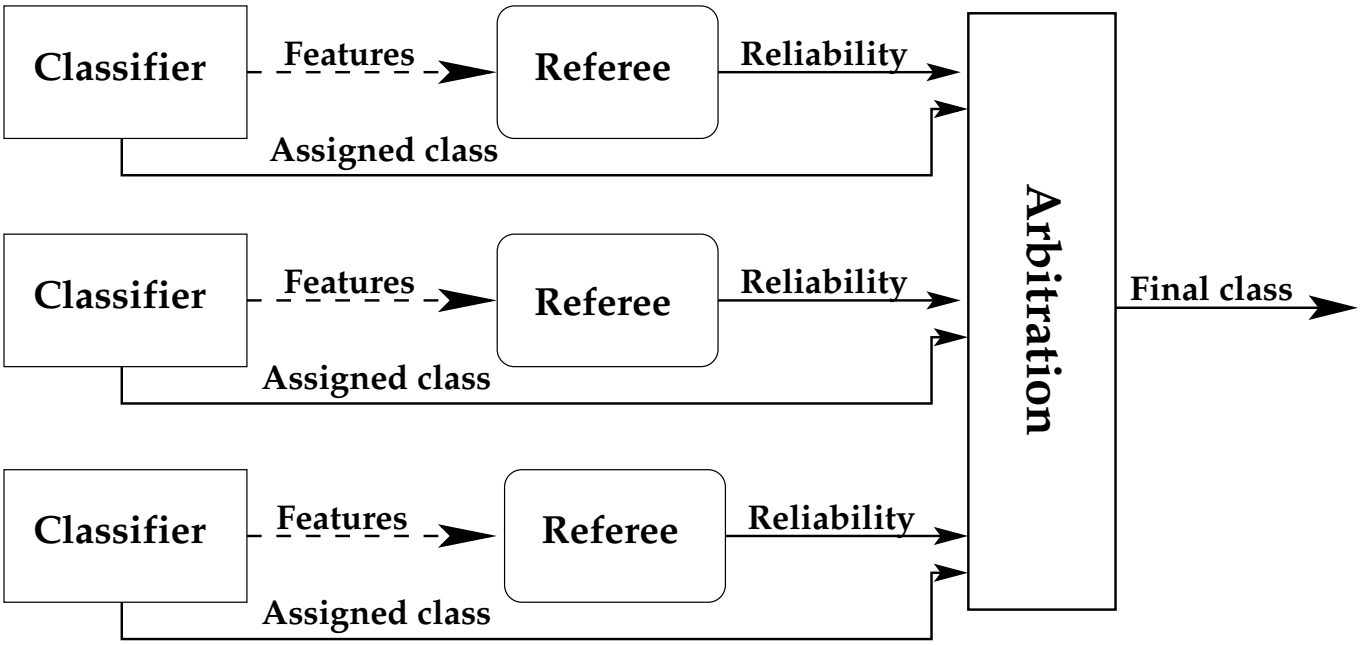

Figure 4.3: The modular architecture of the approach of [Ortega 2001].

I evaluated the grading approach as it is proposed in [Seewald and F¨urnkranz, 2001] on my classification task using Bayesian classification learning procedure for learning of base and meta classifiers for each class. Tables 4.14-4.17 summarize the empirical results. I observe that grading does not help in improving overall classification accuracy compared to single Bayesian classifier.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**522**	34	72	56	684
Extracellular	60	**196**	35	34	325
Mitochondrial	116	13	**172**	20	321
Nuclear	156	49	86	**806**	1097
Sum	854	292	365	916	2427

Table 4.14: Confusion matrix of prediction results of *grading* approach for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	76.3
Extracellular	60.3
Mitochondrial	53.6
Nuclear	73.5
Overall accuracy	**69.9**

Table 4.15: The predictive accuracy for subcellular locations of *grading* approach for Data_Euk.

	Predicted group				
	Cytoplasmic	Plasma membrane	Mitochondrial	Other	Sum
Cytoplasmic	**41**	2	0	0	43
Plasma membrane	3	**23**	3	1	30
Mitochondrial	1	0	**11**	0	12
Other	3	0	3	**8**	12
Sum	48	25	15	9	97

Table 4.16: Confusion matrix of prediction results of *grading* approach for Data_Apoptosis.

4.11 Decision tree of classifiers

An algorithm that forms a decision tree of classifiers by ordering the classifiers in order of their performance on the training data was outlined in [Garg and Pavlovic, 2002].

Cellular location	Accuracy (%)
Cytoplasmic	95.3
Plasma membrane	76.7
Mitochondrial	91.7
Other	66.7
Overall accuracy	**85.6**

Table 4.17: The predictive accuracy for subcellular locations of *grading* approach for Data_Apoptosis.

Let $h_1, h_2, \ldots, h_M$ be the set of M classifiers. A decision tree, which has these classifiers as its nodes and the value of the leaf represents the output, is constructed as follows.

Let $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be the set of training samples, where $y_i \in \{c_1, \ldots, c_K\}$ (a set of K classes). The first step is to evaluate each classifier on the training samples. For each classifier, K different errors are measured:

$$e_j^m = P(y_i = c_k, k \neq j | h_m(x_i) = c_j)$$

The classifiers are ordered based on their performance on the training data. The classifier with the lowest error is picked to form the root node of the decision tree. If e_j^m is the lowest error, then h_m forms the top node of the decision tree. The output of the decision tree is c_j whenever h_m says c_j, else the output is based on the decision made by the right subchild of the root node. The left child of the top node is a leaf with value c_j. The right node points to a decision tree made of the classifiers. To obtain the classifier that forms the root node of the right subtree, the algorithm is repeated. However, this time the sample set is $\tilde{D} = \{(x_i, y_i)\}$ for all i, such that $h_m(x_i) \neq c_j$. This process is repeated until all the classifiers are incorporated in the decision tree.

I tried to combine with this procedure 2 classifiers- one learned with BC approach and one with BCT approach- in order to improve on the results of BCT (the best of these two) on eukaryotic data. Tables 4.18 and 4.19 demonstrate the results. However, no synergethic effect of this combination can be observed. The reason for this can be in the reduction of the available training data due to the necessity of a separate validation set and as a consequence the reduction of the quality of the base learners.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**506**	25	68	85	684
Extracellular	35	**238**	17	35	325
Mitochondrial	98	9	**167**	47	321
Nuclear	80	26	37	**954**	1097
Sum	719	298	289	1121	2427

Table 4.18: Confusion matrix of prediction results of *DT of classifiers* procedure for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	74
Extracellular	73.2
Mitochondrial	52
Nuclear	87
Overall accuracy	**76.8**

Table 4.19: The predictive accuracy for subcellular locations of *DT of classifiers* procedure for Data_Euk.

Chapter 5

Growing Classification Trees using mixture modeling

A meta-level classifier should be capable of identifying the areas of expertise of its base-level classifiers and following the best expert for each new item of discussion. If a meta-level classifier is inaccurate, it will be incapable of exploiting the full data coverage provided by base-classifiers. [Brodley and Lane, 1996] argue that the increasing coverage through diversity is not enough to ensure increased prediction accuracy- if the integration method does not utilize the coverage.

In this chapter I will consider methods that explicitly assign a different classifier for each mutually exclusive subset of the data, so the regions of expertise are defined before learning the base-level classifiers rather than afterwards. The subsets of instances are typically defined using models.

I use an unsupervised approach called *clustering* to find the areas in training data. In the next section I give some background information on clustering, especially on model-based clustering of sequences. Then I give a short introduction to EM algorithm, which helps to identify the distributions in the data and to classify the data points based on the probability to belong to these distributions. I also handle the problem of finding the number of clusters in the data. In section 5.4 I present my algorithm, followed by the section with the results.

5.1 Model-based clustering

Clustering algorithms based on probabilistic models offer a principled alternative to distance-based algorithms. Data is viewed as sample from a population that consists of a number of subpopulations (clusters or components). Each cluster can be described by a probability distribution.

Consider a data set consisting of N sequences, $D = \{s_1, \ldots, s_N\}$. The problem addressed in this section is the discovery of a natural grouping of the sequences into K clusters. This is analogue to clustering in multivariate feature space which is normally handled by methods such as *k-means* and Gaussian mixtures. Here, however, we are trying to cluster the sequences rather than the feature vectors.

A probabilistic model for this problem is that of a finite mixture model.

$$p(s) = \sum_{j=1}^{K} \alpha_j p_j(s|\theta_j),$$

where s denotes a sequence, α_j is the weight of the jth model, and $p_j(s|\theta_j)$ is the density function for the sequence s given the component model p_j with parameters θ_j). When we assume that the p_js are Markov Chains, then the θ_js are the transition matrices and initial state probabilities, all for the js component. There is no crossover in this mixture model. Data is assumed to come from one component or the other.

We are interested in learning the *maximum-likelihood* (ML) or *maximum a posteriori* (MAP) parameter estimates given the data D, i.e.

$$\theta_{ML} = argmax_\theta\{p(D|\theta)\}$$

and

$$\theta_{MAP} = argmax_\theta\{p(D|\theta)p(\theta)\},$$

where under the usual assumption that data instances are conditionally independent given the underlying model, we have

$$p(D|\theta) = \prod_{i=1}^{N} p(s_i|\theta),$$

known as the *likelihood*.

I fit mixture model to the data with the help of **Expectation Maximization algorithm** (EM).

5.1.1 Related work

The concept of using a generative model for clustering non-vector data has been independently pursued in several different contexts. [Poulsen, 1990] introduced a particular form of Markov mixtures and an EM algorithm for modeling heterogeneous behavior in consumer purchasing data. More general versions of Markov mixtures were developed by [Smyth, 1997], [Smyth, 1999] and [Ridgeway, 1997].

