ENSEMBLES FOR PREDICTING STRUCTURED OUTPUTS

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia, October 2010

Supervisor: Prof. Dr. Sašo Džeroski, Jožef Stefan Institute, Ljubljana, Slovenia

Evaluation Board:
Prof. Dr. Nada Lavrač, Chairman, Jožef Stefan Institute, Ljubljana, Slovenia
Prof. Dr. Cesare Furlanello, Member, Fondazione Bruno Kessler, Trento, Italy
Prof. Dr. Marko Robnik-Šikonja, Member, Faculty of Computer Science, University of Ljubljana, Slovenia
Dr. Jan Struyf, Member, Katholieke Universiteit Leuven, Belgium

Dragi Kocev

ENSEMBLES FOR PREDICTING STRUCTURED OUTPUTS

Doctoral Dissertation

ANSAMBLI ZA NAPOVEDOVANJE STRUKTURIRANIH VREDNOSTI

Doktorska disertacija

Supervisor: Prof. Dr. Sašo Džeroski

October 2010

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL
Ljubljana, Slovenia

Contents

1 Introduction

In this chapter, we present an overview of the thesis and motivate it within its research area. We start by outlying the context of the work performed in this thesis. Next, we state the motivation and the original contributions of the thesis. Finally, we sketch a road map for the rest of the thesis.

1.1 General perspective

The work presented in this thesis falls within the area of artificial intelligence (McCarthy *et al.*, 1955), more specifically in the area of machine learning. Machine learning studies the computer programs that have ability to learn, i.e., the computer programs that improve with experience (Mitchell, 1997). A very significant part of the research in machine learning is concerned with extracting new knowledge out of available data, i.e., the experience is given in the form of learning examples (instances). This type of machine learning is called inductive learning (Bratko, 2000).

In the classical inductive learning setting, the available learning examples are given in a form of a table. Each row of the table is an example and each column is a property of the example (called attribute). If the goal is to predict the value of one property of the examples (called target attribute) using the values of the remaining properties (called descriptive attributes), then the task is called *predictive modelling* (or supervised learning). On the other hand, if such target property does not exist and the goal is to provide general descriptions of the examples, then the task is called *descriptive modelling* (or unsupervised learning) (Langley, 1996). Examples of machine learning methods for predictive modelling include decision trees, decision rules, Bayesian networks and support vector machines and examples of machine learning methods for descriptive modelling include clustering, association rules modelling and principal-component analysis (Bishop, 2007).

Predictive and descriptive modelling are considered as different machine learning tasks. The goal of predictive modelling is to identify clusters that are compact in the target space (i.e., instances with similar value of the target variable). The goal of the descriptive modelling, on the other hand, is to identify clusters compact in the descriptive space (i.e.,

instances with similar values of the descriptive variables). Blockeel (1998) presented a machine learning task, called *predictive clustering*, that combines the advantages of both predictive and descriptive modelling. The predictive clustering identifies clusters that are compact both in the target and the descriptive space. The methods presented in this thesis are based on the *predictive clustering* framework (Blockeel, 1998).

The predictive and descriptive modelling are connected by the machine learning methods that partition the instances, such as decision trees and decision rules. These two methods are already available in the predictive clustering framework: Blockeel *et al.* (1998); Struyf and Džeroski (2006) developed the decision trees for predictive clustering, called predictive clustering trees (PCTs), and Ženko (2007) developed the decision rules for predictive clustering, called predictive clustering rules (PCRs). These methods, in addition to providing clusters of the instances, provide symbolic descriptions of the clusters. The clusters from the decision trees are described by the conjunction of the conditions from the nodes that are on the path from the root node to the given cluster (node of the tree, typically a leaf). The clusters from the decision rules are described by the rule's conditions.

Typical machine learning methods for predictive modelling are able to make a prediction for a single target attribute of an example. The target attribute can be either a discrete variable (classification) or a continuous variable (regression). However, there are many real life domains, such as image annotation, text categorization, gene networks, etc., where the input and/or the output can be structured. In this thesis, we are concerned with the latter: tasks with structured outputs.

There are two groups of methods for solving the task of predicting structured outputs (Bakır *et al.*, 2007; Silla and Freitas, 2010): (1) methods that predict component(s) of the output and then combine the separate models to get the overall prediction (called local methods) and (2) methods that predict the complete structure as a whole (called global methods). The latter group of methods has several advantages over the former. They can exploit and use the dependencies that exist between the components of the structured output in the model learning phase and as a result have better predictive performance. Next, they are more efficient: it can easily happen the number of components in the output to be very large (e.g., hierarchies in functional genomics) in which case executing a basic method for each component is not feasible. Furthermore, they produce models that are typically smaller than the sum of the sizes of the models for the components. The predictive clustering framework belongs to the group of global approaches.

The predictive clustering framework was extended so far in the context of prediction of multiple discrete variables (Blockeel *et al.*, 1998; Ženko, 2007), predicting multiple continuous variables (Blockeel *et al.*, 1998; Struyf and Džeroski, 2006; Ženko, 2007),

hierarchical multi–label classification (Vens *et al.*, 2008) (the output is a set of classes that are organized in a hierarchy) and prediction of short time series (Slavkov *et al.*, 2010b). This was done by adjusting the variance and prototype functions (needed for the induction of the trees and the rules) specifically for each task. Each of the tasks was evaluated empirically and confirmed the advantages of the global methods stated above.

To further increase the predictive performance of the predictive clustering trees, in this thesis, we extend the predictive clustering framework in the context of ensemble methods. Ensemble methods construct a set of classifiers (an ensemble) and combine their outputs to obtain a single prediction (Dietterich, 2000a). There are many practical studies that show that ensembles achieve high predictive performance and that they lift the predictive performance of a single classifier (Banfield *et al.*, 2007; Bauer and Kohavi, 1999; Breiman, 1996a; Freund and Schapire, 1996; Opitz and Maclin, 1999). Furthermore, several theoretical explanations were offered that justify and explain the high predictive performance of the ensembles (Allwein *et al.*, 2000; Breiman, 1996b; Domingos, 2000; Geman *et al.*, 1992; Kong and Dietterich, 1995; Mason *et al.*, 2000; Schapire *et al.*, 1997).

The different ensemble methods can differ in how they construct the set of constituent (or base) classifiers and in how they combine their predictions. Having in mind that combining identical or very similar classifiers will not produce an increase of predictive performance, it only makes sense to construct ensembles of classifiers that are diverse. The diversity in the ensemble is obtained by learning heterogeneous classifiers, by modifying the training set or by changing the learning algorithm. To obtain the prediction of the ensemble, classifier fusion or classifier selection can be used (Džeroski *et al.*, 2009). The former selects the best classifier and uses its predictions as predictions of the ensemble. The latter combines the predictions of all base classifiers by means of a *voting scheme* and gives the combined predictions as predictions of the ensemble. There is a plethora of ensemble learning methods and voting schemes that have been proposed in the literature (for an overview, see (Kuncheva, 2004; Seni and Elder, 2010)).

In this thesis, we focus on two widely used ensemble methods that use decision trees as base classifiers: bagging (Breiman, 1996a) and random forests (Breiman, 2001a). As base classifiers, we use predictive clustering trees. We also provide adequate voting schemes for combining the predictions (for the structured outputs) obtained from the base classifiers.

1.2 Motivation

In many real life problems the output (i.e., the target property) is structured, meaning that there are either dependencies between classes (e.g., classes are organized into a tree-shaped hierarchy or directed acyclic graph) or there are some internal relations between the classes (e.g., sequences). These types of problems occur in domains such as: life sciences (predicting the functions of a gene, selecting the most important genes for a given disease, detecting toxic molecules, etc), ecology (analysis of remote sensed data, habitat modelling), multimedia (annotation and retrieval of images and videos), semantic web (categorization and analysis of text and web) etc. Having in mind the needs of the application domains and the increasing generation of structured data, Yang and Wu (2006) listed the machine learning methods for "mining complex knowledge from complex data" as one of the ten challenging problems in machine learning.

There are variety of methods that have been proposed (Bakır *et al.*, 2007) that are specialized for predicting a given type of a structured output (e.g., hierarchy of classes (Silla and Freitas, 2010)). However, many of these are computationally demanding and not suited for dealing with large datasets (especially large outputs). The predictive clustering framework offers an unifying approach for the different types of structured outputs and the algorithms developed in this framework construct the classifiers very efficiently. Moreover, the PCTs and PCRs can be easily interpreted by a domain expert thus support the process of knowledge extraction.

To further increase the predictive performance of a single classifier, one can construct an ensemble of classifiers. In the simple classification and regression tasks, it is widely accepted that an ensemble of classifiers lifts the predictive performance of its base classifiers (Dietterich, 2000a; Džeroski *et al.*, 2009; Kuncheva, 2004; Seni and Elder, 2010). However, in the task of predicting structured outputs using the predictive clustering framework (and the other global classifiers), this is not that obvious. In the case when the base classifiers are decision trees, Bauer and Kohavi (1999) conclude that the increase in performance is related to the trees being unpruned, i.e., overfitting. On the other hand, Blockeel *et al.* (2006) state that predictive clustering trees overfit less than the single classification approach. Having in mind these two conflicting influences, it is not obvious whether an ensemble of predictive clustering trees will significantly increase the predictive performance of a single predictive clustering tree.

The global classifiers exploit the dependencies between the components of the structured outputs and, as a result, have better predictive performance than the local classifiers. However, in the ensemble learning setting, it is not clear if the predictive performance of

an ensemble of global classifiers will be better or worse than the predictive performance of ensembles of local classifiers (i.e., an ensemble per component of the structured output). It is not also clear which of these two methods will be more efficient, in terms of running time and size of the classifiers.

Another open issue in ensemble learning is how many classifiers are enough for getting the optimal performance. Bauer and Kohavi (1999); Opitz and Maclin (1999) observe ensembles of up to 30 classifiers and show that the biggest improvement in terms of predictive performance is achieved after adding the first 10-15 classifiers. After that, the error rate gradually reaches a plateau. They suggest 25 classifiers as a reasonable compromise between the predictive performance and the efficiency of an ensemble. On the other hand, Banfield *et al.* (2007) investigate ensembles of 1000 classifiers and propose an algorithm that chooses when the ensemble learning should stop. The algorithm uses stabilization of the error rates as a stopping criterion for the ensemble learning. This means that the number of classifiers in the ensemble is going to be different for each dataset. Moreover, this approach further adds to the computational complexity of the ensemble learning. Since the issue of the 'ensemble size' is not completely resolved for the simple classification and regression tasks, it is even less known how many global classifiers are enough for optimal performance of an ensemble of global classifiers.

1.3 Contributions

In this thesis, we propose to use ensembles of PCTs for predicting structured outputs to address the issues raised in the previous section. We summarize the main contributions of the work presented in this thesis as follows:

- We develop ensemble learning methods for predicting structured outputs that are based on PCTs. To the best of our knowledge, this is the first work done on ensembles of global classifiers[1]. Moreover, the proposed methods are general in terms of the type of the structured output. Currently, they are suitable for three types of structured outputs: multiple continuous targets, multiple discrete targets and classes organized into a hierarchy (tree-shaped or directed acyclic graph), however, they can

[1]There is a distinction between ensemble and architecture of classifiers. An ensemble of classifiers combines the outputs of each base classifier to obtain the overall prediction. An architecture of classifiers is a set of classifiers whose outputs are not just directly combined to obtain the overall prediction but rather the output of one classifier can be used in the training of some of the next classifiers (Ianakiev and Govindaraju, 2000). An example of architecture of classifiers are the 'classifier chains' (Read *et al.*, 2009).

be easily adapted for other types of structured outputs. With this we extend the predictive clustering framework in the context of ensemble learning.

- We perform extensive empirical evaluation of the proposed methods over a variety of domains. The experimental results show that ensembles of global classifiers lift the predictive performance of a single global classifier. We also construct ensembles of up to 1000 classifiers and select ensembles of 50 global classifiers as optimal in terms of predictive performance and efficiency. Next, although the difference in the predictive performance of the ensembles of global classifiers and the ensembles of local classifiers is not statistically significant, the ensembles of global classifiers often have better predictive performance than the ensembles of local classifiers. Moreover, the ensembles of global classifiers are more efficient in terms of training time and size of the trees in the ensembles.

- We propose a method, based on random forest, that performs feature ranking for structured outputs. Traditionally, in the tasks with structured outputs, the feature ranking is obtained by constructing several feature rankings for the components of the outputs and then aggregating them to obtain a single overall feature ranking. The method we propose produces single feature ranking and takes into account the dependencies and the relations that exist between the components of the structured output. Moreover, the ranking produced this way is more computationally efficient than building feature rankings for the components separately. On a case study for biomarker discovery, we show that feature ranking for multiple targets offers some advantages over the ranking for a single target.

- We suggest a novel ensemble learning method that is based on the beam search technique and uses decision trees as base classifiers. This method offers direct control over the diversity in the ensemble and allow to further investigate the trade-off between the ensemble's diversity and ensemble's predictive performance. In turn, the optimal trade-off will lead towards creating an ensemble with the best predictive performance. Furthermore, by selecting the top-ranked tree from the ensemble (since the beam keeps the trees sorted by a heuristic score) as representative for the whole ensemble, we get an 'interpretable' ensemble.

- We apply the ensembles for predicting structured outputs in three domains: modelling the vegetation condition, image annotation and prediction of gene functions. Each application gives a contribution to the respective domain.

  - We extract knowledge about the resilience of some indigenous vegetation types

and the relative importance of biophysical and landscape attributes that influence their condition. Next, we use the ensembles of PCTs to generate maps of the condition of the indigenous vegetation across the Victoria State, Australia. We construct the ensembles using easily obtained and remotely acquired data in conjunction with adequate field data. The generated maps can be further used in support of biodiversity planning, management and investment decisions.