The use of Hidden Markov Models for clustering sequences was used in the context of discovering subfamilies of proteins in [Krogh et al., 1994].

5.2 Parameter estimation and the EM algorithm

In this section I provide a general description of an EM algorithm that can be applied to mixture-model clustering. The name to the algorithm was given by [Dempster et al., 1977] in their fundamental paper. Complete proofs can be found in the excellent tutorial of [Bilmes,].

The EM algorithm is a general technique for finding maximum likelihood (ML) or maximum a posteriori (MAP) parameters when some aspect of the data is considered as missing. In a

mixture context, the missing data consists of the cluster labels for each individual: if we knew these labels then parameter estimation would be quite straightforward.

The EM algorithm operates in two re-estimation steps.

In the E-step one calculates the posterior probabilities of the unobserved clustering variables $p(c_i|s_i, \theta)$ for each data sample under each of the K cluster models using the current value of the parameters.

In the M-step one updates parameters by weighting each data sample according to its class-conditional probability.

That leads to a sequence of parameter settings, which results in non-decreasing likelihood (or posterior probability), which is guaranteed to find at least a local maximum of the ML or MAP objective function.

Once we define a proper likelihood over the data of interest, then specification of any specific EM algorithm (for any particular type of models) follows in a direct manner from the general principles of EM.

5.2.1 EM algorithm for mixture of Markov Chains

Let $D = \{s_1, \ldots, s_N\}$ denote the sequences to be clustered. It is assumed that each sequence is assigned to one of the clusters $C = \{c_1, \ldots, c_K\}$ with some probability $P(c)$ and $\sum_c P(c) = 1$. The number of clusters K is fixed.

Let $n(s, a_i, a_j)$ indicate the observed frequency of occurring of the amino acid a_j after the amino acid a_i in the sequence s.

$$n(s, a_i) = \sum_{j=1}^{20} n(s, a_i, a_j)$$

Introducing class-conditional transition probabilities $P^c(a_j|a_i)$ and class prior probabilities $P(c)$ (stacked in a parameter vector θ) the likelihood for the mixture model is defined by:

$$L(\theta|D) = \prod_s \sum_c P(c)P(s|c_s = c; \theta),$$

where

$$P(s|c_s = c; \theta) = \prod_{i,j} P^c(a_j|a_i)^{n(s,a_i,a_j)}.$$

The two steps of EM algorithm for the model take the form:

E-step:

$$P(c_s = c|s; \theta^t) = \frac{P(c)P(s|c_s=c;\theta^t)}{P(s|\theta^t)} = \frac{P(c)\prod_{i,j} P^c(a_j|a_i)^{n(s,a_i,a_j)}}{\sum_{c'} P(c')\prod_{i,j} P^c(a_j|a_i)^{n(s,a_i,a_j)}}$$

M-step:

$$P(c) = \frac{1}{N}\sum_s P(c_s = c|s; \theta^t)$$

$$P^c(a_j|a_i) = \frac{\sum_s P(c_s=c|s;\theta^t)n(s,a_i,a_j)}{\sum_s P(c_s=c|s;\theta^t)n(s,a_i)}.$$

5.2.2 EM algorithm for mixture of Multinomial Models

Amino acid frequencies are encoded using count variables $n(s,a)$, which indicate how often an amino acid a occured in a sequence s; $n(s) = \sum_a n(s,a)$ denotes the length of the sequence.

Introducing class-conditional amino acid distributions $P(a|c)$ and class prior probabilities $P(c)$ (stacked in a parameter vector θ) the likelihood for the mixture model is defined by

$$L(\theta|D) = \prod_s \sum_c P(c)P(s|c_s = c;\theta),$$

where

$$P(s|c_s = c;\theta) = \prod_a P(a|c)^{n(s,a)}.$$

For this model the two steps of EM algorithm take the form:

E-step:

$$P(c_s = c|s;\theta^t) = \frac{P(c)P(s|c_s=c;\theta^t)}{P(s|\theta^t)} = \frac{P(c)\prod_a P(a|c)^{n(s,a)}}{\sum_{c'} P(c')\prod_a P(a|c')^{n(s,a)}}$$

M-step:

$$P(c) = \frac{1}{N}\sum_s P(c_s = c|s;\theta^t)$$

$$P(a|c) = \frac{\sum_s P(c_s=c|s;\theta^t)n(s,a)}{\sum_s P(c_s=c|s;\theta^t)n(s)}.$$

Since the EM algorithm is hill-climbing the likelihood surface, the quality of the final solution can depend critically on the initial conditions.

5.3 Learning the number of mixture components

The problem of learning the best value for K in a mixture model is a difficult one in practice even for the case of Gaussian mixtures. There has been considerable prior work on this problem. Penalized likelihood approaches are popular, such as Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC), where the log-likelihood on the training data is penalized by the substraction of a complexity term.

[Smyth, 1997] investigated the use of Monte-Carlo cross-validation approach for determining the number of clusters K from the data.

Let $L_K(D^{test})$ be the log-likelihood, where the model with K components is fitted to the training data D, but the likelihood is evaluated on D^{test}. We can view this likelihood as a function of a parameter K. Maximizing out-of-sample likelihood over K has been shown to be a reasonable model selection strategy, more robust then penalized likelihood methods.

In Monte-Carlo cross-validation approach the data is partitioned into a fraction β for testing and $1-\beta$ for training and this procedure is repeated M times where the partitions are randomly chosen on each run (i.e. need not be disjoint). For the mixture clustering problem $\beta = 0.5$ was found empirically to work well.

5.4 Growing Classification Trees

In this section I propose an algorithm, which can also be seen as a decision-tree hybrid. The main idea is to fit a clustering model in every internal node of the tree and to partition the data at this node into the corresponding groups. The leaves contain Bayesian classifiers.

Top-down methods require a procedure to determine whether the classifier is optimal for a given space (or subspace) or whether the space should be splitted into a new set of subspaces. I use 5-fold cross-validation procedure to assess the quality of the classifier on the leaf and split the leaf, if the quality is below Threshold. The final procedure is as follows:

Input: a set of labelled instances.

- If the quality of the classifier learned on the training examples at the current node is above *Thresh*, create a leaf node with this classifier on it.
- Otherwise, cluster the training examples;
- Let K- number of clusters found at the node. If $K = 1$, create a leaf node with the classifier on it.
- Otherwise, create K child nodes and assign the examples to the corresponding child nodes;
- for each child node, call the algorithm recursively.

5.4.1 Related work

In this connection the work of [Kohavi, 1996] is very interesting, where the author proposes an algorithm, NBTree, which induces a hybrid of decision-tree classifiers and Naive-Bayes classifiers: the decision-tree nodes contain univariate splits as regular decision-trees, but the leaves contain Naive-Bayes classifiers.

The *utility of the node* is 5-fold cross-validation accuracy estimate of using Naive-Bayes classifier at the node. The *utility of a split* is the weighted sum of the utility of the nodes, where the weight given to a node is proportional to the number of instances that go down to that node. A split is defined to be *significant* if the relative (not absolute) reduction in error is greater than 5% and there are at least 30 instances in the node.

In [Seewald et al., 2000] a C4.5-style learner is introduced with alternative leaf models (Naive Bayes, IB1 (nearest neighbour algorithm) and multi-response linear regression), which can replace the original C4.5 leaf nodes during reduced error post-pruning. The authors consider their hybrid approach as a step towards the construction of learners that locally optimize their bias for different regions of the instance space.

The method I propose here resembles also Utgoffs **Perceptron trees** [Utgoff, 1988], in which each decision node contains an attribute test, and each leaf node contains a **Linear Threshold**

Unit (LTU). The associated learning algorithm is called by the author the *perceptron tree error correction procedure* and it is an incremental algorithm. It uses the following splitting criteria: if the space of instances at a node is not *linearly separable*, then it is necessary that the space be splitted (partitioned) into subspaces.

[Torgo, 1999] proposed the usage of kernel regressors in the leaves of regression tree. In this work a good performance level is achieved with much smaller trees than if kernel regression was not used.

5.5 Results and discussion

I have stated that this procedure works comparable to Bayesian Classification Tree procedure described in Section 3, only slightly worser. Tables 5.1, 5.2 and 5.3, 5.4 show the results of the experiments with Data_Euk and Data_Apoptosis.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**493**	13	70	108	684
Extracellular	33	**241**	7	44	325
Mitochondrial	87	5	**162**	67	321
Nuclear	79	11	21	**986**	1097
Sum	692	270	260	1205	2427

Table 5.1: Confusion matrix of prediction results of Growing Classification Tree approach for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	72.1
Extracellular	74.2
Mitochondrial	50.5
Nuclear	89.9
Overall accuracy	**77.5**

Table 5.2: The predictive accuracy for subcellular locations of Growing Classification Tree approach for Data_Euk.

	Predicted group				
	Cytoplasmic	Plasma membrane	Mitochondrial	Other	Sum
Cytoplasmic	**40**	0	1	2	43
Plasma membrane	0	**27**	3	0	30
Mitochondrial	1	0	**11**	0	12
Other	3	0	1	**8**	12
Sum	44	27	16	10	97

Table 5.3: Confusion matrix of prediction results of Growing Classification Tree approach for Data_Apoptosis.

Cellular location	Accuracy (%)
Cytoplasmic	93
Plasma membrane	90
Mitochondrial	91.7
Other	66.7
Overall accuracy	**88.7**

Table 5.4: The predictive accuracy for subcellular locations of Growing Classification Tree approach for Data_Apoptosis.