- We apply the ensembles of PCTs for HMC to two benchmark tasks for hierarchical annotation of medical (X-ray) images and an additional task for photo annotation. The ensembles of PCTs for HMC achieve better results than a collection of SVMs (trained with a χ^2 kernel), the best-performing and most-frequently used approach to (hierarchical) image annotation, on all three tasks. Moreover, for the two medical image datasets, they produce the best results reported in the literature so far. Furthermore, the ensembles of PCTs for HMC are more efficient (smaller training and testing times) than the collection of SVMs.
- We present the use of PCTs for HMC and ensembles of PCTs for HMC in functional genomics, i.e., to predict gene functions (using FunCat and the Gene Ontology as function clsasification schemes), for each of the following three model organisms: *Saccharomyces cerevisiae* (yeast), *Arabidopsis thaliana* (cress) and *Mus musculus* (mouse). The ensembles of PCTs for HMC outperform a statistical learner based on SVMs for *Saccharomyces cerevisiae*, both in predictive performance and in efficiency. Also, they are competitive to statistical and network based methods for *Mus musculus* data. Overall, the ensembles of PCTs for HMC yield state-of-the-art quality (predictive performance) for gene function prediction.

1.4 Organization

2 Background

The work we present in this thesis concerns the learning of ensembles for predicting structured outputs. In this chapter, we first define the machine learning tasks that we consider: the tasks of predicting multiple targets and hierarchical multi-label classification. We then give an overview of the three paradigms that are the basis for the approaches presented in this thesis: ensemble learning, predictive clustering and predicting structured outputs.

2.1 Machine learning tasks considered

First, we formally describe the machine learning tasks that we consider here. We follow the suggestions by Džeroski (2007), where predictive modelling is defined for arbitrary types of input and output data. In particular, we describe the tasks of predicting multiple targets and hierarchical multi-label classification.

2.1.1 The task of predicting multiple targets

This task was previously referred to as multi-objective prediction (Demšar *et al.*, 2006; Kocev *et al.*, 2007b; Struyf and Džeroski, 2006). However, the term 'multi-objective' is already established in the area of optimization. We will thus use the term 'predicting multiple targets' or multi-target prediction (resp. multi-target classification and regression). We define the task of predicting multiple targets as follows.

Given:

- A description space X that consists of tuples of values of primitive data types (boolean, discrete or continuous), i.e., $\forall X_i \in X, X_i = (x_{i_1}, x_{i_2}, ..., x_{i_D})$, where D is the size of the tuple (or number of descriptive variables),

- a target space Y which is a tuple of several variables that can be either continuous or discrete, i.e., $\forall Y_i \in Y, Y_i = (y_{i_1}, y_{i_2}, ..., y_{i_T})$, where T is the size of the tuple (i.e., number of target variables),

- a set of examples E, where each example is a pair of tuples from the description and target space, respectively, i.e., $E = \{(X_i, Y_i)|X_i \in X, Y_i \in Y, 1 \leq i \leq N\}$ and N is the number of examples of E ($N = |E|$), and
- a quality criterion q (which rewards models with high predictive accuracy and low complexity).

Find: a function $f : X \rightarrow Y$ such that f maximizes q.

In this thesis, the function f is represented with decision trees, i.e., predictive clustering trees or ensembles thereof.

Figures 2.1 and 2.2 show examples of datasets with multiple targets. If the tuples from Y (the target space) consist of continuous/numeric variables (Figure 2.1), then the task at hand is multi-target regression. Likewise, if the tuples from Y consist of discrete/nominal variables (Figure 2.2), then the task is called multi-target classification.

	Descriptive variables					**Target variables**						
Site ID	LandCover	TempRange	NativeTreeProb	GrassProb1Ha_RegionStdDev	...	*Large tree score*	*Tree canopy score*	*Understorey score*	*Litter score*	*Logs score*	*Weeds score*	*Recruitment score*
ID1	2.0	225.0	0.0	1.79	...	7	3	15	3	5	15	5
ID2	4.0	278.0	2.0	12.31	...	6	3	10	5	5	13	3
ID3	8.0	191.0	54.0	6.52	...	0	5	10	5	0	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 2.1: An example of a dataset with multiple continuous targets used for modelling the condition of indigenous vegetation (Kocev *et al.*, 2010). The descriptive variables are obtained from a geographical information system, while the target variables are indices describing the condition of the vegetation.

2.1.2 The task of hierarchical multi-label classification

Classification is defined as the task of learning a model using a set of classified instances and applying the obtained model to a set of previously unseen examples (Breiman *et al.*,

	Descriptive variables						Target variables												
Sample ID	Temperature	$K_2Cr_2O_7$	NO_2	Cl	CO_2	...	*Chlorophyta Cladophora sp.*	*Chlorophyta Gongrosira incrustans*	*Chlorophyta Oedogonium sp.*	*Chlorophyta Stigeoclonium tenue*	*Bacillariophyta Melosira varians*	*Bacillariophyta Nitzschia palea*	*Rhodophyta Audouinella chalybea*	*Hirudinea Erpobdella octoculata*	*Amphipoda Gammarus fossarum*	*Ephemeroptera Baetis rhodani*	*Trichoptera Hydropsyche sp.*	*Trichoptera Rhyacophila sp.*	*Diptera Simulim sp.*	*Oligochaeta Tubifex sp.*
ID1	0.66	0.00	0.40	1.46	0.84	...	1	0	0	0	0	1	1	0	1	1	1	1	1	1
ID2	2.03	0.16	0.35	1.74	0.71	...	0	1	0	1	1	1	1	0	1	1	1	1	1	0
ID3	3.25	0.70	0.46	0.78	0.71	...	1	1	0	0	1	0	1	0	1	1	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 2.2: An example of a dataset with multiple discrete targets used for habitat modelling of bioindicator organisms (Džeroski *et al.*, 2000). The descriptive variables are chemical parameters of the water samples, while the target variables are the abundances of 14 bioindicator organisms.

1984; Langley, 1996). The unseen examples are classified into a single class from a set of possible classes.

Hierarchical classification differs from the 'traditional' classification in the following: the classes are organized in a hierarchy: An example that belongs to a given class automatically belongs to all its super-classes (this is known as the 'hierarchy constraint'). Furthermore, an example can belong simultaneously to multiple classes that can follow multiple paths from the root class. This task is then called hierarchical multi-label classification (HMC) (Silla and Freitas, 2010; Vens *et al.*, 2008).

We formally define the task of hierarchical multi-label classification as follows:

Given:

- A description space X that consists of tuples of values of primitive data types (boolean, discrete or continuous), i.e., $\forall X_i \in X, X_i = (x_{i_1}, x_{i_2}, ..., x_{i_D})$, where D is the size of the tuple (or number of descriptive variables),
- a target space S, defined with a class hierarchy $(C, \leq_h)$, where C is a set of classes and $\leq_h$ is a partial order (structured as a rooted tree) representing the superclass relationship ($\forall\ c_1, c_2 \in C : c_1 \leq_h c_2$ if and only if c_1 is a superclass of c_2),
- a set E, where each example is a pair of a tuple and a set from the descriptive and target space respectively, and each set satisfies the hierarchy constraint, i.e., $E = \{(X_i, S_i) | X_i \in X, S_i \subseteq C, c \in S_i \Rightarrow \forall c' \leq_h c : c' \in S_i, 1 \leq i \leq N\}$ and N is the number of examples of E ($N = |E|$), and

- a quality criterion q (which rewards models with high predictive accuracy and low complexity).

Find: a function $f : X \rightarrow 2^C$ (where 2^C is the power set of C) such that f maximizes q and $c \in f(x) \Rightarrow \forall c' \leq_h c : c' \in f(x)$, i.e., predictions made by the model satisfy the 'hierarchy constraint'.

In our case, the function f is represented with decision trees, i.e., predictive clustering trees or ensembles thereof.

Figure 2.3 gives an example of hierarchical multi-label classification. In particular, it presents an example dataset for annotation of medical X-ray images. The descriptive variables are descriptors extracted from the images using the edge histogram technique, while the targets are the annotations of the images using the IRMA coding scheme (Lehmann *et al.*, 2003).

Image	Descriptive variables						Classes
	|	—	/	\	S		
	48	24	59	66	37	...	cervical spine@ musculoskeletal system
	36	25	53	45	15	...	middle abdomen@renal pelvis
	35	25	56	52	19		lumbar spine@ musculoskeletal system
...	...	...	...	...	...	...	...

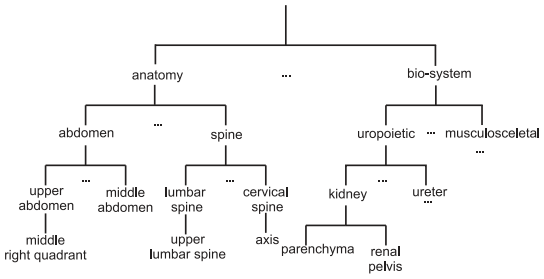


Figure 2.3: An example of a dataset for hierarchical multi-label classification of medical X-ray images (Dimitrovski *et al.*, 2008).

2.2 Related work

Having defined the machine learning tasks we address, we now present basic ideas and concepts from three machine learning paradigms relevant to our work: ensemble methods, predictive clustering and predicting structured outputs. First, we discuss why and how ensembles are built. Then, we present the predictive clustering framework and its advantages. Finally, we present related approaches for predicting structured outputs.

2.2.1 Ensemble learning

Ensemble methods are machine learning techniques that generate a set of classifiers and combine their predictions into a single prediction (Dietterich, 2000a; Džeroski *et al.*, 2009; Kuncheva, 2004; Valentini, 2003). Each of the constituent classifiers is called a *base*

classifier and the set of classifiers is called an *ensemble*. The notion of ensemble is general and applies to other types of predictive models, such as regression models. However, most of the survey literature in this area only talks about ensemble classifiers, and so does this section. Many practical studies show that ensembles achieve high predictive performance and lift the predictive performance of a single classifier (Banfield *et al.*, 2007; Bauer and Kohavi, 1999; Breiman, 1996a; Freund and Schapire, 1996; Opitz and Maclin, 1999). Furthermore, several theoretical explanations are offered that justify and explain the high predictive performance of ensembles (Allwein *et al.*, 2000; Breiman, 1996b; Domingos, 2000; Geman *et al.*, 1992; Kong and Dietterich, 1995; Mason *et al.*, 2000; Schapire *et al.*, 1997).

Ensemble learning is now an established research area in the field of machine learning. It attracts a great deal of research effort reflected in the amount of published literature (Dietterich, 2000a,b; Džeroski *et al.*, 2009; Kittler *et al.*, 1998; Kuncheva, 2004; Seni and Elder, 2010; Valentini, 2003). In the remainder of this section, we explain how ensembles are constructed, how the base classifiers are combined to obtain a single prediction, and why ensembles have good predictive performance.

Ensemble creation techniques

An ensemble is a set of classifiers. We present the three most widely used approaches to ensemble learning (i.e., constructing the different base classifiers): (1) use of heterogeneous classifiers; (2) manipulating the training set (manipulating the training instances, manipulating the feature space, or both) and (3) manipulating the learning algorithm. Table 2.2.1 summarizes the most often used ensemble learning methods that utilize these approaches. In the following, we shortly describe these approaches and some representative methods.

In the first approach, the ensemble is constructed by learning heterogeneous classifiers (such as decision trees, neural networks, naïve Bayes, nearest neighbours, etc). One can use a voting scheme (Kuncheva, 2004) to combine the predictions of the different classifiers into a single prediction. However, the most prominent ensemble learning method that employs this technique is *stacking* (Džeroski and Ženko, 2004; Wolpert, 1992). Stacking combines the classifiers not by a fixed voting scheme, but by learning an additional *meta* classifier that uses as input the predictions of the base classifiers. The performance of stacking highly depends on the attributes that are used in the dataset for constructing the *meta* classifier and the selection of the learning algorithm for the *meta* classifier.

In the second approach, the base classifiers are constructed by manipulating the training set. This approach is typically used in combination with unstable classifiers. An unstable

Table 2.1: Summarized approaches to ensemble learning.

Method	Use of heterogeneous classifiers	Manipulate the data instances	Manipulate the data features	Manipulate the learning algorithm
Stacking (Wolpert, 1992)	√			
Bagging (Breiman, 1996a)		√		
Random forests (Breiman, 2001a)		√	√	√
Bootstrap ensemble with noise (Raviv and Intrator, 1996)		√		
Boosting (Freund and Schapire, 1996)		√		
Random subspaces (Ho, 1998)			√	
Bagging of subspaces (Panov and Džeroski, 2007)		√	√	
Neural networks ensemble (Hansen and Salamon, 1990)				√
Randomized FOIL (Ali and Pazzani, 1996)				√
Randomized C4.5 (Dietterich, 2000b)				√
Extra-Trees ensemble (Geurts *et al.*, 2006a)				√

classifier is one that suffers great changes in its structure with small changes in the training set. A typical example of such a classifier is the decision tree classifier (Breiman, 1996a).

The manipulation of the training set is performed by manipulating the instances, manipulating the feature space, or both. The manipulation of the instances is done using different techniques, such as *bootstrapping* or *boosting*. Bootstrapping creates several bootstrap replicates of the training dataset by random selection with replacement (Berthold and Hand, 2003). A classifier is then learned using each of the bootstrap replicates. The most prominent ensemble learning method that uses bootstraping is *Bagging* (Breiman, 1996a). *Bagging* can use any type of classifier as a base classifier, but, most often it uses decision

trees.

Raviv and Intrator (1996) construct ensembles of neural networks using bootstrap replicates of the training set. Additionaly, noise is added to the instances of the bootstrap replicates. The noised replicates are then used to train the neural networks.

Boosting (Freund and Schapire, 1996) is a cascade procedure. It re-weights the instances of the training set based on the predictions from the previously trained classifier, thus creating a chain of classifiers. If an instance was correctly classified, then its weight is decreased when used to train the next classifier or if an instance was miss-classified, then its weight is increased when it is used to train the next classifier. The training set with the re-weighted instances is used to train the next classifier. This ensures that the different classifiers are focused on different areas of the instance space. The procedure iterates until the predictive performance of the ensemble or the number of trained classifiers reaches some user defined threshold.

The manipulation of the feature space can be done by random selection of feature sub-spaces from the original feature space. Each of the base classifiers is then learned using a different feature sub-space. The most widely used ensemble learning method that manipulates the feature space is the *Random Subspaces Method* (Ho, 1998). This approach is expected to perform well when the data have a high dimensionality (i.e., large feature space) and a small number of instances. Redundancy in the feature space can positively influence the performance of this method.

There are several ensemble learning methods that change both the instance and the feature space to build an ensemble; here we mention two of them: *Bagging of subspaces* (Panov and Džeroski, 2007) and *Random forests* (Breiman, 2001a). *Bagging of subspaces* constructs the base classifiers using both bootstrap replicates of the training set and feature sub-spaces. This method can use any type of classifier as base classifier.

Random forests are the most famous ensemble learning method that can only use decision trees as base classifiers. This method combines bootstrapping with feature sub-space selection as follows. It constructs each tree using a different bootstrap replica of the training set. During tree construction, at each node of the tree it considers a different (randomly selected) subset of the features. This method is more 'time efficient' (especially when the feature space is large) than the rest of the ensemble methods. *Random forests* can also be considered as a ensemble learning method that manipulates the learning algorithm itself.

The manipulation of the learning algorithm is the last ensemble construction approach that we present here. It constructs the base classifiers by changing the learning algorithm (e.g., some of its parameters) for each base classifier. There are several ensemble learning

methods that use this approach. One of the earliest ensembles of this type was constructed by Hansen and Salamon (1990), where each base classifier is a neural network obtained with different initial parameters. Another group of ensemble methods that use trees and rules as base classifiers perform random selection of a split from the set of possible splits, as described bellow.

Ali and Pazzani (1996) randomized the FOIL rule learning algorithm as follows. First, all candidate solutions with score at least 80% of the top-ranked candidate are calculated. Then, the selection of a condition is done using a weighted random choice algorithm. Dietterich (2000b) has proposed a similar method with C4.5 decision trees as base classifiers. At each node of a decision tree, the top 20 best ranked tests are calculated. One test is selected from these 'test candidates' randomly (with equal probability) and is used as the test at the given node. Geurts *et al.* (2006a) have proposed the 'E-Tree Ensemble' algorithm. For choosing a test in each internal node, K attributes are randomly selected first; for each of these attributes a random split is picked next. From the resulting set of tests, the best performing test is then selected and placed at the given node.

Ensemble combination schemes

One of the most important issues in ensemble learning is the proper combination of the predictions of the base classifiers into a single prediction (Kittler *et al.*, 1998; Kuncheva, 2004). There are generally two approaches for obtaining a single prediction from an ensemble: classifier selection and classifier fusion/combination (Džeroski *et al.*, 2009).

The classifier selection approach first evaluates the performance of each base classifier. The prediction of the ensemble in that case is the prediction of the best performing classifier. This approach, however, uses only one classifier to make a prediction and its performance is limited by the performance of the best classifier. The advantages of this approach are that the final classifier is simpler, understandable and can be executed fast.

The classifier fusion/combination approach combines the predictions of all base classifiers into an overall prediction of the ensemble. *Stacking* can be viewed as a classifier fusion approach: it uses the predictions of the base classifiers to train a meta classifier, and is then used to combine the predictions of the base classifiers to produce the overall prediction from of the ensemble. However, by far the most common method for classifier fusion is the use of a voting scheme. There are many different voting schemes that can be selected based on the task (classification or regression) or based on the problem at hand. Here, we describe the ones that are most often used in real-world domains.

The most widely used voting schemes for classification tasks are the *majority* and *probability distribution vote*. Majority voting counts how many of the classifiers predicted

each of the possible class values. Each base classifier has a single 'vote', i.e., it predicts a single class. The final prediction of the ensemble is the class with the most 'votes', i.e., the class that was most often predicted by the base classifiers.

A weighted sum of the votes can also be used, where the vote from each classifier is weighted by a number in the interval [0, 1]. Weights are assigned based on the the classifier's overall performance (such as accuracy, area under the ROC curve, F-measure etc...) or in some more complex manner. (Kuncheva, 2004). The overall prediction of the ensemble is then the class value with the highest weighted sum of votes.

In the probability distribution voting scheme, the base classifiers predict the probability that an example belongs to each possible class. Thus, each base classifier gives its vote (i.e., probability estimate) for each class separately. At the end, the predicted class is the one that has highest sum of probabilities from all base classifiers. As for the majority voting scheme, one can weight the votes of the base classifiers by their overall performance. There are more complex voting schemes, but they are seldomly used by the community. These voting schemes include naïve Bayes combination (Domingos and Pazzani, 1997), multinomial methods to estimate the posterior probabilities for each class (e.g., the behavior knowledge space method (Huang and Suen, 1995) and Wernecke's method (Wernecke, 1992)), probabilistic approximations (Kuncheva, 2004) and singular value decomposition (i.e., correspondence analysis) (Merz, 1999).

For regression tasks, the most widely used scheme for combinining the predictions of the base models is *averaging*. This combining scheme is simple: It takes the predictions of all classifiers and calculates their mean value. This mean value is then used as the prediction of the ensemble. One can use weights for the predictions of the base classifiers. The weights (as for classification), can be related to the performance of the classifires (e.g., correlation coefficient, relative root mean squared error, etc...) or more complex (Kuncheva, 2004). Other voting schemes for regression (Kittler *et al.*, 1998; Kuncheva, 2004) include the (weighted) median, (weighted) geometric mean, generalized mean, fuzzy integral, decision templates, etc.

Why do ensembles perform well?

A necessary condition for an ensemble to perform better than any of its base classifiers is that the base classifiers are accurate and diverse (Hansen and Salamon, 1990; Hastie and Tibshirani, 1990). An accurate classifier makes smaller error on unseen instances than random guessing. Diverse classifiers make different errors on unseen instances (i.e., the errors of the classifiers are independent). These conditions were regarded as a sufficient requirement for the effective ensemble. However, Kuncheva and Whitaker (2003) have

shown that this is not always the case: the classifiers producing independent errors not always do outperform the ones that produce dependent errors. Actually, there exists a trade-off between the accuracy and the independence of the base classifiers. Dietterich (2000a); Džeroski *et al.* (2009); Valentini (2003) offer several fundamental reasons and theoretical analyses as to why ensembles of classifiers perform well.

First, learning algorithms search for the best model in a given space of models. However, in the real world problems there are only limited quantities of data available. The learning algorithm can thus find several models that are equally good for the data at hand. By combining them into an ensemble, the algorithm reduces the risk of choosing the wrong model.

The second reason for the success of ensembles comes from the fact that learning algorithms perform some kind of local search and can easily get stuck in local optima. If an ensemble is constructed with multiple restarts of the search, it can provide a better approximation to the true model.

Another reason is that, the true model of the problem under consideration may not reside in the space of possible models. By combining the multiple different models, the space of possible models is expanded. This extended space of models may include also the true model or a better approximation thereof.

There are two main theories that explain why ensembles are successful. The first theory considers ensembles from the view point of large margin classifiers (Allwein *et al.*, 2000; Mason *et al.*, 2000; Schapire *et al.*, 1997). According to this theory, the ensembles enlarge the margins, thus enhancing the generalization capability. The second theory uses bias-variance decomposition of the error (Breiman, 1996b; Geman *et al.*, 1992; Kong and Dietterich, 1995) to show that the ensemble can reduce the variance and the bias. Domingos (2000) has proved that the margin-based and bias-variance-based explanations are equivalent.

Interpretability of ensembles

Fayyad *et al.* (1996) define the process of knowledge discovery in databases as "the overall process of discovering useful knowledge from data". This proces consists of several steps: data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, data mining and interpretation of the obtained data mining results. The proper interpretation of the results is crucial "to ensure that useful knowledge is derived from the data". Considering this, high predictive performance is not always sufficient for the results (extracted models) to be regarded as useful: They need to be insepected and understood by human users. Furthermore, for many real life problems, the users need models that give

better insight into the domain rather than high predictive performance.

Many studies, both theoretical and empirical (the references above), show that the ensembles often outperform their base classifiers/models and offer high predictive performance. Since the ensembles are set of classifiers they do not offer additional insight about the problem at hand. However, some useful knowledge from an ensemble can be extracted by using a meta model that represents the whole ensemble (thus sacrificing some of its predictive performance) (Assche, 2008) or by performing feature ranking using the ensembles mechanism(Breiman, 2001a). We briefly discuss these two approaches in the following.

The meta model can be constructed by constructing a complex model while building the ensemble or by learing a new model based on the ensemble. The first approach constructs an complex understandable model (such as alternating decision tree, consolidated tree and orthogonal decision tree) that has some ensemble characterists: it combines several predictions to get the final prediction (Freund and Mason, 1999; Kargupta *et al.*, 2006; Pérez *et al.*, 2004). The second approach uses the models that are in the ensemble to construct or extract a model representative for the whole ensemble. This can be done by selecting a representative model by some measure (Ferri *et al.*, 2002) or by generating artificial data using the base models and use these data to learn a representative model (Assche, 2008; Craven, 1996; Domingos, 1998).

Breiman (2001a) proposed to further expolit the random forests (or bagging) mechnism for providing ranking of the descriptive variables (i.e., feature ranking). To calculate the importance of each descriptive variable for the class, this approach performs random permutations of the variable's values and out-of-bag error estimates. This approach offers additional insights into the domain while preserving the predictive performance of the ensemble.

2.2.2 Predictive clustering

The notion of *predictive clustering* was first introduced by Blockeel (1998). The predictive clustering framework unifies two machine learning techniques, predictive modelling and clustering, usually viewed as completely different. The connection between these techniques is made by machine learning methods that partition the instances into subsets, such as decision trees and decision rules. These methods can be considered both as predictive and as clustering methods (Langley, 1996). In particular, the predictive clustering framework regards the decision tree as a hierarchy of clusters: each node corresponds to a cluster and the top node contains all instances. Similarly, a decision rule represents a cluster that contains the instances which it covers.

The benefit of using predictive clustering methods is that, besides the clusters themselves, they also provide symbolic descriptions of the constructed clusters. Each node from the tree (i.e., cluster) can be described with a conjunction of conditions, namely those on the path from the root node to the given node. A cluster represented by a rule is described by the rule's conditions. The difference between the 'tree' and 'rule' clusters is that the 'tree' clusters are ordered in a hierarchy and do not overlap, while the 'rule' clusters represent a flat clustering, where clusters may overlap.

Predictive clustering combines predictive modelling and clustering techniques (Blockeel, 1998; Blockeel *et al.*, 1998; Ženko, 2007). The task of predictive clustering is to identify clusters of instances that are close to each other both in the target and in the descriptive space. Figure 2.4 illustrates the tasks of predictive modelling (Figure 2.4(a)), clustering (Figure 2.4(b)) and predictive clustering (Figure 2.4(c)). Note that Figure 2.4 presents the target and the descriptive space as one-dimensional axes for easier visual interpretation, but they can be of higher dimensionality.

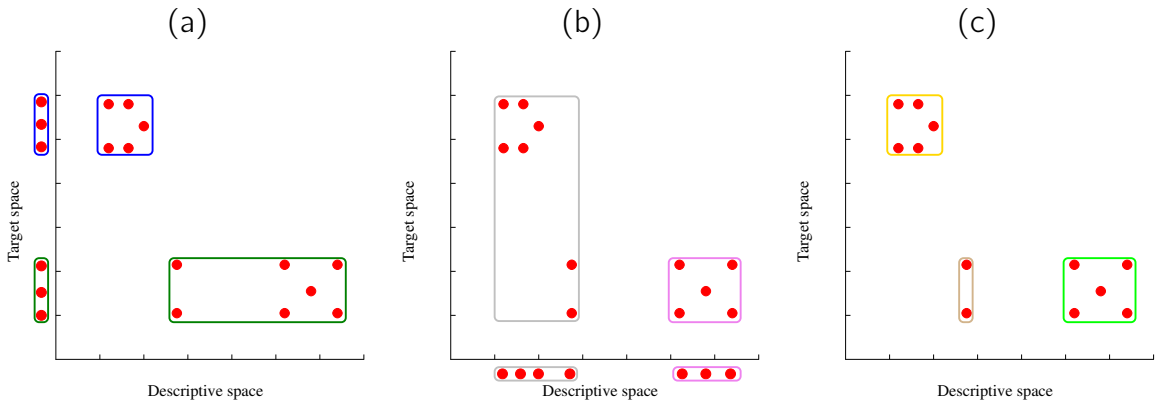


Figure 2.4: An illustration of predictive clustering: (a) clustering in the target space, (b) clustering in the descriptive space, and (c) clustering in both the target and the descriptive space. Figure taken from Blockeel (1998) and Ženko (2007).

The clusters that were obtained using the target space only (Figure 2.4(a)) are homogeneous in the target space (the target variables of the instances belonging to the same cluster have similar values). On the other hand, the clusters obtained using the descriptive space only (Figure 2.4(b)) are homogeneous in the descriptive space (the descriptive variables of the instances belonging to the same cluster have similar values). The predictive clustering combines these two and produces clusters that are homogeneous both in the target and in the descriptive space (Figure 2.4(c)).

Each cluster that is identified by predictive clustering is associated with a predictive model. The predictive model makes a prediction for the target space using the descriptive

space for all the instances belonging to that cluster. Typically, the prediction of the model is the projection of the prototype of the cluster on the target space: The prototype is an instance representative of the cluster and most similar to all elements of the cluster.

The predictive clustering framework is implemented using decision trees (called predictive clustering trees) (Blockeel *et al.*, 1998; Struyf and Džeroski, 2006) and decision rules (called predictive clustering rules) (Ženko, 2007) as predictive models. These two machine learning methods use a heuristic function to split the instances into clusters. The heuristic function, in the predictive clustering framework, is based on minimization of the intra-cluster variance and maximization of the inter-cluster variance. The variance and prototype function for performing the clustering of the instances need to be instantiated depending on the prediction task at hand. So far, the predictive clustering framework has been used for the prediction of multiple continuous variables, prediction of multiple discrete variables, hierarchical multi-label classification (HMC) and prediction of time series. The predictive clustering framework is implemented in the CLUS system[1] (Blockeel and Struyf, 2002; Kocev *et al.*, 2007b; Slavkov *et al.*, 2010b; Struyf and Džeroski, 2006; Vens *et al.*, 2008; Ženko, 2007).