I have also conducted another kind of experiment. I implemented the algorithm, where I used classifiers at each terminal node, but have not used clustering at each decision node, instead the classification like in Bayesian Classification Tree procedure described in Section 3. The results of this experiment with Data_SWISS are shown in Tables 5.5 and 5.6. Note, that the overall accuracy is even higher than achieved with Bayesian Classification Tree procedure (compare with the Table 3.7).

	Predicted group											
	Chlor	Cytop	Cytos	End	Ext	Gol	Lys	Mit	Nuc	Per	Vac	Sum
Chloroplast	**853**	80	1	6	34	0	3	98	61	5	0	1141
Cytoplasm	68	**1803**	5	11	136	3	9	125	260	16	1	2437
Cytoskeleton	0	10	**8**	0	0	0	0	0	6	0	0	24
Endoplasmic	5	10	0	**91**	7	0	3	5	10	0	1	132
Extracellular	45	168	0	9	**3578**	10	25	63	244	12	11	4165
Golgi	0	8	0	0	6	**6**	1	1	10	0	0	32
Lysosome	0	10	0	0	16	0	**102**	1	1	1	0	131
Mitochondria	82	141	0	7	66	0	3	**693**	97	11	0	1100
Nuclear	41	271	2	12	135	5	7	62	**2787**	3	1	3326
Peroxisome	5	18	0	1	4	0	2	15	3	**74**	0	122
Vacuole	1	3	0	1	15	0	2	4	3	0	**24**	53
Sum	1100	2522	16	138	3997	24	157	1067	3482	122	38	12663

Table 5.5: Confusion matrix of prediction results of Growing Classification Tree approach for Data_SWISS.

Cellular location	Accuracy (%)
Chloroplast	74.8
Cytoplasm	74
Cytoskeleton	33.3
Endoplasmic ret	68.9
Extracellular	85.9
Golgi apparatus	18.8
Lysosome	77.9
Mitochondria	63.0
Nuclear	83.8
Peroxisome	60.7
Vacuole	45.3
Overall accuracy	**79.1**

Table 5.6: The predictive accuracy for subcellular locations of Growing Classification Tree approach for Data_SWISS.

Chapter 6

Classification using Mixtures of Experts

The Mixture of Experts model is the application of the divide-and-conquer principle where the problem is treated as one of combining multiple models (*experts*), each of which is defined over a local region of the input space. The selection of the most appropriate expert can be governed by a supervisor learning machine. This idea led to the architecture, where a *gating model* performs the division of the input space and experts (e.g. small neural networks) perform the effective calculation at each assigned region separately. An extension of this approach is the Hierarchical Mixture of Experts method, where the outputs of the different experts are combined by different supervisor gating models hierarchically organized.

The Mixture of Experts algorithm differs from other ensemble algorithms in the relation between the combination model and the basic learners. Most ensemble learning algorithms, e.g. *stacking*, first train the basic predictors (or use existing predictors) and then try to tune the combination model. The Mixture of Experts algorithm trains the combination model simultaneously with the basic learners and the current model determines the data sets provided to each learner for its further training.

The problem of training Mixtures of Experts can be treated as a maximum likelihood estimation problem. A general technique for this task is the Expectation Maximization algorithm (EM), wich I have already discussed in Chapter 5. For Mixtures of Experts the EM decouples the estimation process in a manner that fits well with the modular structure of the architecture.

Section 6.1 describes a mathematical framework for Mixtures of Experts model and the EM algorithm.

In Section 6.2 I propose the model, which I call Mixture of Bayesian Classifiers.

The two main advantages of the Mixture of Experts are localization of the different experts and usage of a dynamic model for combining the outputs. The Mixture of Experts is suitable when the patterns can be naturally divided to *simpler* (homogeneous) subsets and the learning task in each of these subsets is not as difficult as the original one. However, real world problems may not exhibit this property and furthermore, even when such a partition exists, the required gating function may be complex and the initial stage of localizing the experts has a hen-and-egg nature.

I have not found any paper which describes the use of Mixture of Experts in the field of Bioinformatics. In the biomedical research [Ansch¨utz, 2001] applied this model for skin cancer

diagnosis.

6.1 Mixtures of Experts

A *mixture of experts* consists of m experts, the outputs $y_j(x)$ of which are weighted by the outputs of a gating network $g_j(x)$ for input vector x:

$$y(x) = \sum_{j=1}^{m} g_j(x) y_j(x).$$

A mixture of experts is a probabilistic model that can be interpreted as a mixture model for estimating conditional probability distributions:

$$p(c|x) = \sum_{j=1}^{m} g_j(x) \phi_j(c|x),$$

where the ϕ_j represent the conditional densities of target value c for expert j. In terms of a mixture model, the gating network corresponds to input-dependent mixture coefficients. Note that, the gating network splits the data in a *soft* way, allowing several experts to be selected at a time.

A gating model learns to partition the data space and experts are attributed to these different regions. Since the gating model deals with the decomposition in smaller tasks, the choice of the type of gating model is an important one. Originally, it has been assumed that the gate is a feed-forward neural network [Jordan and Jacobs, 1994]. In [Xu et al., 1995] another type of gate was proposed based on an unsupervised mixture model. Such a *localized* gating model was based on GMMs (Gaussian Mixture Models). [Moerland, 1998] gives a general derivation of the EM algorithm for a localized mixtures of experts and showed that one can choose any type of mixture model as gating model. [Moerland, 1998] used also mixtures of latent variable models.

So, for localized model we have:

$$g_j(x) = \frac{\alpha_j p_j(x)}{\sum_i \alpha_i p_i(x)},$$

where $\sum_i \alpha_i = 1$, $\alpha_i \geq 0$, and the p_is are probability density functions; thus the gating network outputs g_j sum to one and are non-negative.

To obtain a one-pass solution for the gating model parameters, maximum likelihood estimation is performed on the joint density:

$$p(x, c) = p(c|x)p(x) = \sum_{j=1}^{m} \alpha_j p_j(x) \phi_j(c|x),$$

which by maximum likelihood leads to the following error function on the training data $\{x^n, c^n\}$:

$$E = -\sum_n log \sum_{j=1}^{m} \alpha_j p_j(x^n) \phi_j(c^n|x^n).$$

Minimization of this error function is done by iteratively repeating a two-step procedure. The E-step consists of calculating the expected values of the so-called *missing variables* z_j^n:

$$\varepsilon(z_j^n) = \frac{\alpha_j p_j(x^n)\phi_j(c^n|x^n)}{\sum_{i=1}^{m} \alpha_i p_i(x^n)\phi_i(c^n|x^n)} = h_j(x^n, c^n).$$

The error function can be interpreted as the sum of an unsupervised part that encourages good density estimation (gate) and a supervised part that encourages correct classification (experts). The gate error function is:

$$-\sum_n \sum_{j=1}^{m} h_j(x^n, c^n) log(\alpha_j p_j(x^n)).$$

This is almost the error function that is minimized in the M-step when applying the EM algorithm to a simple mixture model. The only difference is in the definition of their posteriors h that in the case of a localized mixture of experts include both input and output values and thus incorporate the supervised errors at the output of the expert networks.

The expert error function and consequently the M-step for the experts of the localized model is identical to the one obtained in [Jordan and Jacobs, 1994] for standard mixture of experts.

The exact form of the M-step of the EM algorithm depends on the choice of the model for the gate and experts.

6.2 Mixtures of Bayesian Classifiers

The closely related work to my approach is that of [Wiering, 2003]. His architecture is similar to the architecture of Hierarchical Mixture of Experts model of [Jordan and Jacobs, 1994], but instead of using linear networks as models, the author used Naive Bayesian Classifiers (NBC). Thus, the architecture consists of gating NBCs which partition the data and weight the expert NBCs predicting the class probabilities.

Have a look at Figure 6.1 which depicts the architecture of the model I propose. This is the Mixture of Experts model, where gate and experts are Bayesian Classifiers, which I have already used in all previous Chapters.

For the initialization of Mixture of Experts model I use the procedure proposed in the paper of [Avnimelech and Intrator, 1999], which initializes the partition of the data set to different experts in a boost-like manner. The algorithm is as follows:

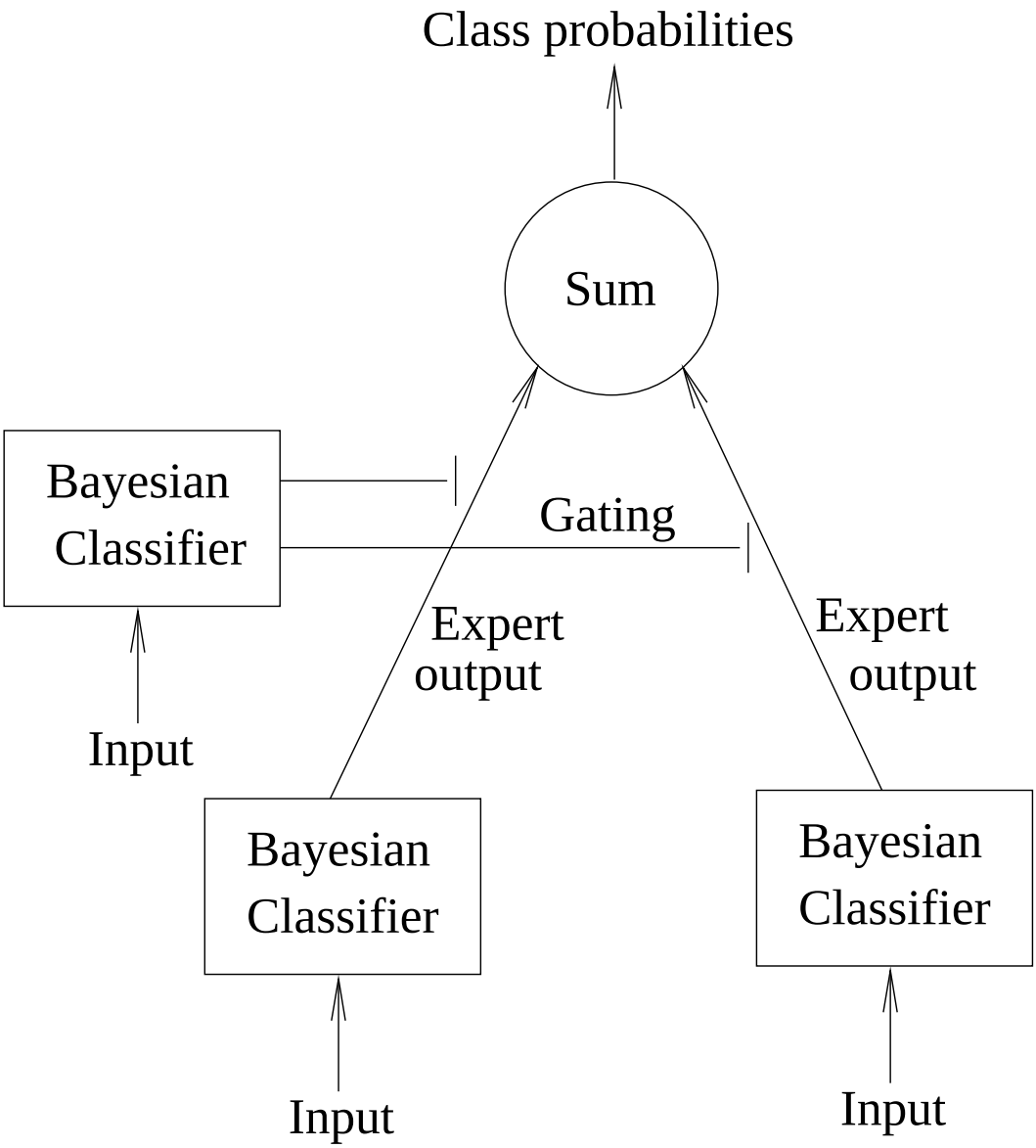

Figure 6.1: The Mixture of Bayesian Experts architecture. The total output is the weighted sum of the expert outputs, where the weights are gating classifier outputs.

The Boosted Mixture of Experts (BME) algorithm

1. Train the first expert on all the training set.
2. Assign the data instances on which the current experts are not confident to the initial training set of the new expert and train it.
3. Refining stage:
   - Partition the data according to the confidence of each expert on each data instance
   - Train each expert on its training set.
4. If more experts are required - return to step 2.

One can also use a randomizing initialization process. Two runs on the same data produce, in general, different results.

Also a Hierarchical Mixture of Experts (HME), which has a tree structure, can be considered further. The leaves of the tree contain the expert classifiers and the non-terminal nodes contain the gates, which in my case can be also Bayesian classifiers. The difference to the model I have proposed in Chapter 5 is that HMEs can train the parameters of the gates and experts simultaneously employing the EM procedure.

6.3 Results and discussion

Following Tables demonstrate that some improvement of the overall performance of classification can be achieved with ME compared to single Bayesian classification for eukaryotic and apoptosis data. But MEs did not manage to outperform BCT.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**483**	10	62	129	684
Extracellular	38	**173**	49	65	325
Mitochondrial	106	4	**125**	86	321
Nuclear	86	15	22	**974**	1097
Sum	713	202	258	1254	2427

Table 6.1: Confusion matrix of prediction results of Mixture of Experts approach for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	70.6
Extracellular	53.2
Mitochondrial	38.9
Nuclear	88.8
Overall accuracy	**72.3**

Table 6.2: The predictive accuracy for subcellular locations of Mixture of Experts approach for Data_Euk.

	Predicted group				
	Cytoplasmic	Plasma membrane	Mitochondrial	Other	Sum
Cytoplasmic	**38**	0	0	5	43
Plasma membrane	0	**29**	1	0	30
Mitochondrial	0	1	**11**	0	12
Other	2	2	0	**8**	12
Sum	40	32	12	13	97

Table 6.3: Confusion matrix of prediction results of Mixture of Experts approach for Data_Apoptosis.

Cellular location	Accuracy (%)
Cytoplasmic	88.4
Plasma membrane	96.7
Mitochondrial	91.7
Other	66.7
Overall accuracy	**88.7**

Table 6.4: The predictive accuracy for subcellular locations of Mixture of Experts approach for Data_Apoptosis.

Chapter 7

Delegating classifiers

In this section I want to apply and analyse the idea of delegating classifiers, which was proposed in [Ferri et al., 2004]. Section 7.1 first summarizes the delegation framework. In Section 7.2 I use the Bayesian classification as learning algorithm for the base classifiers and apply the delegation procedure to my classification task. In Section 7.3 and Section 7.4 I propose two possibilities to extend a fundamental notion of delegation and combine it with the ensemble methods.

7.1 Cautious classifiers and delegation

The novel approach of *delegation* can be summarized by the motto: let others do the things that you cannot do well.

[Ferri, 2004] introduced the notion of a *cautious classifier* as one that classifies only the examples for which it is sure of being able to make the right decision, and abstains for the rest of its inputs, leaving them for another classifier.

The closest idea to delegation comes from a variant of separate-and-conquer technique introduced in [Frank and Witten, 1998]. PART algorithm learns a decision tree, selects the branch with largest coverage, removes the rest of the tree and trains a second tree with the remaining examples.The process continues until all the examples are covered.

Delegation is a serial (not parallel or hierarchical), transferring (no combination), multi-classifier method. The resulting classifier is not a combination of classifiers, but a decision list.

Since each classifier retains part of the examples, the next classifier has fewer examples for training and, hence, the process can be much more efficient than other ensemble methods.

The idea of delegation arises two main questions. First, we have to determine a threshold or decision rule to decide when to apply the classifier and when to delegate to the next one. Second, we have to determine good techniques to generate classifiers that perform better as previous classifiers for the examples that they have delegated. As answers to these questions, [Ferri et al., 2004] proposes following:

- The decision whether an example has to be tackled by the classifier is made by the classifier itself, using its own estimated reliability.

- The next classifier is trained solely with the examples rejected by the previous classifier.

It may happen that the first classifier retains all the examples of one class and hence the next classifier has fewer classes than the first one.

Let us consider for a classifier *f* the associated functions $f_{class}(e)$, $f_{conf}(e)$. The function $f_{class}(e)$ returns the class assigned by classifier *f* to example *e*. The function $f_{conf}(e)$ returns the *confidence* (i.e. an estimate of the reliability) of the prediction given by classifier *f* to example *e*. If we have n classifiers, each with corresponding confidence threshold, then the *delegating decision rule* is as follows:

Decision rule for a delegating classifier with thresholds $\tau^{\mathbf{1}}, \tau^{\mathbf{2}}, \ldots, \tau^{\mathbf{n}}$

IF $f^{\mathbf{1}}_{conf}(e) > \tau^{\mathbf{1}}$ THEN PREDICT $f^{\mathbf{1}}_{class}(e)$
ELSE IF $f^{\mathbf{2}}_{conf}(e) > \tau^{\mathbf{2}}$ THEN PREDICT $f^{\mathbf{2}}_{class}(e)$
...
ELSE IF $f^{\mathbf{n-1}}_{conf}(e) > \tau^{\mathbf{n-1}}$ THEN PREDICT $f^{\mathbf{n-1}}_{class}(e)$
ELSE PREDICT $f^{\mathbf{n}}_{class}(e)$

One can consider the scenario, where the last classifier delegates its low-confidence examples back to the first. It is called then *round rebound*. The rationale is that if an example is rejected by all the classifiers, it would be best classified by the first rather than the last classifier because the first classifier is more general and potentially less overfitting.

The general algorithm for learning a delegating classifier, which was proposed in the work of [Ferri et al., 2004], is following:

Procedure Learn_delegating_classifier(Train)

$Tr^{\mathbf{1}} \leftarrow Train, i \leftarrow 0$
do
$i \leftarrow i + 1$
LEARN $f^{\mathbf{i}}$ with $Tr^{\mathbf{i}}$
Obtain $\tau^{\mathbf{i}}$
$Tr^{>} := \{e \in Tr^{\mathbf{i}} : f^{\mathbf{i}}_{conf}(e) > \tau^{\mathbf{i}}\}$
$Tr^{\leq} := \{e \in Tr^{\mathbf{i}} : f^{\mathbf{i}}_{conf}(e) \leq \tau^{\mathbf{i}}\}$
$Tr^{\mathbf{i+1}} \leftarrow Tr^{\leq}$
until $Tr^{>} = 0$ or $i > maxIterations$

[Ferri et al., 2004] proposes also some approaches to determine the thresholds of the classifiers. One of them is that a classifier retains a fixed percentage of the examples. More formally, given a fraction ρ, a classifier *f* and a training set *Tr*, we can obtain the threshold τ as following:

$$\tau = max\{t : |\{e \in Tr : f_{conf}(e) > t\}| \geq \rho * |Tr|\}$$

The method is called *Global Absolute Percentage.*

An alternative approach is to have a different threshold τ_c for each class *c* in order to handle imbalanced data sets. This is called *Stratified Absolute Percentage.*

Since there are examples, that are removed for the next classifier in the chain, patterns in the data, obscured for the previous classifier, can be revealed by the next classifiers in the chain. For example when delegation is used with *fence-and-fill methods* (i.e. methods that partition the instance space into regions), after the clearing the space the remaining small areas may be joined into bigger areas.

When using divide-and-conquer methods, such as decision trees, there will be several *delegating nodes*, which can be joined into a *graft* node.

When using decision trees, the method of safe pre-pruning can be used, which would detect when a node is not leading to leaves with confidence greater than the threshold.