The variance function has been instantiated as follows. For predicting multiple discrete variables, the variance is calculated as the average value of the Gini index for each variable. The variance can also be calculated by using information gain or entropy (Blockeel *et al.*, 1998; Ženko, 2007). When predicting multiple continuous variables the variance, is calculated by using the Euclidean distance for each variable. The contribution of each variable is normalized, thus, each target variable contributes equally to the overall variance (Struyf and Džeroski, 2006; Ženko, 2007). Moreover, the contribution of each target variable, both when predicting continuous or discrete variables, to the overall variance can be weighted, thus making the model better for some of the target variables. In the task of HMC, the variance is calculated by using a weighted Euclidean distance (Vens *et al.*, 2008). Some other distance measures, such as the weighted Jaccard distance, a semantic similarity measure etc, can be also used (Aleksovski *et al.*, 2009). The variance for the prediction of time series (Slavkov *et al.*, 2010b) is calculated by using the dynamic time warping distance (Sakoe and Chiba, 1978) or a qualitative distance measure (Todorovski *et al.*, 2002), or the correlation of the time series. The predictive clustering framework can be easily extended with new variance functions, thus extending it for other prediction tasks.

The prototype function is also appropriately instantiated for each prediction task. The prototype when predicting multiple continuous variables is the vector of the mean values

[1]The CLUS system is available for download at `http://www.cs.kuleuven.be/~dtai/clus`.

of each variable Blockeel *et al.* (1998); Struyf and Džeroski (2006). The median can be used instead of the mean in the prototype. Moreover, a complex prototype function that weights the instances can be used to calculate the prototype. In the task for prediction of multiple discrete variables, the prototype is calculated as a vector of the probability distributions with each distribution containing the probabilities of the class values for each target separately. Afterwards, the majority classes per target are easily retrieved (Blockeel *et al.*, 1998). The prototype in the case of HMC is calculated by using the average values per class and then applying some user defined threshold (see Section 3.1.2 for details). When predicting time series, the prototype is calculated as the mean and/or the medoid value, where the medoid is taken with respect to the used distance measure. Both prototypes are reported when all time series have equal length, while only the median is reported when the time series have different lengths.

The predictive clustering framework offers a unifying view over several machine learning tasks. A proper instantiation of the variance and prototype function enables the framework to handle a given prediction task. So far, the predictive clustering framework has used only decision trees and decision rules as predictive models.

In this thesis, we extend the predictive clustering framework towards ensemble learning. In particular, we investigate whether an ensemble of predictive clustering trees improve the performance of individual predictive clustering trees. We also investigate whether ensemble for predicting structured outputs outperform the ensembles learned separately for each component of the target.

2.2.3 The task of predicting structured outputs

The task of predicting structured outputs is gaining more and more attention within the machine learning research community (Bakır *et al.*, 2007; Silla and Freitas, 2010). The methods for predicting structured outputs can be divided into two main groups: local and global. The local methods decompose the output to its smallest components, construct a classifier/model for each of the components and then combine their outputs to obtain a structured prediction. Standard, traditionally developed machine learning methods (Berthold and Hand, 2003; Breiman *et al.*, 1984; Langley, 1996; Mitchell, 1997; Tan *et al.*, 2005) can be used to construct the classifiers for each sub-componenent.

The global methods, on the other hand, construct only a single classifier that predicts the complete structured output at once (the so-called 'big-bang' approach (Silla and Freitas, 2010)). The main advantage of the global approaches is that they are able to exploit the interdependencies between the components of the outputs (given in the form of constraints or statistical correlations) (Bakır *et al.*, 2007; Blockeel *et al.*, 2006; Ženko,

2007).

The proposed methods for predicting structured outputs are typically 'computationally demanding and ill-suited for dealing with large datasets' (Bakır *et al.*, 2007). In this thesis, we propose a global method for predicting structured outputs that has good predictive performance and is very efficient. We use the predictive clustering framework both for predicting multiple targets and for hierarchical multi-label classification. In the literature, there are mostly methods that solve one of these two tasks. In the remainder of this section, we first present the methods that predict multiple targets and then the methods for hierarchical multi-label classification.

Methods for multi-target prediction

The task of predicting multiple targets is connected with the 'multi-task learning' (Caruana, 1997) and 'learning to learn' (Thrun and Pratt, 1998) paradigms. These paradigms include the task of predicting a variable (continuous or discrete) using multiple input spaces (i.e., biological data for a disease obtained using different technologies); predicting multiple variables from multiple input spaces, and predicting multiple variables from a single input space. We consider here the last task: The approach we take can handle two types of outputs/targets: discrete targets (classification) and continuous targets (regression), while most of the approaches from the literature can handle only one type of targets.

There is extensive empirical work showing an increase in predictive performance when multiple tasks are learned simultaneously as compared to learning each task separately (for example, see (Baxter, 2000; Ben-David and Borbely, 2008; Caponnetto *et al.*, 2008; Evgeniou *et al.*, 2005) and the references therein).

The key for the success of multi-task learning is the 'relatedness' between the multiple tasks. The notion of 'relatedness' is differently perceived and defined by different researchers. For example, Ando *et al.* (2005) assume that all related tasks have some common hidden structure. Greene (2007) models the relatedness under the assumption of correlation between the noise for the different regression estimates. Baxter (2000) views the similarity through a model selection criterion, i.e., learning multiple tasks simultaneously is beneficial if the tasks share a common optimal hypothesis space. To this end, a generalized VC-dimension is used for bounding the average empirical error of a set of predictors over a class of tasks.

We present and categorize the related work along four dimensions: statistics, statistical learning theory, Bayesian theory and kernel learning. To begin with, in statistics, Brown and Zidek (1980) extend the standard ridge regression to multivariate ridge regression, while Breiman and Friedman (1997) propose the curds&whey method, where the relations

between the task are modeled in a post-processing phase. In statistical learning theory, for handling multiple tasks, an extension of the VC-dimension and the basic generalization bounds for single task learning are proposed by Baxter (2000) and Ben-David and Borbely (2008).

Most of the work in multi-task learning is done using Bayesian theory (Bakker and Heskes, 2003; Thrun and Pratt, 1998; Wilson *et al.*, 2007). In this case, simultaneously with the parameters of the models for each of the tasks, a probabilistic model that captures the relations between the various tasks is being calculated. Most of these approaches use hierarchical Bayesian models.

Finally, there are many approaches for multi-task learning using kernel methods. For example, Evgeniou *et al.* (2005) extend the kernel methods to the case of multi-task learning by using a particular type of kernel (multi-task kernel). The regularized multi-task learning then becomes equivalent to single-task learning when such a kernel is used. They show experimentally that the support vector machines with multi-task kernels have significantly better performance than the ones with single-task kernels. For more details on kernel methods and SVMs for multi-task learning, we refer the reader to (Argyriou *et al.*, 2008; Cai and Cherkassky, 2009; Caponnetto *et al.*, 2008; Micchelli and Pontil, 2004) and the references therein.

Methods for hierarchical multi-label classification

A number of approaches have been proposed for the task of hierarchical multi-label classification (Bakır *et al.*, 2007). Silla and Freitas (2010) survey and categorize the HMC approaches based on their characteristics and the application domains. The characteristics of the approaches they consider as most important are: prediction of single or multiple paths from the hierarchy, the depth of the predicted class, the type of the taxonomy that can be handled (tree or directed acyclic graph) and whether the approach is local (constructs a model for each part of the taxonomy) or global (constructs a model for the whole taxonomy). The most prominent application domains for these approaches are functional genomics (biology), image classification, text categorization, and genre classification.

Here, we present and group some existing approaches based on the learning technique they use. We group the methods as follows: network based methods, kernel base methods and decision tree based methods.

Network based methods. The network based approaches predict functions of unannotated genes based on known functions of genes that are nearby in a functional association network or protein-protein interaction network (Chen and Xu, 2004). Mostafavi *et al.* (2008) calculate per gene function a composite functional association network from mul-

tiple networks derived from different genomic and proteomic data sources. Since the network base approaches are based on label propagation, a number of approaches were proposed to combine predictions of functional networks with those of a predictive model. Tian *et al.* (2008), for instance, use logistic regression to combine predictions made by a functional association network with predictions from a random forest.

Kernel based methods. Lee *et al.* (2006) combine Markov random fields and support vector machines which are generated for each class separately. They compute diffusion kernels and use them in kernel logistic regression. Obozinski *et al.* (2008) present a two-step approach in which SVMs are first learned independently for each class separately (allowing violations of the hierarchy constraint) and are then reconciliated to enforce the hierarchy constraint. Similarly, Barutcuoglu *et al.* (2006) use un-thresholded SVMs learned for each class separately and then combine the SVMs by using a Bayesian network so that the predictions are consistent with the hierarchical relationships.

Guan *et al.* (2008) extend the method by Barutcuoglu *et al.* (2006) to an ensemble framework. Valentini and Re (2009) also propose a hierarchical ensemble method that uses probabilistic SVMs as base learners. It combines the predictions by propagating the weighted true path rule both top-down and bottom-up through the hierarchy, which ensures consistency with the hierarchy constraint.

Rousu *et al.* (2006) present a more direct approach that does not require a second step to make sure that the hierarchy constraint is satisfied. Their approach is based on a large margin method for structured output prediction which defines a joint feature map over the input and the output space. Next, it applies a SVM based techniques to learn the weights of a discriminant function (defined as the dot product of the weights and the joint feature map). Rousu *et al.* (2006) propose a suitable joint feature map and an efficient way for computing the argmax of the discriminant function (which is the prediction for a new instance).

Decision tree based methods. The disadvantage of sub-symbolic learning techniques, such as SVMs, is the lack of interpretability: it is very hard to find out why a SVM assigns certain classes to an example, especially if a non-linear kernel is used. In contrast to the output of the previously described models, decision trees are easily interpreted by a domain expert.

Clare (2003) adapts the well-known decision tree algorithm C4.5 (Quinlan, 1993) to cope with the issues introduced by the HMC task. This version of C4.5 (called C4.5H) uses the sum of entropies of the class variables to select the best split. C4.5H predicts classes on several levels of the hierarchy, assigning a larger cost to misclassifications higher up in the hierarchy. The resulting tree is then transformed into a set of rules, and the best

rules are selected, based on a significance test on a validation set.

Geurts *et al.* (2006b) present a decision tree based approach related to predictive clustering trees. They start from a different definition of variance and then kernelize this variance function. The result is a decision tree induction system that can be applied to structured output prediction using a method similar to the large margin methods mentioned above. Therefore, this system could also be used for HMC after defining a suitable kernel. To this end, an approach similar to that of Rousu *et al.* (2006) could be used.

Blockeel *et al.* (2002, 2006) proposed the idea of using predictive clustering trees (Blockeel *et al.*, 1998) for HMC tasks. This work (Blockeel *et al.*, 2006) presents the first thorough empirical comparison between an HMC and SC decision tree method in the context of tree shaped class hierarchies. Vens *et al.* (2008) extend the algorithm towards hierarchies structured as DAGs and show that learning one decision tree for predicting all classes simultaneously outperforms learning one tree per class (even if those trees are built by taking into account the hierarchy).

3 Ensembles for predicting structured outputs

In this chapter, we present the main contribution of this thesis: ensemble methods for predicting structured outputs. We begin by presenting the predictive clustering trees and their instantiations for predicting multiple continuous variables, predicting multiple discrete variables and hierarchical multi-label classification. Next, we describe how ensemble learning methods can be adapted to use predictive clustering trees as base predictive models. Finally, we show approach to prediction of structured outputs using local predictive models.

3.1 PCTs for structured outputs

The Predictive Clustering Trees (PCTs) framework sees a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The PCT framework is implemented in the CLUS system (Blockeel and Struyf, 2002), which is available for download at `http://www.cs.kuleuven.be/~dtai/clus`.

CLUS takes as input a set of examples $E = \{(x_i, y_i)|i = 1, ...N\}$, where each x_i is a vector of attribute values and y_i are values of a structured (output) datatype T_Y. In this thesis, we consider three different classes of datatypes T_Y: tuples of discrete values, tuples of real values, and hierarchies. For each type T_Y, CLUS needs two functions to be defined. The prototype function returns a representative structured value given a set of such values. The variance function describes how homogeneous a set of structured values is: It is typically based on a distance function on the space of structured values.

PCTs can be induced with a standard 'top-down induction of decision trees' (TDIDT) algorithm (Breiman *et al.*, 1984). The algorithm is presented in Table 3.1. It takes as input a set of examples (E) and outputs a tree. The heuristic (h) that is used for selecting the tests (t) is the reduction in variance caused by partitioning ($\mathcal{P}$) the instances (see line 4 of BestTest procedure in Table 3.1). Maximizing the variance reduction maximizes cluster homogeneity and improves predictive performance.

The main difference between the algorithm for learning PCTs and a standard decision tree learner (for example, see the C4.5 algorithm proposed by Quinlan (1993)) is that

Table 3.1: The top-down induction algorithm for PCTs.

procedure PCT(E) **returns** tree

1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$
2: **if** $t^* \neq none$ **then**
3: **for each** $E_k \in \mathcal{P}^*$ **do**
4: $tree_k = \text{PCT}(E_k)$
5: **return** node(t^*, $\bigcup_k \{tree_k\}$)
6: **else**
7: **return** leaf(Prototype(E))

procedure BestTest(E)

1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **for each** possible test t **do**
3: $\mathcal{P} =$ partition induced by t on E
4: $h = Var(E) - \sum_{E_k \in \mathcal{P}} \frac{|E_k|}{|E|} Var(E_k)$
5: **if** $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ **then**
6: $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: **return** $(t^*, h^*, \mathcal{P}^*)$

the former considers the variance function and the prototype function, that computes a label for each leaf, as parameters that can be instantiated for a given learning task. So far, the PCTs have been instantiated for the following tasks: multiple targets prediction (Kocev *et al.*, 2007b; Struyf and Džeroski, 2006), hierarchical-multi label classification (Vens *et al.*, 2008) and prediction of time-series (Slavkov *et al.*, 2010b). In this thesis, we focus on the first two tasks.

3.1.1 PCTs for multiple target variables

PCTs that are able to predict multiple targets simultaneously are called multi-target decision trees (MTDTs). The MTDTs that predict a tuple of continuous variables (regression tasks) are called multi-target regression trees (MTRTs), while the MTDTs that predict a tuple of discrete variables are called multi-target classification trees (MTCTs). The instantiation of the CLUS system that learns multi-target trees is called CLUS-MTDT.

PCTs for multiple continuous variables

An example of a MTRT is shown in Figure 3.1. The internal nodes of the tree contain tests on the descriptive variables (in this case, data from a geographical information system) and the leafs store the predictions (in this case, a vector of indices describing the condition of the vegetation).

The variance and prototype functions for MTRTs are instantiated as follows. The variance is calculated as the sum of the variances of the target variables, i.e., $Var(E) = \sum_{i=1}^{T} Var(Y_i)$. The variances of the targets are normalized, so each target contributes equally to the overall variance. The prototype function (calculated at each leaf) returns as a prediction the vector of the mean values of the target variables, calculated by using the training instances that belong to the given leaf.

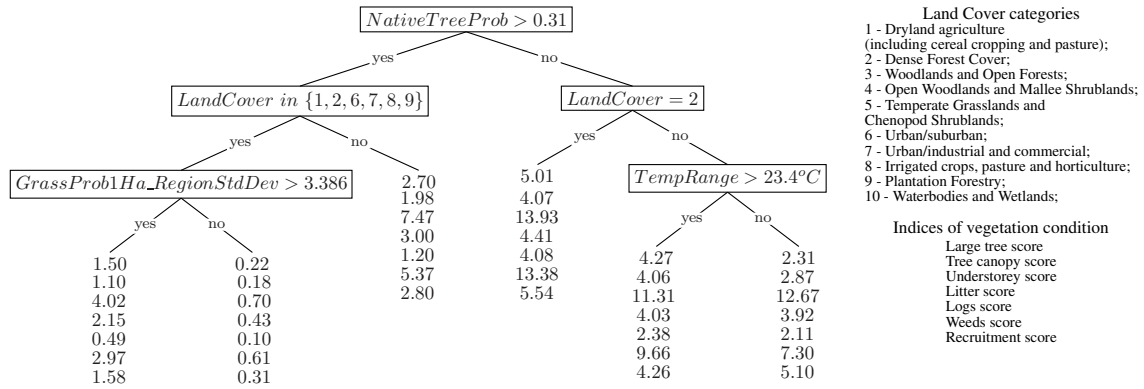


Figure 3.1: Example of a predictive clustering tree for predicting multiple continuous targets (Kocev *et al.*, 2009). Each leaf stores a prediction for the values of a set of indices of the state of indigenous vegetation in Victoria, Australia. The tree was learned using the data depicted in Figure 2.1.

PCTs for multiple discrete variables

An example of a MTCT is shown in Figure 3.2. This MTCT presents a habitat model for 14 bioindicator species (Džeroski *et al.*, 2000). The internal nodes of the tree contain tests on the descriptive variables (in this case, chemical parameters of the water samples) and the leafs store the predictions (in this case, which species are encountered and which not in a given water sample).

The variance function for the MTCTs is computed as the sum of the Gini indices of the target variables, i.e., $Var(E) = \sum_{i=1}^{T} Gini(E, Y_i)$. Furthermore, one can also use the sum of the entropies of class variables as a variance function, i.e., $Var(E) = \sum_{i=1}^{T} Entropy(E, Y_i)$ (this definition has also been used in the context of multi–label prediction (Clare, 2003)).

The prototype function returns a vector of probabilities that an instance belongs to a given class for each target variable. Using these probabilities, the most probable (majority) class for each target attribute can be calculated. In addition to the two aforementioned instantiations of the variance function for classification problems, the CLUS system also implements other variance functions, such as reduced error, information gain, gain ratio and the *m*-estimate.

3.1.2 PCTs for hierarchical multi–label classification

Hierarchical multi-label classification is a variant of classification where a single example may belong to multiple classes at the same time and the possible classes are organized

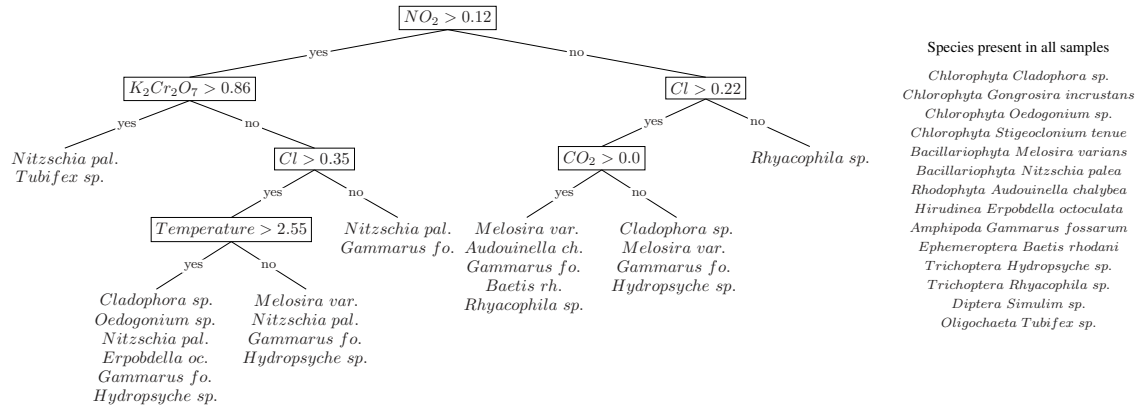


Figure 3.2: Example of a predictive clustering tree for predicting multiple discrete targets. Each leaf stores a prediction for the presence or absence of each bioindicator species. The tree was learned using the data depicted in Figure 2.2.

in a hierarchy. An example that belongs to some class c automatically belongs to all super-classes of c: This is called the hierarchical constraint. Problems of this kind can be found in many domains including text classification, functional genomics, and object/scene classification. Silla and Freitas (2010) give a detailed overview of the possible application areas and the available approaches to HMC.

Silla and Freitas (2010) describe the algorithms for hierarchical classification with a 4-tuple $\langle \Delta, \Sigma, \Omega, \Theta \rangle$. In this 4-tuple, Δ indicates whether the algorithm makes predictions for a single or multiple paths in the hierarchy, Σ is the depth of the predicted classes, Ω is the taxonomy structure of the classes that the algorithm can handle, and Θ is the type of the algorithm (local or global). Using this categorization, the algorithm we present here can be described as follows:

- Δ = multiple path prediction: the algorithm can assign multiple paths or predicted classes to each instance.
- Σ = non-mandatory leaf-node prediction: an instance can be labeled with a label at any level of the taxonomy.
- Ω = tree or directed acyclic graph: the algorithm can handle both tree-shaped or DAG hierarchies of classes.
- Θ = global classifier: the algorithm constructs a single model valid for all classes.

CLUS-HMC is the instantiation (with the distances and prototypes as defined bellow) of the PCT algorithm for hierarchical classification implemented in the CLUS system.

An example of a PCT for HMC is shown in Figure 3.3. This PCT is predicting the annotations of medical X-ray images (Dimitrovski *et al.*, 2008). The internal nodes of the tree contain tests on the descriptive variables (in this case, descriptors of the images extracted by the edge histogram technique) and the leafs store the predictions/annotations (in this case, classes organized into a tree-shaped hierarchy called IRMA coding scheme (Lehmann *et al.*, 2003)).

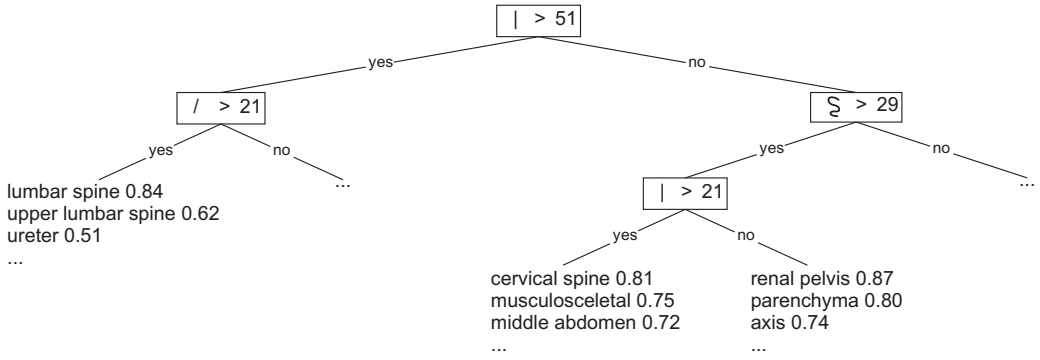


Figure 3.3: Example of a predictive clustering tree for hierarchical multi-label classification. Each leaf stores a prediction for the annotations of a given medical X-ray image from the IRMA coding scheme. The tree was learned using the data depicted in Figure 2.3.

Instantiation of PCTs for HMC

To apply PCTs to the task of hierarchical multi-label classification, the variance and prototype are defined as follows (Vens *et al.*, 2008). First, the set of labels of each example is represented as a vector with binary components; the i'th component of the vector is 1 if the example belongs to class c_i and 0 otherwise. It is easily checked that the arithmetic mean of a set of such vectors contains as i'th component the proportion of examples of the set belonging to class c_i.

The variance of a set of examples E is defined as the average squared distance between each example's class vector (L_k) and the set's mean class vector ($\overline{L}$), i.e.,

$$Var(E) = \frac{\sum_k d(L_k, \overline{L})^2}{|E|}.$$

In the HMC context, the similarity at higher levels of the hierarchy is more important than the similarity at lower levels. This is reflected in the distance measure used in the above formula, which is a weighted Euclidean distance:

$$d(L_1, L_2) = \sqrt{\sum_i w(c_i) \cdot (L_{1,i} - L_{2,i})^2},$$

where $L_{k,i}$ is the i'th component of the class vector L_k of an instance X_k, and the class weights $w(c)$ decrease with the depth of the class in the hierarchy. More precisely, $w(c) = w_0 \cdot \text{avg}_j \{w(p_j(c))\}$, where $p_j(c)$ denotes the j'th parent of class c and $0 < w_0 < 1$).

For example, consider the toy class hierarchy shown in Figure 3.4(a,b), and two data examples: (X_1, S_1) and (X_2, S_2) that belong to the classes $S_1 = \{c_1, c_2, c_{2.2}\}$ (boldface in Figure 3.4(b)) and $S_2 = \{c_2\}$, respectively. We use a vector representation with consecutive components representing membership of class c_1, c_2, $c_{2.1}$, $c_{2.2}$ and c_3, in that order (preorder traversal of the tree). The distance is then calculated as follows:

$$d(S_1, S_2) = d([1, 1, 0, 1, 0], [0, 1, 0, 0, 0]) = \sqrt{w_0 + w_0^2}.$$

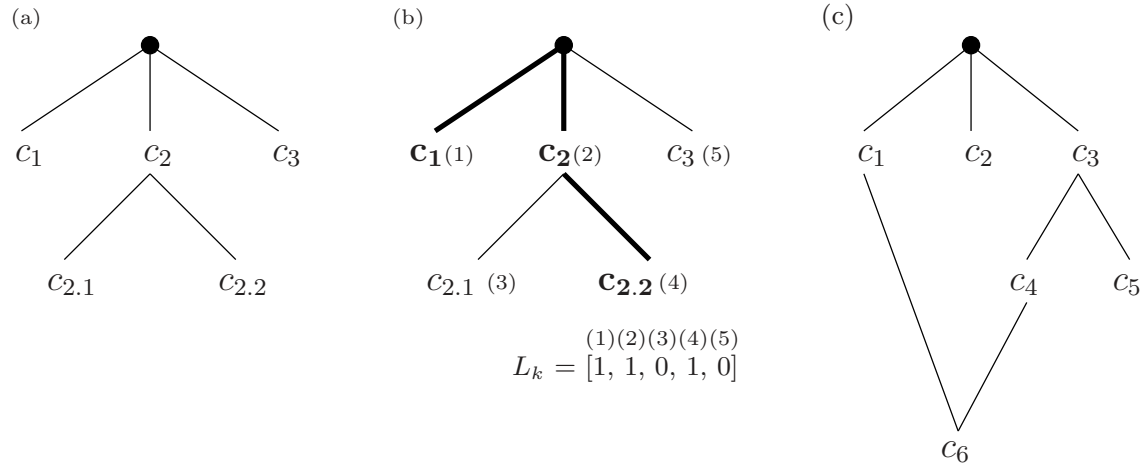


Figure 3.4: Toy examples of hierarchies structured as a tree and a DAG.(a) Class label names contain information about the position in the hierarchy, e.g., $c_{2.1}$ is a subclass of c_2. (b) The set of classes $\{c_1, c_2, c_{2.2}\}$, shown in bold in the hierarchy, represented as a vector. (c) A class hierarchy structured as a DAG. The class c_6 has two parents: c_1 and c_4.

Note that our definition of $w(c)$ allows the classes to be structured in a directed acyclic graph (DAG). Figure 3.4(c) depicts an example of a DAG structured hierarchy. In general, a DAG hierarchy can have two interpretations: if an example belongs to a given class c, it either a) also belongs to all super-classes of c, or b) belongs to at least one superclass of c. Here, we adapt the first case, i.e., the multiple inheritance interpretation.

The variance function used for tree-shaped hierarchies uses the weighted Euclidean distance between the class vectors, where the weight of a class depends on its depth in the hierarchy. When the hierarchy is a DAG, then the depth of a class is not unique: classes do not have a single path from the top-node (for example see class c_6 in Figure 3.4(c)).

To resolve this issue, Vens *et al.* (2008) suggest four aggregation schemes of the possible paths from the top-node to a given class: average, maximum, minimum and sum. The aggregation schemes use the observation that $w(c) = w_0^{depth(c)}$ can be rewritten as the recursive relation $w(c) = w_0 \cdot w(par(c))$, with $par(c)$ as the parent class of c, and the weights of the top-level classes equal to w_0. After an extensive experimental evaluation, Vens *et al.* (2008) recommend to use the average as aggregation function ($w(c) = w_0 \cdot \text{avg}_j\{w(par_j(c))\}$).