The authors of [Ferri et al., 2004] proposed also the use of different learning algorithms for the base classifiers. The first classifier can be an efficient classifier (e.g. Naive Bayes), while further down the delegation chain will be more data-intensive methods.

7.2 Delegating Bayesian Classifiers

I investigated the use of the idea of delegating classifiers with Bayesian classification algorithm for learning of base classifiers.

Assume the function $f_{prob_c}(e)$ returns the probability of class *c* for example *e*. The class assigned by classifier *f* to example *e* is $f_{class}(e) = argmax_c f_{prob_c}(e)$.

Let me denote as $f_{conf}(c)$ the reliability of the prediction of class *c* by classifier *f*. This value can be calculated as the fraction of training instances of class *c* assigned to the class *c* by the classifier *f* during the training phase. Then I assume that

$$f_{conf}(e) = max_c\{f_{prob_c}(e)\} * f_{conf}(f_{class}(e)).$$

If $f_{conf}(c)$ is not greater than the threshold, we can prune this class *c* by labeling it as *delegating*. In the prediction phase, the instances, assigned by the classifier to the *delegating class*, will be directed to the subsequent classifier.

The top-down classification procedure, proposed in Chapter 5, can be used also as learning algorithm for the base classifiers. In this case, as the reliability $f_{conf}(e)$ of the prediction given by classifier *f* to example *e* the 5-fold cross-validation accuracy estimate of Bayesian classifier at the terminal node, which the example *e* reaches, can be used. Non-accurate nodes can be labeled as *delegating*. Subtrees leading to *delegating nodes* could be pruned. This would increase efficiency.

7.2.1 Results and discussion

The experiments show that delegating approach yields improvements on the overall accuracy results of Bayesian Classification approach, but works not better than Bayesian Classification Tree

approach. Tables 7.1-7.4 show the results of delegating procedure for eukaryotic and apoptosis data.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**480**	20	61	123	684
Extracellular	41	**240**	11	33	325
Mitochondrial	88	11	**154**	68	321
Nuclear	92	28	28	**949**	1097
Sum	701	299	254	1173	2427

Table 7.1: Confusion matrix of prediction results of *delegating* approach for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	70.2
Extracellular	73.8
Mitochondrial	48
Nuclear	86.5
Overall accuracy	**75.1**

Table 7.2: The predictive accuracy for subcellular locations of *delegating* approach for Data_Euk.

	Predicted group				
	Cytoplasmic	Plasma membrane	Mitochondrial	Other	Sum
Cytoplasmic	**39**	3	0	1	43
Plasma membrane	1	**27**	2	0	30
Mitochondrial	0	1	**11**	0	12
Other	4	0	0	**8**	12
Sum	44	31	13	9	97

Table 7.3: Confusion matrix of prediction results of *delegating* approach for Data_Apoptosis.

7.3 Delegating ensembles

Here I want to develop further the delegating paradigm and propose to use at each stage of delegation not the single classifier, but the combined classifier. I use the **unanimity** variation of voting (see Section 4.2) to combine the predictions of base classifiers. The combined classifier decides that an input sample comes from class C_j if and only if all the classifiers decide for class C_j, otherwise it rejects the sample (gives out unknown class). In order to make each of the base classifiers more strict, I use the same procedure as described in 4.5. I allow a classifier

Cellular location	Accuracy (%)
Cytoplasmic	90.7
Plasma membrane	90
Mitochondrial	91.7
Other	66.7
Overall accuracy	**87.6**

Table 7.4: The predictive accuracy for subcellular locations of *delegating* approach for Data_Apoptosis.

only to vote for a class if the posterior probability of this class is bigger than a certain threshold. Otherwise, it rejects an example.

Let f be a combined classifier. The *delegating decision rule* is as follows:

Decision rule for a delegating combined classifier

IF $f^{\mathbf{1}}_{class}(e) \neq$ *unknown* THEN PREDICT $f^{\mathbf{1}}_{class}(e)$
ELSE IF $f^{\mathbf{2}}_{class}(e) \neq$ *unknown* THEN PREDICT $f^{\mathbf{2}}_{class}(e)$
...
ELSE IF $f^{\mathbf{n}}_{class}(e) \neq$ *unknown* THEN PREDICT $f^{\mathbf{n}}_{class}(e)$
ELSE PREDICT $f^{\mathbf{1}}_{class}(e)$

Note, that I used here the *round rebound*.

Table 7.5 and Table 7.6 demonstrate the results of the experiment with Data_Euk. As base classifiers I used three classifiers: two learned with multinomial and chain model correspondently and one learned with chain model but with reduced alphabet.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**531**	22	60	71	684
Extracellular	58	**208**	31	28	325
Mitochondrial	103	10	**177**	31	321
Nuclear	130	38	68	**861**	1097
Sum	822	278	336	991	2427

Table 7.5: Confusion matrix of prediction results of Delegating Ensembles approach for Data_Euk.

7.4 Using delegating approach with meta-learning

In this Section I want to propose how the approaches of delegating and meta-learning can be combined.

Cellular location	Accuracy (%)
Cytoplasmic	77.6
Extracellular	64
Mitochondrial	55.1
Nuclear	78.5
Overall accuracy	**73.2**

Table 7.6: The predictive accuracy for subcellular locations of Delegating Ensembles approach for Data_Euk.

The idea is following. For each class, which can be predicted by the base classifier, one meta classifier is learned, whose task is to predict, when the base classifier will err assigning the query instance to this class. So this meta classifier will be learned on two-class training set (classes + and -), consisting of instances, which were truely classified (class +) and falsely classified (class -) by the base classifier.

In the prediction phase, an instance will be directed to the subsequent base classifier, if it is assigned by the actual base classifier to some *delegating* class. Otherwise, it will be considered by the corresponding meta classifier. If this meta classifier predicts class -, the instance will be directed to the subsequent base classifier. If it predicts class +, the class label will be given out as final prediction.

The class will be labeled as *delegating* during the training phase, if its corresponding meta classifier is *precise* for class +. The *precision* for class + is defined as following:

$$\text{precision(+)} = \frac{T(+)}{A(+)},$$

where $A(+)$ is the number of instances assigned to class + and $T(+)$ is the number of instances correctly assigned to class + during the validation of meta classifier.

The reason for this, why I use here by the assessment of the meta classifier not the overall accuracy of the meta classifier, but its precision for class +, is following. Since the sequences, falsely classified by the meta classifier as belonging to the class -, will be delegated for further learning of base classifiers, they will have the chance to be classified (hopefully correctly) by the following classifiers. The sequences, which are falsely classified by the meta classifier as belonging to the class +, will never get the chance to participate in the subsequent learning procedure, which could possibly learn to classify them correctly.

Table 7.7 and Table 7.8 report the experimental results with eukaryotic data.

	Predicted group				
	Cytoplasmic	Extracellular	Mitochondrial	Nuclear	Sum
Cytoplasmic	**486**	34	69	95	684
Extracellular	37	**246**	16	26	325
Mitochondrial	97	15	**164**	45	321
Nuclear	120	26	40	**911**	1097
Sum	740	321	289	1077	2427

Table 7.7: Confusion matrix of prediction results of *delegating* approach used with meta-learning for Data_Euk.

Cellular location	Accuracy (%)
Cytoplasmic	71.1
Extracellular	75.7
Mitochondrial	51.1
Nuclear	83
Overall accuracy	**74.5**

Table 7.8: The predictive accuracy for subcellular locations of *delegating* approach used with meta-learning for Data_Euk.

Chapter 8

Conclusions and future perspectives

Subcellular localization is a key functional characteristic of proteins.

In this thesis some novel methods for subcellular localization prediction of proteins are presented. Many already existing approaches are applied with some extensions and evaluated on this prediction task, which is the first attempt in the field of genome sequence analysis and protein function prediction.

Reported results relative to Bayesian Classification used with Markov Chain Model are good and competitive with the results of other previously published approaches.

The Bayesian Classification Tree approach shows an impressive increase of performance, when compared with the single Bayesian classifier from which it evolved. It improves in all data sets and never performs worst. The method is efficient and demonstrates good results in the empirical evaluation. It outperforms the system PSORT-B for Gram-negative bacteria data, improves significantly previously obtained results for the apoptosis proteins. It outperforms also other methods for combining classifiers.

I can not claim that BCT is the best algorithm, but there is strong evidence that, if I need to use a learning algorithm for a new data set, without any more information, I will first try BCT.

The procedure for Growing Classification Trees using mixture modeling proposed in Chapter 5 produces almost the same results as BCT in terms of prediction accuracy, but it is computationally more expensive.

The ensemble learning methods I investigated, Mixture of Experts model and delegating procedure as well, can show certain improvement of classification performance compared to basic learner, however they did not reach the overall accuracies shown by BCT.

Some improvements over the proposed BCT approach is possible. In particular, the application of post-pruning can be investigated.

Hierarchical Mixtures of Experts were left beside the scope of this thesis, but it will be interesting to compare their behaviour with the method proposed in Chapter 5. The proposed method can probably be used as the initialization procedure for further learning of HME with EM-algorithm.

One possible venue for future research may be to use Bayesian classifiers based on variable memory Markov models (VMMs) [Bejerano, 2004].

Because all the methods I have proposed in this thesis need only raw sequence data, we can

apply them for different classification tasks and for proteins that has only sequence information.