Calculation of the prediction

A classification tree stores in a leaf the majority class for that leaf, which will be the tree's prediction for all examples that will arrive in the leaf. In the case of HMC, an example may have multiple classes, thus the notion of 'majority class' does not apply in a straightforward manner. Instead, the mean $\bar{L}$ of the class vectors of the examples in the leaf is stored as a prediction. Note that the value for the i-th component of $\bar{L}$ can be interpreted as the probability that an example arriving at the given leaf belongs to class c_i.

The prediction for an example that arrives in the leaf can be obtained by applying a user defined threshold τ on the probability; if the i-th component of $\bar{L}$ is above τ then the examples belong to the class c_i. When a PCT is making a prediction it preserves the hierarchy constraint (the predictions comply to the parent child relationships from the hierarchy) if the values for the thresholds τ are chosen as follows: $\tau_i \leq \tau_j$ whenever $c_i \leq_h c_j$ (c_i is ancestor of c_j). The threshold is selected depending on the context. The user may set the threshold such that the resulting classifier has high precision at the cost of lower recall or vice versa, to maximize the F-score, to maximize the interpretability or plausibility of the resulting model etc. In this work, we use a threshold-independent measure (precision-recall curves) to evaluate the performance of the HMC models.

3.2 Ensembles of PCTs for predicting structured outputs

An ensemble is a set of predictive models (called base predictive models). In homogeneous ensembles, such as the ones we consider here, the base predictive models are constructed by using the same algorithm. The prediction of an ensemble for a new instance is obtained by combining the predictions of all the base predictive models from the ensemble. In this dissertation, we consider ensembles of PCTs for structured prediction. The PCTs in the ensembles are constructed by using the bagging and random forests approaches that are often used in the context of decision trees: We have adapted these approaches to use PCTs.

3.2.1 Constructing ensembles of PCTs

A necessary condition for an ensemble to have better predictive performance than any of its individual members, is that the base predictive models are accurate and diverse (Hansen and Salamon, 1990). An accurate predictive model does better than random guessing on new examples. Two predictive models are diverse if they make different errors on new examples. There are several ways to introduce diversity in a set of base predictive models: by manipulating the training set (by changing the weight of the examples (Breiman, 1996a; Freund and Schapire, 1996), by changing the attribute values of the examples (Breiman, 2001b), by manipulating the feature space (Breiman, 2001a; Ho, 1998)) and by manipulating the learning algorithm itself (Breiman, 2001a; Dietterich, 2000a).

We have implemented the bagging, random forests, random subspaces and bagging of subspaces methods within the CLUS system. The algorithms of these ensemble learning methods are presented in Table 3.2. For the random forests approach (top-right in Table 3.2), the PCT algorithm for structured prediction needed changes: A randomized version of the selection of attributes was implemented, which replaced the standard selection of attributes. However, in the empirical evaluation of these approaches, we only consider the two ensemble learning techniques that are most widely known and have primarily been used in the context of decision trees: bagging and random forests.

3.2.2 Bagging

Bagging (Breiman, 1996a) is an ensemble method that constructs the different classifiers by making bootstrap replicates of the training set and using each of these replicates to construct a predictive model (Table 3.2(top-left)). Each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set, until an equal number of instances as in the training set is obtained. Breiman (1996a) has shown that bagging can give substantial gains in predictive performance, when applied to an unstable learner (i.e., a learner for which small changes in the training set result in large changes in the predictions), such as classification and regression tree learners.

3.2.3 Random forests

A random forest (Breiman, 2001a) is an ensemble of trees, where diversity among the predictors is obtained by using bootstrap replicates as in bagging, and additionally by changing the feature set during learning (Table 3.2(top-right)). More precisely, at each

Table 3.2: The four ensemble induction algorithms: bagging, random forests, random subspaces and bagging of subspaces. Here, E is the set of the training examples, k is the number of trees in the forest, and $f(D)$ is the size of the feature subset that is used to learn the model (for random subspaces and bagging of subspaces) and that is considered at each node during tree construction (for random forests).

procedure Bagging(E, k)
returns Forest
1: $F = \emptyset$
2: **for** $i = 1$ **to** k **do**
3: $E_i = boostrap(E)$
4: $T_i = PCT(E_i)$
5: $F = F \bigcup T_i$
6: **return** F

procedure RForest($E, k, f(D)$)
returns Forest
1: $F = \emptyset$
2: **for** $i = 1$ **to** k **do**
3: $E_i = boostrap(E)$
4: $T_i = PCT\_rnd(E_i, f(D))$
5: $F = F \bigcup T_i$
6: **return** F

procedure RSubspaces($E, k, f(D)$)
returns Forest
1: $F = \emptyset$
2: **for** $i = 1$ **to** k **do**
3: $E_i = feature\_space(E)$
4: $T_i = PCT(E_i)$
5: $F = F \bigcup T_i$
6: **return** F

procedure BagSubspaces($E, k, f(D)$)
returns Forest
1: $F = \emptyset$
2: **for** $i = 1$ **to** k **do**
3: $E_t = boostrap(E)$
4: $E_i = feature\_space(E_t)$
5: $T_i = PCT(E_i, f(D))$
6: $F = F \bigcup T_i$
7: **return** F

node in the decision trees, a random subset of the descriptive attributes is taken, and the best feature is selected from this subset. The number of attributes that are retained is given by a function f of the total number of descriptive attributes D (e.g., $f(D) = 1$, $f(D) = \lfloor\sqrt{D} + 1\rfloor$, $f(D) = \lfloor log_2(D) + 1\rfloor$ ...). By setting $f(D) = D$, we obtain the bagging procedure. The algorithm for learning a random forest using PCTs as base classifiers is presented in Table 3.2.

3.2.4 Random subspaces

The random subspaces method (Ho, 1998) creates an ensemble by learning each of the base models on different feature subspaces (Table 3.2 (bottom-left)). This method first selects a subset of the descriptive attributes and then learns a base model using the dataset

that contains only these attributes. The number of descriptive attributes that are used to learn the base models, similarly as for random forests, is given with a function f of the total number of descriptive attributes D: Ho (1998) suggests that best results are obtained with $f(D) = \lfloor 0.5 \cdot D + 1 \rfloor$. The random subspaces method performs better when the number of descriptive attributes is large and when the number of examples is not small. This method is also more successful when redundant attributes are present.

3.2.5 Bagging of subspaces

The bagging of subspaces method (Panov and Džeroski, 2007) combines bagging and random subspaces. This method is outlined in Table 3.2 (bottom-right) and learns the base models as follows. It generates a training set for a base model by first creating a bootstrap replicate of the whole training set (similar as for bagging) and then by random selection of a subset of the descriptive attributes (similar as for random subspaces). The number of descriptive attributes used to learn the base models are given as a function f of the number of descriptive attributes D (similarly as for random forests and random subspaces): Panov and Džeroski (2007) evaluated the method using $f(D) = \lfloor 0.75 \cdot D + 1 \rfloor$. The performance of this method is comparable to the performance of random forests when decision trees are used as base models, while it performs statistically significantly better than bagging and random subspaces. Furthermore, this method can use any type of predictive model (such as decision trees, classification rules, nearest neighbors, etc) as base model.

3.2.6 Combining the predictions of individual PCTs

The prediction of an ensemble for a new instance is obtained by combining the predictions of all the base predictive models from the ensemble. The predictions from the models can be combined by taking the average (for regression tasks) and the majority or probability distribution vote (for classification tasks), as described in (Bauer and Kohavi, 1999; Breiman, 1996a), or by taking more complex aggregation schemes (Kuncheva, 2004).

We use predictive clustering trees as base predictive models for the ensembles for structured outputs. To obtain a prediction from an ensemble for predicting structured outputs, we accordingly extend the voting schemes. For the datasets with multiple continuous targets, as prediction of the ensemble, we take average of the predictions of the base classifiers. For the datasets for hierarchical classification we also use the average of the predictions and apply the thresholding described in Section 3.1.2. We obtain the ensemble predictions for the datasets with multiple discrete targets using probability distribution

voting (as suggested by Bauer and Kohavi (1999)) per target.

3.3 Local prediction of structured outputs with PCTs and ensembles

The presented structured output learning algorithms (CLUS-MTDT and CLUS-HMC) belong to the group of approaches known as 'big-bang' or global predictive models (Bakır *et al.*, 2007; Silla and Freitas, 2010). Global predictive models make a single prediction for the entire structured output, i.e., simultaneously predict all of its components. Local predictive models of structured outputs uses a collection of predictive models, each predicting a component of the overall structure that needs to be predicted.

The local predictive models for the task of predicting multiple targets are constructed by learning a predictive model for each of the targets separately. In the task of hierarchical multi-label classification, however, there are four approaches (Silla and Freitas, 2010): flat classification, local classifiers per level, local classifiers per node and local classifiers per parent node.

The first approach, flat classification, constructs a classifier for each leaf node from the hierarchy, typically, using one vs. all strategy (the examples belonging to a given leaf node from the hierarchy are labeled as positive, while the other examples as negative). In this approach, the classifiers are not aware of the hierarchical dependencies that exist between the classes and they are incapable of making a prediction for the non-leaf nodes from the hierarchy.

The second approach, local classifiers per level, constructs a classifier for each level from the hierarchy. This approach also requires a post-processing to solve the class-membership inconsistencies that may appear. It hasn't been used much by the community (only as a baseline comparison in Clare and King (2003) and Costa *et al.* (2007)).

The third approach, local classifiers per node, constructs a classifier for each node from the hierarchy except the root. This is the most widely used approach by the community. There are several policies for selection of the positive and negative examples that will be used to train the local classifiers (for details, see (Ceci and Malerba, 2007; Eisner *et al.*, 2005)).

The last approach, local classifiers per parent node, constructs a classifier for each non-leaf node from the hierarchy. One can learn a multi-class classifier for each parent node or transform the problem using One-Against-All scheme and then use binary classifiers for each child node (i.e., construct a classifier for each edge in the hierarchy).

Vens *et al.* (2008) investigated the performance of the last two approaches with local classifiers over a large set of datasets from functional genomics. The conclusion of the study was that the last approach performs better in terms of predictive performance, smaller total model size and faster induction times.

In particular, the CLUS-HSC algorithm by Vens *et al.* (2008), presented in Figure 3.5, constructs a decision tree classifier for each edge (connecting a class c with a parent class $par(c)$) in the hierarchy, thus creating an architecture of classifiers. The corresponding tree predicts membership to class c, using the instances that belong to $par(c)$. The construction of this type of tree uses few instances: only instances labeled with $par(c)$ are used for training. The instances labeled with class c are positive instances, while the ones that are labeled with $par(c)$, but not with c are negative.

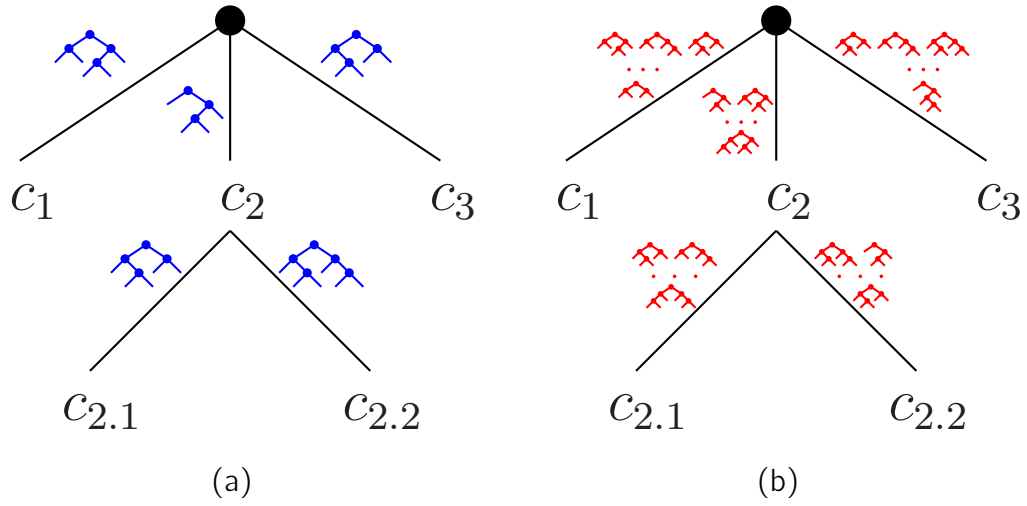


Figure 3.5: An illustration of the hierarchical single-label classification approach used by Vens *et al.* (2008). The local classifiers at each branch from the hierarchy are: (a) decision trees and (b) ensembles of decision trees.

The resulting HSC tree predicts the conditional probability $P(c|par(c))$. A new instance is predicted by recursive application of the product rule $P(c) = \min_j P(c\,|\,par_j(c)) \cdot P(par_j(c))$ (with $par_j(c)$ denoting the j-th parent of c in case of a DAG), starting from the tree for a top-level class. Again, the probabilities are thresholded to obtain the set of predicted classes. To satisfy the hierarchy constraint, the threshold τ should be chosen as in the case of CLUS-HMC.

In this thesis, we extend the approach of Vens *et al.* (2008) by applying ensembles as local classifiers instead of decision trees. The CLUS-HSC algorithm can be applied to ensemble learning in two ways by constructing: ensemble of architectures or architecture of ensembles. The first approach creates the ensemble by creating multiple architectures

(as shown in Figure 3.5(a)). These multiple architectures can be created on different bootstrap replicates, different feature spaces, different local classifiers etc. The second approach is simpler and, instead of a single local classifier (for example a decision tree), uses an ensemble as classifier at each branch (depicted in Figure 3.5(b)). We prefer here the second approach since it is closer to the learning of local classifiers for the prediction of multiple targets task.

In Chapter 4, we will compare global predictive models to collections of local predictive models. Single PCTs for structured prediction (global predictive models) will be compared to collections of PCTs for the components of the output (local predictive models). Ensembles of PCTs (global predictive models) will be compared to collections of PCT ensembles for the components of the output (local predictive models).