One of this classification tasks is the prediction of structural groups of proteins. Early work on proteins identified the existence of helices and extended sheets in protein secondary structures, a high-level classification which remains popular today. Although the protein folding process may require catalysts, such as chaperonins, it is widely accepted that the three-dimensional (3D) structure of a protein is related to its sequence of amino acids. This implies that it is possible to predict protein 3D structure from sequence. The most general way of obtaining the three-dimensional structure from sequnce data is to predict secondary structure, especially in the absence of a homologous sequence of known structure. With the increasing number of amino acid sequences generated by large-scale sequencing projects, and the continuing shortfall in crystallized homologous structure, the need for reliable structural prediction methods becomes ever greater.

It will be possible to use the methods, I have proposed in this work, in multiple areas of biological analysis, including classification of G-protein coupled receptors (GPCRs), nuclear receptors [Bhasin, 2004], enzyme families [Cai et al., 2004], analysis of proteins function ([Cai et al., 2003] and [Han et. al., 2005]) and prediction of RNA binding proteins (see the work of [Han et. al., 2004]).

I want to make an attempt to develop a method for recognizing the subfamilies of nuclear receptors. Nuclear receptors are key transcription factors that regulate crucial gene networks responsible for cell growth, differentiation and homeostasis. They control functions associated with major diseases (e.g. diabetes, osteoporosis and cancer). The recognition of nuclear receptors is crucial, because many of them are potential drug targets for developing therapeutic strategies for diseases like breast cancer and diabetes.

My methods could be applied in the future to predict heterologous expression of proteins in prokaryotic hosts. The availability of such predictive system would be helpful to researchers working on recombinant protein expression.

One more way to further improve the prediction performance in future is to incorporate other kind of knowledge, including gene expression profile and regulatory pathway information. Meta-learning methods as information fusion technologies provide a convenient framework for this.

Acknowledgements

I would like to thank Prof. Dr. Roland Eils, the head of the Division Theoretical Bioinformatics at the German Cancer Research Center (DKFZ), for making it possible for me to work in the excellent environment of an active biological research. I appreciate greatly his constant engagement to work on application of informatics to the biological problems. Thanks for his readiness to support my scientific interests. Also many thanks for his insightful comments on the draft of this thesis.

I am grateful to the head of our Subdivision Molecular Oncology Dr. Benedikt Brors for many interesting and stimulating discussions related to this work. Thanks for his valuable suggestions and comments on my work, sharing biological knowledge.

Many thanks to my sister and colleague Svetlana Bulashevska for our discussions on bioinformatics problems.

The work was done on permanently running computer systems, which were administered by Karlheinz Gross and Peter Weyrich, whose support I highly acknowledge.

Gratitude is expressed to Prof. Dr. Roland Vollmar, Faculty of Informatics, University of Karlsruhe, for his motivation to take the responsibility for the supervision of this interdisciplinary work and great interest on biological applications of computer science. I enjoyed our fruitful discussions and would like to thank sincerely for his comments and feedback. Special thanks for his encouragement and support.

Bibliography

[Ali and Pazzani, 1996] Ali, A.K., Pazzani, M.J. (1996) Error reduction through learning multiple descriptions. *Machine Learning*, Volume 24 , Issue 3, 173 - 202

[Ansch¨utz, 2001] Ansch¨utz, M. (2001) Aufbau eines Systems zur Unterst¨utzung der Diagnose von Hautkrebs unter Verwendung einer Mixture-of-Experts Architecture, Zentrum f¨ur Neuroinformatik, Bochum, TU Ilmenau

[Avnimelech and Intrator, 1999] Avnimelech, R., Intrator, N. Boosted Mixture of Experts: an ensemble learning scheme. *Neural Computation*, 11, 475-490

[Bairoch, 2000] Bairoch, A., Apweiler, R. (2000) The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.*, 28, 45-48

[Baxt, 1992] Baxt, W.G. (1992) Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation*, 4, 772-780

[Bay, 2000] Bay, S.D., Pazzani, M.J. (2000) Characterizing model errors and differences. *Proceedings of the 17th International Conference on Machine Learning*. Morgan Kaufmann.

[Bejerano, 2004] Bejerano, G. (2004) Algorithms for variable length Markov chain modeling. *Bioinformatics*, 20(5), 788-789

[Bhasin, 2004] Bhasin, M., Raghava, G. (2004) Classification of nuclear receptors based on amino acid composition and dipeptide composition. *J. Biol. Chem.*, Volume 279, Issue 22, 23262-23266

[Bilmes,] Bilmes, J. A gentle tutorial of the em algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models. *Technical report: tr-97-021* http://www.icsi.berkeley.edu/techreports/1997.abstracts/tr-97-021.html

[Boeckmann et al., 2003] Boeckmann, B. et al. (2003) The SWISS-PROT protein knowledge base and its supplement TrEMBL in 2003. *Nucleic Acids Res.*, 31, 365-370

[Breiman et al., 1984] Breiman, L., Friedman, J.H., Olshen, R. A., Stone, C.J. (1984) Classification and regression trees. Wadsworth, Belmont.

[Breiman, 1996a] Breiman, L. (1996) Bagging predictors. *Machine Learning*, vol. 24, 123-140

[Breiman, 1996b] Breiman, L. (1996) Arcing Classifiers. Technical Report

[Brodley and Lane, 1996] Brodley, C.E., Lane, T. (1996) Creating and exploiting coverage and diversity. *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models*, 8-14

[Brodley and Utgoff, 1995] Brodley, C.E., Utgoff, P.E. (1995) Multivariate decision trees. *Machine Learning*, 19, 45-77

[Cai et al., 2003] Cai, C.Z. et al. (2003) Protein function classification via support vector machine approach. *Math Biosci.* 185, 111-122

[Cai et al., 2004] Cai, C.Z. et al. (2004) Enzyme family classification by support vector machines. *Proteins*, 55, 66-76

[Cestnik 1986] Cestnik, B., Kononenko, I. (1986) Assistant 86: a knowledge elicitation tool for sophisticated users. *Progress in Machine Learning, EWSL*

[Chan and Stolfo, 1995] Chan, P.K., Stolfo, S.J. (1995) Learning arbiter and combiner trees from partitioned data for scaling machine learning. *KDD 95*

[Chou, 2001] Chou, K.C. (2001) Prediction of protein cellular attributes using pseudo-amino-acid-composition. *Proteins: Structure, Function and Genetics*, 43, 246-255

[Chou, 2002] Chou, K.C. (2002) A new branch of proteomics: Prediction of Protein Cellular Attributes. *Gene Cloning and Expression Technologies*, 4, 57-70

[Dempster et al., 1977] Dempster, N.M., Laird, A.P., Rubin, D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *J.R. statist.Soc. B*, 39, 185-197

[Dietterich, 2000] Dietterich, T. G. (2000) Ensemble methods in machine learning. *First International Workshop on Multiple Classifier Systems*, 1-15

[EBI, 2003] EBI (2003) European Bioinformatics Institute, http://www.ebi.ac.ik/genomes/

[Emanuelsson et al., 2000] Emanuelsson, O. et al. (2000) Predicting subcellular localization of proteins based on their N-terminal amino acid sequence.*J. Mol. Biol.*, 300, 1005-1016

[Ferri et al., 2004] Ferri, C., Flach, P.,Hernandez-Orallo, J. (2004) Delegating classifiers. *Proceedings of the 21 International Conference on Machine Learning, Canada*

[Ferri, 2004] Ferri, C., Hernandez-Orallo, J. (2004) Cautious classifiers. *ROC Analysis in Artificial Intelligence ROCAI*, 27-36

[Fisher, 1987] Fisher, D.H. (1987) Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139-172

[Frank and Witten, 1998] Frank, E., Witten, I.H. (1998) Generating accurate rule sets without global optimization. *Proceedings of the 15 International Conference on Machine Learning (ICML-98)*, 144-151

[Freund and Schapire, 1996] Freund, Y. Schapire, R. (1996) Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning*, 148-156

[F¨urnkranz, 2002a] F¨urnkranz, J. (2002) Round robin classification. *Journal of Machine Learning Research*, 2, 721-747

[F¨urnkranz, 2002b] F¨urnkranz, J. (2002) Pairwise classification as an ensemble technique. *Proceedings of the 13th European Conference on Machine Learning (ECML-02)*, 97-110

[Gama 1997] Gama, J. (1997) Probabilistic Linear Tree. *Machine Learning: Proc. of the 14th International Conference, ed. D.Fisher*

[Gama 1998] Gama, J. (1998) Combining classifiers by constructive induction. *Machine Learning: ECML-98, ed. C. Nedellec*

[Gama and Brazdil, 2000] Gama, J., Brazdil, P. (2000) Cascade generalization. *Machine Learning*, 41(3), 315-343

[Gardy et al., 2003] Gardy, J.L. et al. (2003) PSORT-B: improving protein subcellular localization prediction for Gram-negative bacteria. *Nucleic Acids Research*, Vol. 31, No.13, 3613-3617

[Garg and Pavlovic, 2002] Garg, A., Pavlovic, V., (2002) Bayesian Networks as ensemble of classifiers. *ICPR*

[Guo et al., 2004] Guo, J., et al. (2004) A novel method for protein subcellular localization Based on Boosting and Probabilistic Neural Network. *The Second Asia-Pacific Bioinformatics Conference: APBC2004 , New Zealand*

[Han et. al., 2005] Han, L.Y., Cai, C.Z. (2005) Prediction of functional class of novel viral proteins by a statistical learning method irrespective of sequence similarity. *Virology*, 5, 331(1), 136-43

[Han et. al., 2004] Han, L.Y., et al. (2004) Prediction of RNA-binding proteins from primary sequence by a support vector machine approach. *RNA*, 10, 355-368

[Horton and Nakai, 1997] Horton, P., Nakai, K. (1997) Better prediction of protein cellular localization sites with the k nearest neighbors classifier. *Proc. Of the Fifth ISMB, AAAI Press*, 298-3055. http://www.psort.nibb.ac.jp/

[Hua and Sun, 2001] Hua, S., Sun, Z. (2001) Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8), 721-728

[Huang and Li, 2004] Huang, Y., Li, Y. (2004) Prediction of protein subcellular locations using fuzzy k-NN method. *Bioinformatics*, V.20, N.121-28

[Jordan and Jacobs, 1994] Jordan, M.I., Jacobs, R.A. (1994) Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2), 181-214

[Kohavi, 1996] Kohavi, R. (1996) Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*

[Krogh et al., 1994] Krogh, A. et al. (1994) Hidden markov models in computational biology: applications to protein modeling. *J. Mol. Bio.*, 235, 1501-1531

[Krogh et al., 2001] Krogh, A. et al. (2001) Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *J. Mol. Bio.*, 305, 567-580

[Kuncheva, 2003] Kuncheva, L.I. , Whitaker, C.J. (2003) Measures of diversity in classifier ensembles. *Machine Learning*, 51, 181-207

[Langley 1993] Langley, P. (1993) Induction of recursive bayesian classifiers. *Machine Learning: ECML-93, ed. P.Brazdil*

[Li et al., 2003] Li, T., Wang, J., Fan, K., Wang, W. (2003) How simple can the proteins be: from the prediction of the classes of protein structures. *Modern Physics Letters B*, V. 17, N. 5, 1-8

[Loh, 1988] Loh, W., Vanichsetakul, N. (1988) Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*.

[Melville and Kokku] Melville, P., Kokku, R. Functional classification of genes using expression data and phylogenetic profiles. http://www.cs.utexas.edu/users/rkoku/bioinformatics/

[Merz, 1999] Merz, C.J. (1999) Using correspondence analysis to combine classifiers. *Machine Learning*, 36, 33-58

[Moerland, 1998] Moerland, P. (1998) Localized mixtures of experts. *IDIAP-RR*, 98-14, http://www.idiap.ch/

[Murthy, 1997] Murthy, S.K. (1997) Automatic construction of decision trees from data: a multi-disciplinary survey.*Data Mining and Knowledge Discovery*

[Nakai and Kanehisa, 1991] Nakai, K., Kanehisa, M. (1991) Expert system for predicting protein localization sites in Gram-negative bacteria. *Proteins*, 11, 95-110

[Nakai and Kanehisa, 1992] Nakai, K., Kanehisa, M. (1992) A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14: 897-911

[Nair and Rost, 2002] Nair, R., Rost, B. (2002) Inferring subcellular localization through automated lexical analysis. *Bioinformatics*, 18, S78-S86

[Ortega 2001] Ortega, J., Koppel, M., Argamon, S. (2001) Arbitrating among competing classifiers using learned referees. *Knowledge and Information Systems*, 3, Issue 4

[Poulsen, 1990] Poulsen, C.S. (1990) Mixed Markov and latent Markov modeling applied to brand choice behavior. *International Journal of Research in Marketing*, 7, 5-19

[Quinlan 1986] Quinlan, R. (1986) Induction of decision trees. *Machine Learning*, 1, 81-106

[Quinlan 1993] Quinlan, R. (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann.

[Reinhardt and Hubbard, 1998] Reinhardt, A., Hubbard, T. (1998) Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Res.*, 26, 2230-2236

[Ridgeway, 1997] Ridgeway, G. (1997) Finite discrete Markov process clustering. *Technical report TR 97-24, Microsoft Research*

[Savicky and F¨urnkranz, 2003] Savicky, P., F¨urnkranz, J. (2003) Combining pairwise classifiers with stacking. *Proceedings of the 5th International Symposium on Intelligent Data Analysis (IDA-03), Berlin, Germany*

[Seewald and F¨urnkranz, 2001] Seewald, A.K., F¨urnkranz, I. (2001) An evaluation of grading classifiers. *Advances in Intelligent Data analysis: Proceedings of the Fourth International Symposium (IDA-01)*, 221-232

[Seewald et al., 2000] Seewald, A.K. et al. (2000) Hybrid Decision Tree Learners with Alternative Leaf Classifiers: An Empirical Study, ÖFAI-TR-2000-10

[Skalak, 1995] Skalak, D.B. (1995) Prototype selection for composite nearest neighbor classifiers. CMPSCI Technical report 95-74

[Smyth, 1997] Smyth, P. (1997) Clustering sequences using hidden markov models. *Advances in Neural Information Processing*, 9, 648-654

[Smyth, 1999] Smyth, P. (1999) Probabilistic model-based clustering of multivariate and sequential data. *Proceedings of the Seventh International Workshop on AI and Statistics*.

[Ting and Witten, 1999] Ting, K.M., Witten, I.H. (1999) Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10, 271-289

[Todorovski and Dzeroski, 2000] Todorovski, L., Dzeroski, S. (2000) Combining multiple models with meta decision trees. *Proceedings of the Fourth European Conference on Principles of Data Mining and knowledge Discovery*, 54-64

[Torgo, 1999] Torgo L. (1999) Inductive learning of tree-based regression models. Ph.D. dissertation. http: //www.liacc.up.pt/ltorgo/PhD/

[Tsymbal et al., 2002] Tsymbal, A., Puuronen, S., Patterson, D. (2002) Ensemble feature selection with the simple Bayesian classification. *Information Fusion, Special Issue Fusion of Multiple Classifiers*

[Tsymbal et al., 1999] Tsymbal, A., Puuronen, S., Terziyan, V. (1999) Arbiter meta-learning with dynamic selection of classifiers and its experimental investigation. *Advances in Databases and Information Systems: 3rd East European Conference ADBIS'99, Maribor, Slovenia, Lecture Notes in Computer Science*, V. 1691, 205-217.

[Utgoff, 1988] Utgoff, P.E. (1988) Perceptron trees: a case study in hybrid concept representation. *Proceedings of the Seventh National Conference on Artificial Intelligence*, 601-606

[Wiering, 2003] Wiering, M.A. Hierarchical mixtures of naive bayesian classifiers. Technical report: UU-CS-2003-003

[Wolpert, 1992] Wolpert, D.H. (1992) Stacked generalization. *Neural Networks*, 5(2), 241-260

[Xu et al., 1995] Xu, L., Jordan, M., Hinton, G. (1995) An alternative model for mixtures of experts. *Advances in NIPS*, V. 7, 633-640

[Xie et al., 2002] Xie H. et al. (2002) Large-scale protein annotation through gene ontology. *Genome Research*, 12, 785-794

[Yu et al., 2003] Fine-grained protein fold assignment by support vector machines using generalized n-peptide coding schemes and jury voting from multiple-parameter sets. *Proteins*, 50, 531-536

[Yu et al., 2004] Yu, C., Lin, C., Hwang, J. (2004) Predicting subcellular localization of proteins for Gram-negative bacteria by support vector machines based on n-peptide compositions. *Protein Science*, 13, 1402-1406

[Yuan, 1999] Yuan, Z. (1999) Prediction of protein subcellular locations using Markov chain models. *FEBS Lett.*, 451, 23-26

[Zheng, 1998] Zheng, Z. (1998) Naive Bayesian Classifier Committees. *Machine Learning: ECML-98*

[Zhou and Doctor, 2003] Zhou, G., Doctor, K. (2003) Subcellular location prediction of apoptosis proteins. *Proteins: Structure, Function and Genetics*, 50, 44-48

Deutsche Zusammenfassung

Die vorliegende Dissertation ist das Ergebnis meiner Arbeit im Deutschen Krebsforschungszentrum (DKFZ) auf dem interdisziplin¨aren Gebiet Bioinformatik.

Die Bioinformatik verbindet die Gebiete Molekularbiologie, Biochemie und Genetik mit der Theoretischen und Praktischen Informatik sowie der Computerlinguistik. Sie verf¨ugt ¨uber einen rapide wachsenden Bestand an offenen Problemen und gewinnt immer mehr an Bedeutung in allen Bereichen der Biologie. Die Analyse von experimentellen und anderen Daten (z.B. Text) ist eine zentrale Aufgabe in der Bioinformatik.

Die ersten Algorithmen zur Sequenzanalyse wurden in den 50er Jahren ben¨otigt, als die ersten Proteinsequenzen verf¨ugbar wurden. Nachdem Fred Sanger 1975 die enzymatische Sequenzierung von DNA erfunden hatte, stieg auch die Anzahl der Nukleotidsequenzen kontinuierlich an. Das exponentielle Wachstums an biologischen Daten, die im Rahmen nationaler und internationaler Genomprojekte generiert werden, revolutioniert die Molekularbiologie und Biotechnologie. Über 230 vollst¨andig sequenzierte Genome sind bereits publiziert worden. Die resultierende grosse Datenmenge wird erg¨anzt durch weitere Datenquellen, die steigende Bedeutung gewinnen, z.B. mRNA Expressionsdaten (DNA Chips, EST Daten), Proteomikdaten (2D Gele, Massenspektren), dreidimensionale Strukturen von Biomolek¨ulen und biomolekularen Komplexen, Protein Interaktionsdaten und metabolitendaten.

Diese Datenvielfalt bietet ein herausragendes Anwendungsfeld f¨ur die moderne Informatik.

Die Anwendungsm¨oglichkeiten umfassen u.a.:

- die Analyse von Genexpressionsdaten,
- die Klassifikation von Tumortypen und Toxicogenomics,
- die Erkennung entfernter Homologien,
- Sekund¨arstrukturvorhersage,
- Single-Nucleotide Polymorphisms (SNPs), und
- die Analyse von Proteomikdaten.

Erfolge in der biologischen Forschung gr¨unden sich immer mehr auf Informationen aus Datenbanken, die die Generierung eigener Daten im Labor unterst¨utzen.