4 Experimental design and results

4.1 Experimental design

In this section, we describe the procedure for experimental evaluation of the proposed ensemble methods for predicting structured outputs. First, we state the questions we consider. Next, we present the datasets we use to evaluate the algorithms, and then the evaluation measures we applied. In the last subsection, we give the parameter values used in the algorithms and the statistical tests that we used.

4.1.1 Experimental questions

Given the methodology from Chapter 3, we construct several types of trees and ensembles. First, we construct PCTs that predict components of the structured output: a separate tree for each variable from the target tuple (CLUS-STDT) and a separate tree for each hierarchy edge (CLUS-HSC). Second, we learn PCTs that predict the entire structured output simultaneously: a tree for the whole target tuple (CLUS-MTDT) and a tree for the whole hierarchy (CLUS-HMC). Finally, we construct the ensemble classifiers in the same manner (CLUS-ENS-ST, CLUS-ENS-HSC, CLUS-ENS-MT, CLUS-ENS-HMC) by using both bagging and random forests.

We consider the following questions:

- *Predictive performance*: Can exploitation of the structure of the output lift the predictive performance of an ensemble?
- *Convergence*: Does the performance of the ensembles for structured outputs converge/saturate faster than ensembles that predict components of the output?
- *Suitability*: Which ensemble method should be preferred given the size of a dataset, as measured by the number of instances, the number of descriptive attributes and the size of the structured output?
- *Efficiency*: How much can the learning process benefit, in terms of time and memory consumption, from the ensembles for structured outputs as compared to the sets of

ensembles that predict components of the structured output?

We compare the algorithms that predict the complete structured output (CLUS-MTDT, CLUS-HMC, CLUS-ENS-MT, CLUS-ENS-HMC) to the algorithms that predict the components of the structured outputs separately (CLUS-STDT, CLUS-HSC, CLUS-ENS-ST, CLUS-ENS-HSC). First, we inspect the predictive performance of all algorithms. Then, we focus only on the ensembles and examine their predictive performance at different ensemble sizes (i.e., we construct 'saturation curves'). Our intention is to investigate whether the performance of the ensembles for structured outputs saturates at a smaller number of trees as compared to the saturation of ensembles that predict the components of the structured outputs. At the end, we compare the running times and the sizes of the obtained models.

4.1.2 Descriptions of the datasets

In this subsection, we present the datasets that were used to evaluate the predictive performance of the ensembles. The datasets are divided into three groups of datasets based on the type of their targets: multiple continuous targets datasets (regression), multiple discrete targets datasets (classification) and hierarchical multi–label classification datasets (HMC). Statistics about the used datasets are presented in Tables 4.1, 4.2, and 4.3, respectively.

The datasets with multiple continuous targets (13 in total, see Table 4.1) are mainly from the domain of ecological modelling. The datasets with multiple discrete targets (9 in total, see Table 4.2) are from various domains: ecological modelling (*Sigmea Real* and *Water Quality*), biological (*Yeast*), multimedia (*Scene* and *Emotions*), media space (*Mediana*), etc. The datasets that have classes organized in a hierarchy come from various domains, such as: biology (*Expression-FunCat, SCOP-GO, Yeast-GO* and *Sequence-FunCat*), text classification (*Enron, Reuters* and *WIPO*) and image annotation/classification (*ImCLEF07D, ImCLEF07A* and *Diatoms*). Hence, we use 10 datasets from 3 domains (see Table 4.3). Note that two datasets from the biological domain have a hierarchy organized as a DAG (they have GO in the dataset name), and the remaining datasets have tree-shaped hierarchies. For more details on the datasets, we refer the reader to the referenced literature.

4.1.3 Evaluation measures

Empirical evaluation is the most widely used approach for assessing the performance of machine learning algorithms. The performance of a machine learning algorithm is com-

Table 4.1: Properties of the datasets with multiple continuous targets (regression datasets); N is the number of instances, $\overline{D/C}$ the number of descriptive attributes (discrete/continuous), and T the number of target attributes.

Name of dataset	N	$\overline{D/C}$	T
Collembola (Kampichler *et al.*, 2000)	393	8/39	3
EDM (Karalič, 1995)	154	0/16	2
Forestry–Kras (Stojanova *et al.*, 2010)	60607	0/160	11
Forestry–Slivnica-LandSat (Stojanova, 2009)	6218	0/150	2
Forestry–Slivnica-IRS (Stojanova, 2009)	2731	0/29	2
Forestry–Slivnica-SPOT (Stojanova, 2009)	2731	0/49	2
Sigmea real (Demšar *et al.*, 2005)	817	0/4	2
Soil quality (Demšar *et al.*, 2006)	1944	0/142	3
Solar–flare 1 (Asuncion and Newman, 2007)	323	10/0	3
Solar–flare 2 (Asuncion and Newman, 2007)	1066	10/0	3
Vegetation Clustering (Gjorgjioski *et al.*, 2008)	29679	0/65	11
Vegetation Condition (Kocev *et al.*, 2009)	16967	1/39	7
Water quality (Blockeeel *et al.*, 1999; Džeroski *et al.*, 2000)	1060	0/16	14

Table 4.2: Properties of the datasets with multiple discrete targets (classification datasets); N is the number of instances, $\overline{D/C}$ the number of descriptive attributes (discrete/continuous), and T the number of target attributes.

Name of dataset	N	$\overline{D/C}$	T
EDM (Karalič, 1995)	154	0/16	2
Emotions (Trohidis *et al.*, 2008)	593	0/72	6
Mediana (Skrjanc *et al.*, 2001)	7953	21/58	5
Scene (Boutell *et al.*, 2004)	2407	0/294	6
Sigmea real (Demšar *et al.*, 2005)	817	0/4	2
Solar–flare 1 (Asuncion and Newman, 2007)	323	10/0	3
Thyroid (Asuncion and Newman, 2007)	9172	22/7	7
Water quality (Blockeeel *et al.*, 1999; Džeroski *et al.*, 2000)	1060	0/16	14
Yeast (Elisseeff and Weston, 2001)	2417	0/103	14

puted using some evaluation measure. The different machine learning tasks, described in Section 2.1, use 'task-specific' evaluation measures. We first describe the evaluation

Table 4.3: Properties of the datasets with hierarchical targets; N_{tr}/N_{te} is the number of instances in the training/testing dataset, D/C is the number of descriptive attributes (discrete/continuous), $|\mathcal{H}|$ is the number of classes in the hierarchy, $\mathcal{H}_d$ is the maximal depth of the classes in the hierarchy, $\overline{\mathcal{L}}$ is the average number of labels per example, and $\overline{\mathcal{L}_L}$ is the average number of leaf labels per example.

Domain	N_{tr}/N_{te}	D/C	$\lvert\mathcal{H}\rvert$	$\mathcal{H}_d$	$\overline{\mathcal{L}}$	$\overline{\mathcal{L}_L}$
ImCLEF07D(Dimitrovski *et al.*, 2008)	10000/1006	0/80	46	3.0	3.0	1.0
ImCLEF07A(Dimitrovski *et al.*, 2008)	10000/1006	0/80	96	3.0	3.0	1.0
Diatoms (ADIAC, 2008)	2065/1054	0/371	377	3.0	1.95	0.94
Enron (Klimt and Yang, 2004)	988/660	0/1001	54	3.0	5.30	2.84
Reuters (Lewis *et al.*, 2004)	3000/3000	0/47236	100	4.0	3.20	1.20
WIPO (Rousu *et al.*, 2006)	1352/358	0/74435	183	4.0	4.0	1.0
Expression–FunCat (Clare, 2003)	2494/1291	4/547	475	4.0	8.87	2.29
SCOP–GO (Clare, 2003)	6507/3336	0/2003	523	5.5	6.26	0.95
Sequence–FunCat (Clare, 2003)	2455/1264	2/4448	244	4.0	3.35	0.94
Yeast–GO (Barutcuoglu *et al.*, 2006)	2310/1155	5588/342	133	6.3	5.74	0.66

measures for multiple continuous targets (regression), then for multiple discrete targets (classification) and at the end for hierarchical classification.

For the task of predicting multiple continuous targets (regression), we employed three well known measures: the correlation coefficient (CC), root mean squared error ($RMSE$) and relative root mean squared error ($RRMSE$). For each of these measures, we performed statistical analysis and constructed saturation curves. We present only the results in terms of $RRMSE$, but same conclusions hold for the other two measures.

What evaluation measure to use in the case of classification algorithms is not as clear as in the case of regression. Sokolova and Lapalme (2009) conducted a systematic analysis of twenty four performance measures that can be used in a classification context. They conclude that evaluation measures for classification algorithms should be chosen based on the application domain.

In our study, we used seven evaluation measures for classification: accuracy, precision, recall, F-score, the Matthews correlation coefficient, balanced accuracy (also known as Area Under the Curve) and discriminant power. We used two averaging approaches to adapt these measures for multi-class problems: micro and macro averaging (note that averaging is not needed for accuracy). More about these measures can be found in Sokolova *et al.* (2006). Since the goal of this study is not to assess the evaluation

measures themselves, we present here only the results in terms of the micro average F-score ($F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$). However, the conclusions drawn from the evaluation of the performance of the algorithms using the other measures concur with the ones presented here.

In the case of hierarchical classification, we evaluate the algorithms using the Area Under the Precision-Recall Curve ($AUPRC$), and in particular, the Area Under the Average Precision-Recall Curve ($AU\overline{PRC}$) as suggested by Vens *et al.* (2008). A Precision-Recall curve plots the precision of a classifier as a function of its recall. The points in the PR space are obtained by varying the value for the threshold τ from 0 to 1 with step 0.02. The precision and recall are micro averaged for all classes from the hierarchy.

In these domains, the positive examples for a given class are only few as compared to the negative ones. The PR evaluation of these algorithms is most suitable in this context because we are typically more interested in recognizing the positive examples (i.e., that an example belongs to a given class), rather than correctly predicting negative instances.

Finally, we compare the algorithms by measuring their efficiency in terms of time consumption and size of the models. We measure the processor time needed to construct the models: in the case of predicting the components of the structure, we sum the times needed to construct the separate models. In a similar way, we calculated the sizes of the models as the total number of nodes (internal nodes and leafs). The experiments for predicting multiple targets were performed on a server running Linux, with two Intel Quad-Core Processors@2.5GHz and 64GB of RAM. The experiments for the hierarchical classification were run on a cluster of AMD Opteron processors (1.8 – 2.4GHz, $\geq$ 2GB RAM).

4.1.4 Experimental setup

Here, we first state the parameter values used in the algorithms for constructing the single trees and the ensembles for all types of targets. We then describe how we assessed the statistical significance of the differences in performance of the studied algorithms.

The single trees for all types of targets are obtained using F-test pruning. This pruning procedure uses the exact Fisher test to check whether a given split/test in an internal node of the tree produces a reduction in variance that is statistically significant at a given significance level. If there is no split/test that can satisfy this, then the node is converted to a leaf. An optimal significance level was selected by using internal 3-fold cross validation, from the following values: 0.125, 0.1, 0.05, 0.01, 0.005 and 0.001.

The construction of an ensemble takes the the size of the ensemble as an input parameter: number of base predictive models to be constructed. We constructed ensembles with

10, 25, 50, 75 and 100 base predictive models for all types of targets and all datasets. In addition, for the datasets with multiple continuous targets we constructed ensembles with 150 and 250 base predictive models, and for the datasets with multiple discrete targets ensembles with 250, 500 and 1000 base predictive models. Following the findings from the study conducted by Bauer and Kohavi (1999), the trees in the ensembles were not pruned.

The random forests algorithm takes as input the size of the feature subset that is randomly selected at each node. For the multiple targets datasets, we apply the logarithmic function of the descriptive attributes $\lfloor \log_2 DescriptiveAttributes \rfloor + 1$, which is recommended by Breiman (2001a). For the hierarchical classification datasets, we used $\lfloor 0.1 \cdot DescriptiveAttributes \rfloor + 1$, since the feature space of some of these datasets is large (several thousands of features) and the logarithmic function is under-sampling the feature space.

On the datasets with multiple targets, the predictive performance of the algorithms is estimated by 10-fold cross-validation. The hierarchical datasets were previously divided (by the data providers) into train and test sets. Thus, we estimate the predictive performance of the algorithms on the test sets.

We adopt the recommendations by Demšar (2006) for the statistical evaluation of the obtained results. We use the Friedman test (Friedman, 1940) for statistical significance with the correction from Iman and Davenport (1980). Afterwards, to check where the statistically significant differences appear (between which algorithms), we use the Nemenyi post-hoc test (Nemenyi, 1963). We present the results from the statistical analysis with 'average ranks diagrams' (see Figures 4.3, 4.4, 4.7, 4.8, 4.11 and 4.12).

4.2 Results and discussion

The results from the experiments we performed can be analyzed along several dimensions. First, we present the saturation curves of the ensemble methods (both for predicting the structured output and the components). Then, we compare models that predict the complete structured output vs. models that predict components of the structured output. Next, we compare the performance of single trees and ensembles of trees. At the end, we evaluate the algorithms by their efficiency in terms of running time and model size. We make these comparisons for each task separately: predicting multiple continuous targets, predicting multiple discrete targets and hierarchical multi–label classification.

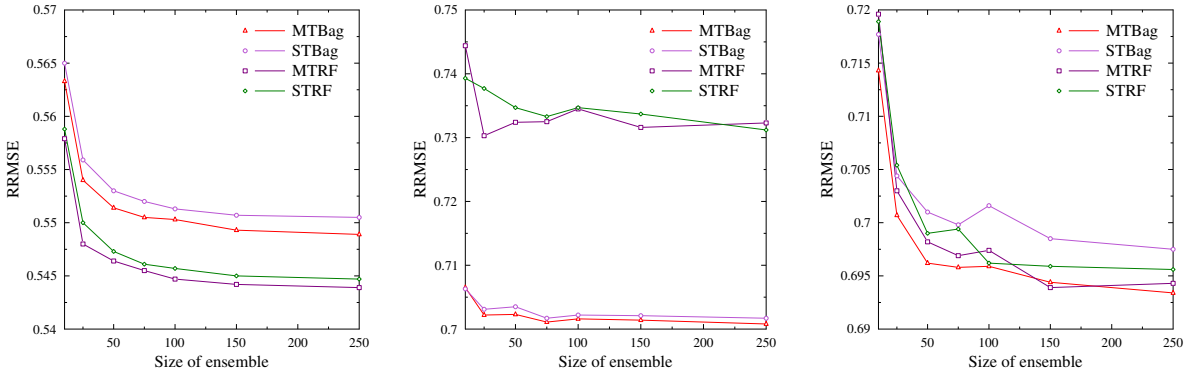
4.2.1 Multiple continuous targets

The results from the experiments for evaluating the algorithms on the task of prediction of multiple continuous targets are presented in Figures 4.1, 4.3 and 4.4. First, we discuss the results with respect to the saturation curves (Figure 4.1). Next, we discuss the statistical comparison of predictive performance (Figure 4.3). Finally, we compare the efficiency of the algorithms (Figure 4.4).