Die Berechnung von Expressionsmustern bestimmter Gene und die Bestimmung von Proteinprofilen und Funktionsvorhersagen in der Proteomforschung liefern dabei wichtige Impulse

f¨ur die Grundlagenforschung und sind u.a. relevant f¨ur das Verst¨andnis von Erbkrankheiten. Die theoretische Bestimmung von Zielmolek¨ulen wird f¨ur das Drug Design ben¨otigt. Basierend auf den empirisch ermittelten Strukturdaten werden therapeutische Ans¨atze und potenzielle Wirkstoffe entwickelt.

Die potentiellen Einsatzm¨oglichkeiten der Informatik in den Biowissenschaften gehen weit ¨uber ihre derzeitigen Anwendungen hinaus. Die Rolle, die die Informatik bei den Biowissenschaften derzeit spielt, ¨ahnelt der Rolle der Mathematik in der Physik: erst der Einsatz von Informatikmethoden erm¨oglicht es, in den Biowissenschaften mathematische Modelle zu bilden und damit zu rechnen.

Die Bioinformatik setzt Methoden aus verschiedenen Gebieten der Informatik ein: u.a. Kombinatorische Optimierung, Methoden der Formalen Sprachen, Genetische Algorithmen, Stochastische Algorithmen, Neuronale Netze, Mustererkennung, Maschinelles Lernen, Inductive Logic Programming, Datenbanksysteme und Data Mining, sowie Computerlinguistik.

Das Ziel dabei ist, Muster und Regelm¨assigkeiten in Daten zu erkennen, die neue wissenschaftliche Erkenntnisse erm¨oglichen. Die Muster und Regelm¨assigkeiten k¨onnen pr¨adiktiv sein (wie z.B. bei Klassifikations- oder Regressionsproblemen) oder deskriptiv (wie z.B. bei Problemen, bei denen es nur um das Finden von Abh¨angigkeiten in Daten geht).

Zur L¨osung der obigen Probleme wird in den letzten Jahren immer h¨aufiger auf Algorithmen und Techniken des Maschinellen Lernens und des Data Mining zur¨uckgegriffen. Das Maschinelle Lernen besch¨aftigt sich mit Algorithmen, die durch Erfahrung ihre F¨ahigkeit, eine Aufgabe zu l¨osen, verbessern k¨onnen. Als Data Mining bezeichnet man den Datenanalyseschritt im Prozess der Entdeckung neuen Wissens in Datenbanken („Knowlege Discovery in Databases“).

Ziel dieser Doktorarbeit ist die Entwicklung eines intelligenten Systems zur Vorhersage der Lokalisierung eines Proteins in der Zelle.

Mit der Entschl¨usselung zahlreicher Genome und davon abgeleiteter Proteome stellt sich unweigerlich die Frage nach der Funktion dieser Proteine. Mitunter kann man mittels Datenbankvergleichen homologe Proteine finden, deren Funktion bekannt ist. Alternativ w¨are es schon ein wertvoller Hinweis zur physiologischen Bedeutung eines Proteins, wenn man seine Lokalisation bestimmen oder vorhersagen k¨onnte.

Es existieren bereits Programme, die es versuchen, f¨ur einen unbekannten Protein seine Zellul¨are Lokalisierung vorherzusagen.

PSORT ist ein Programm, das unter Verwendung verschiedener Vorhersage-Algorithmen (z.B. zur Identifizierung von membranspannenden Proteinsegmenten bzw. von Signalsequenzen zur Sekretion durch die Cytoplasmamembran) solche Vorhersagen erlaubt. iPSORT ist eine Weiterentwicklung dieses Ansatzes und basiert auf der Erkennung N-terminaler Sortingsignale.

Ein anderer Ansatz ist im Programm NNPSL verwirklicht und beruht auf der Verwendung eines neuronalen Netzwerkes, mit dessen Hilfe lediglich von der Aminos¨aurezusammensetzung eines Proteins auf dessen Lokalisation geschlossen wird. Im Unterschied zum PSORT-Algorithmus ist hierbei f¨ur die erfolgreiche Vorhersage das Vorhandensein von Targetting-Signalen nicht erforderlich. Daher arbeitet dieses Programm auch mit Sequenzen aus Genvorhersage-Algorithmen, bei denen der N-Terminus unter Umst¨anden nicht korrekt vorhergesagt wurde.

Aus dem Labor von Gunnar von Heijne stammt ein Vorhersage-Algorithmus, TargetP, der davon ausgeht, dass die meisten Export- und Import-Wege N-terminale Peptidsequenzen als Lo-

kalisationssignale erkennen. Auch dieses Programm basiert auf einem neuronalen Netzwerk. Es ist eines der derzeit besten Vorhersageprogramme.

In der vorliegenden Arbeit befasse ich mich mit den realen Datens¨atzen aus der wichtigsten Datenbank f¨ur Proteinsequenzen SWISSPROT.

Im Kernpunkt dieser Arbeit steht Klassifikation. Klassifikation beinhaltet das Lernen einer Konzeptbeschreibung, die es erm¨oglicht, Datens¨atze zu vordefinierten Klassen zuzuordnen. Eine Klassifizierungsregel versucht, den Wert einer abh¨angigen Zielvariablen (die Klasse) aus den Werten von bekannten Variablen vorauszusagen. Da dem Lernverfahren vorgegebene Klassifikationen f¨ur alle Trainingsdatens¨atze zur Verf¨ugung gestellt werden, ist es ein Verfahren des ¨uberwachten Lernens. Der Erfolg des Lernens kann beurteilt werden, indem man das gelernte System f¨ur eine unabh¨angige Testdatenm¨ange ausprobiert, wobei die richtigen Klassen im Voraus bekannt sind, aber dem System nicht zur Verf¨ugung gestellt werden.

Als erstes wird in der Arbeit die Bayessche Klassifikationsmethode untersucht. Diese Methode basiert auf der Regel der bedingten Wahrscheinlichkeit von Bayes.

Beim Einlernen von stochastischen Klassifikatoren sind die Verteilungen der klassenbedingten Wahrscheinlichkeiten zu bestimmen. Ich schlage in der Arbeit zwei Modelle f¨ur die klassenbedingten Wahrscheinlichkeiten von Proteinsequenzen vor, arbeite haupts¨achlich mit dem Modell, das auf Markov-Ketten basiert, weil es bessere Klassifikationsergebnisse liefert.

Eine Markov-Kette ist ein stochastischer Prozess, der aufgrund seiner einfachen Struktur Eingang in zahlreiche Modelle gefunden hat, die unter dem Begriff Probabilistisches Maschinelles Lernen zusammengefasst werden.

Als Evaluierungskriterium f¨ur die Klassifikationsverfahren wurde die Erfolgsrate genutzt.

Mit dem Ziel die Erfolgsrate der Klassifikation zu verbessern, schlage Ich eine Hybrid-Methode vor, die die Partitionierungsst¨arke einer weiteren Klassifizierungsmethode, n¨amlich *Entscheidungsbäume*, mit der Bayesschen Klassifikation verbindet. Die empirischen Resultate zeigen, dass die Erfolgsrate der Klassifikation f¨ur alle Datens¨atze deutlich steigt.

In dieser Arbeit werden Ensemble-Lernmethoden zur Klassifikation untersucht. Ensemble-Lernen erm¨oglicht die Konstruktion eines„starken“ („m¨achtigeren“) Klassifikators durch geeignete Kombination einer Anzahl „schwacher“ Klassifikatoren. *Bagging* ist ein Verfahren bei dem zun¨achst mehrere Modelle eines Lernverfahrens parallel erzeugt werden, z.B. mehrere Bayessche Klassifikatoren. Daf¨ur wird die Trainingsmenge entsprechend aufgeteilt. Die verschiedenen Ausgaben werden dann zu einem einzigen Modell verschmolzen, dies kann z.B. durch Mehrheitsentscheidung geschehen.

Ich erforsche verschiedene Methoden, um eine Menge von Klassifikatoren zu generieren und zu kombinieren.

Ich untersuche auch einige Varianten von *Stacking*. Stacking verfolgt das Konzept eines *Metalernsystems*, welches das Abstimmungsverfahren von Bagging ersetzt. Dabei wird ein Meta-Klassifikator konstruiert, der als input die Ausgabe von mehreren Klassifikatoren verwendet. Die Idee hinter Stacking ist es, mehrere unterschiedliche Lernalgorithmen auf dasselbe Problem anzusetzen und deren Ergebnis dann mit einem Meta-Klassifikator, dessen Aufgabe ist es zu lernen, welcher Algorithmus in welchen F¨allen gute Entscheidungen trifft, zu einem Ergebnis zusammenzufassen.

Ich stelle noch ein weiteres Verfahren vor, der ein Baum aus probabilistischen Modellen

bildet, wobei die Klassifikation in den terminalen Knoten (Bl¨attern) des Baumes stattfindet. Die Modelle auf den internen Knoten des Baumes werden durch Clusteringverfahren optimiert. Die Aufgabe des Clustering besteht darin, eine Unterteilung der Dateninstanzen in sinnvolle Gruppen vorzunehmen. Es wird nach einer Beschreibung dieser Gruppen gesucht. Dabei verwendet man der Expectation-Maximization Algorithmus (EM).

Weiter schlage Ich die Erweiterung von dem Mixture-of-Experts-Ansatz vor, der statt Neuronale Netze die Bayessche Klassifikatoren als Gate-Modell und Experten besch¨aftigen soll. Das Modell wird ebenfalls mit EM-Algorithmus optimiert.

Zus¨atzlich untersuche ich das neue Konzept vom Delegieren zur Klassifikation und schlage noch zwei weitere Erweiterungen vor, die das Delegieren mit dem Meta-Lernen verbinden sollen.