In Figure 4.1, we present the saturation curves for the ensemble methods. Although these curves are averaged across all target variables for a given dataset (and in Figure 4.1(c) averaged across all datasets), they still provide useful insight into the performance of the algorithms. Random forests perform better than bagging, both when predicting the multiple targets simultaneously and separately, on the 'larger' datasets (the ones with more than 10000 examples), such as *Forestry-Kras* from Figure 4.1(a). On the other hand, bagging outperforms random forests, in both scenarios, on the 'medium' datasets (that contain between 1000 and 10000 examples), such as *Soil quality* from Figure 4.1(b). For the 'small' datasets (the ones with less than 1000 examples and less than 10 descriptive attributes), the curves are variable and it is not clear which algorithm should be preferred. Also, there is no clear connection between the performance of the algorithms and the number of target variables (i.e., the size of the target tuple). However, on the majority of all datasets the ensembles for prediction of multiple targets simultaneously perform better than the ensembles that predict the targets separately.

The saturation curves averaged across all datasets are shown in Figure 4.1(c). They show that the ensembles for predicting multiple targets simultaneously perform better than the ones predicting the targets separately across all ensemble sizes (except with 100 trees where random forests for multiple targets are worse than random forests for single targets). The saturation point in such a curve is the ensemble size after which the difference in performance achieved by increasing the ensemble size is no longer statistically significant. To detect the saturation point, we perform Friedman and Nemenyi tests for assessment of statistical significance for each method/algorithm separately. Figure 4.2 shows the results of these tests. In this case, for each of the algorithms, the difference is not statistically significant after 50 trees. Thus, we compare the performance of the ensembles with 50 and with 250 trees (the maximal number of trees).

The statistical tests (illustrated by the average rank diagrams in Figure 4.3) show that the difference in performance of the ensemble methods is not statistically significant at the 0.05 level. The best performing method is random forests for predicting multiple targets simultaneously (average rank 2.53) and the worst performing method is bagging for predicting the multiple targets separately (average rank 3.11). If more trees are added,



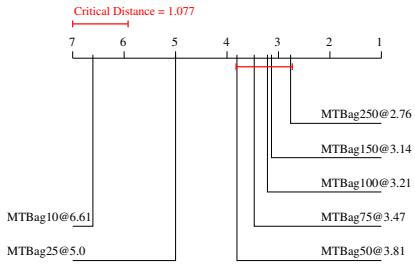
(a) *Forestry-Kras* (b) *Soil quality* (c) *Overall*

Figure 4.1: Saturation curves for the different ensemble approaches to the prediction of multiple continuous targets. These curves are obtained by averaging the $RRMSE$ values over all target variables in a dataset. Smaller $RRMSE$ values mean better predictive performance. The algorithm names are abbreviated as follows: random forests for the prediction of multiple targets – $MTRF$, random forests for the prediction of single targets – $STRF$, bagging for the prediction of multiple targets – $MTBag$ and bagging for the prediction of single targets – $STBag$.

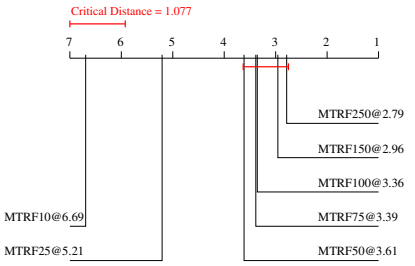
the performance ordering of the algorithms does not change (only small changes appear in the average ranks). The difference in performance between all ensembles and the single trees is statistically significant at the 0.05 level. The single trees for predicting multiple targets simultaneously are better than the single trees for predicting the multiple targets separately.

Finally, we compare the algorithms by their running time and the size of the models, considering ensembles that consist of 50 trees (see Figure 4.4). The statistical tests show that both random forests and bagging for predicting multiple targets simultaneously outperform significantly, in terms of model size, the ensembles that predict multiple targets separately. In terms of running time, random forests for multiple targets outperform significantly both ensemble methods for predicting the targets separately. Also, bagging for multiple targets is significantly faster than bagging for separate prediction of the targets.

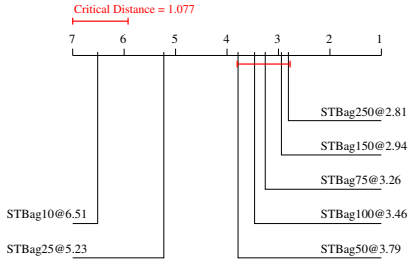
Let us further examine the speed-up and model size ratios. Random forests for predicting multiple targets simultaneously are $\sim$3.3 times faster to construct and the models are $\sim$3.75 times smaller than the random forests predicting single targets. In addition, they are $\sim$3.7 times faster to construct and yield $\sim$1.14 times smaller models as compared to bagging for multiple targets. Furthermore, bagging for predicting multiple targets is $\sim$3 times faster and yields $\sim$3.6 times smaller models than bagging for predicting single targets.



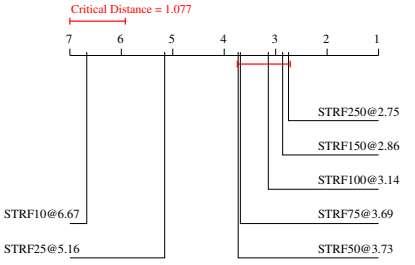
(a) Bagging for multiple targets



(b) Random forests for multiple targets

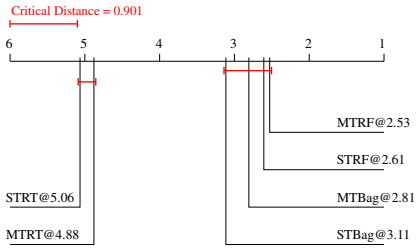


(c) Bagging for single target

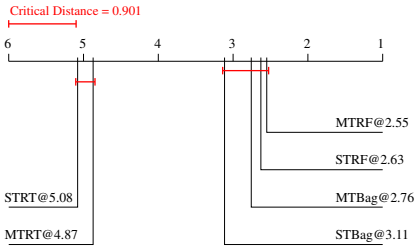


(d) Random forests for single target

Figure 4.2: Average rank diagrams (with the critical distance at a significance level of 0.05) for detection of the saturation points for the prediction of multiple continuous targets. The differences in performance of the algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 4.1.



(a) ensemble with 50 trees



(b) ensemble with 250 trees

Figure 4.3: Average rank diagrams (with the critical distance at a significance level of 0.05) for the prediction of multiple continuous targets. The differences in performance of the algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 4.1 with the addition of predicting clustering trees for multiple continuous targets – $MTRT$ and predictive clustering trees for single continuous targets – $STRT$.

To summarize, ensembles for predicting multiple continuous targets simultaneously perform better than ensembles predicting multiple targets separately. While the differences

(a) *Time efficiency* (b) *Size of the models*

Figure 4.4: Efficiency (running time and model size) of the ensembles for prediction of multiple continuous targets. The size of the ensembles is 50 trees.

in predictive performance are not statistically significant, the differences in efficiency are. Random forests have higher predictive performance than bagging on the larger datasets, while on the medium datasets bagging ensembles are better. In terms of efficiency, the ensembles , especially random forests, that predict multiple targets simultaneously are significantly better.

4.2.2 Multiple discrete targets

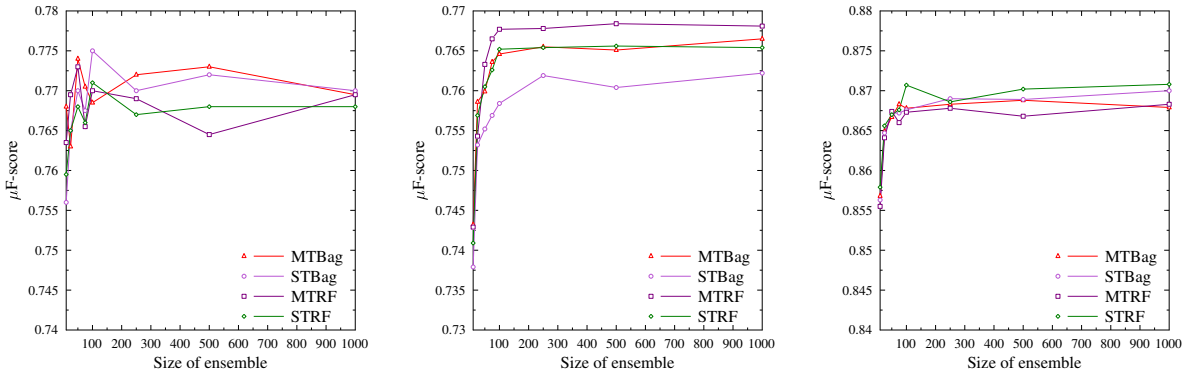
The performance of the algorithms for multi-class classification can be assessed using different measures, some of which we listed in Section 4.1.3. The selected evaluation measure should be appropriate for the application domain (Sokolova and Lapalme, 2009). In our study, we used the micro weighted averaged F-score ($\mu F - score$). We believe this is a reasonable compromise between all the considered measures, since it combines precision and recall.

The results for algorithms that predict multiple discrete targets are presented in Figures 4.5, 4.7 and 4.8. In Figure 4.5, we present the saturation curves. Next, we discuss the statistical analysis of the results (Figure 4.7). At the end, we compare the algorithms in terms of efficiency (Figure 4.8).

In Figure 4.5, we present three saturation curves for the four ensemble methods. As for predicting multiple continuous targets, these values are averaged over all target variables for a given dataset (and in Figure 4.5(c) averaged across all datasets). These saturation curves offer us several insights into the performance of the ensembles on the task of predicting multiple discrete targets. The saturation curves for the smaller datasets (with less than 1000 examples) are variable (for instance, see the saturation curve for the *Sigmea real* dataset shown in Figure 4.5(a)). However, we can note that, for smaller ensemble sizes, the ensembles that predict the targets simultaneously outperform the ensembles that predict the targets separately.

The saturation curves for the larger datasets (with more than 1000 examples) are more

stable and we can observe two types of behavior: (1) on the datasets with less than 30 descriptive variables, the ensembles for predicting the targets simultaneously outperform the ensembles that predict the targets separately (for instance, see the saturation curve for the *Water quality* dataset shown in Figure 4.5(b)); (2) on the datasets with more than 30 descriptive variables, the ensembles for predicting the targets simultaneously are better when the size of the ensemble is small than the ensembles that predict the multiple targets separately, while on the ensembles with bigger sizes the situation is reversed. Similar behavior can be also noticed on the *Overall* saturation curve (Figure 4.5(c)). Finally, as for the multiple continuous targets, there is no connection between the predictive performance of the algorithms and the size of the target tuple.

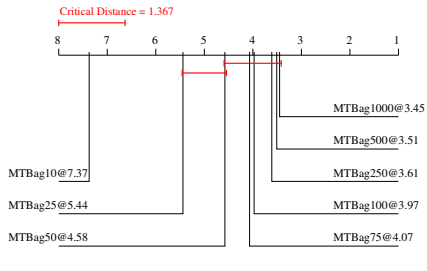


(a) *Sigmea real* (b) *Water quality* (c) *Overall*

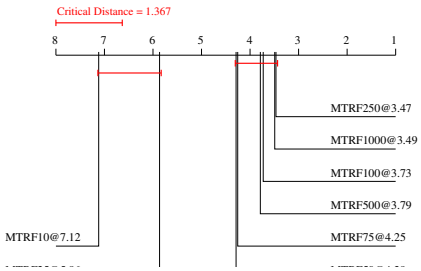
Figure 4.5: Saturation curves for the prediction of multiple discrete targets. These curves are obtained by averaging the $\mu F - score$ values for all of the target variables in a dataset. Larger $\mu F - score$ values mean better predictive performance. The algorithm names are abbreviated as follows: random forests for the prediction of multiple targets – $MTRF$, random forests for the prediction of single targets – $STRF$, bagging for the prediction of multiple targets – $MTBag$ and bagging for the prediction of single targets – $STBag$.

For each ensemble method separately, we check at which ensemble size the predictive performance saturates, i.e., the difference in performance due to increasing the size of the ensemble is no longer statistically significant. The results from the Friedman and Nemenyi tests for statistical significance are given in Figure 4.6. The ensembles for predicting multiple targets simultaneously saturate at 50 trees, while the ensembles for separate prediction of the targets require more trees: 75 for random forests and 250 for bagging. Considering these results, we select the ensembles sizes of 50 and 1000 (maximal number of trees) and compare the algorithms.

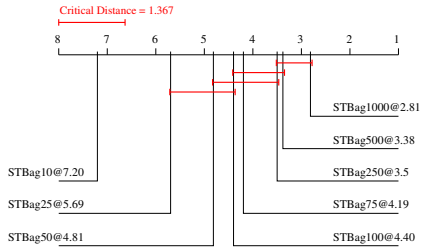
The results from the statistical analysis of the predictive performance ($\mu F - score$) are shown in Figure 4.7. The statistical tests reveal that there is no statistically significant



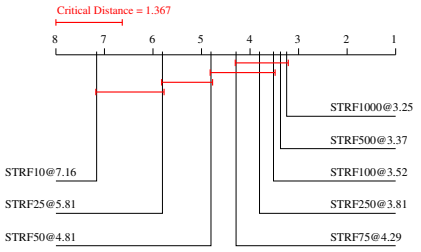
(a) Bagging for multiple targets



(b) Random forests for multiple targets



(c) Bagging for single target

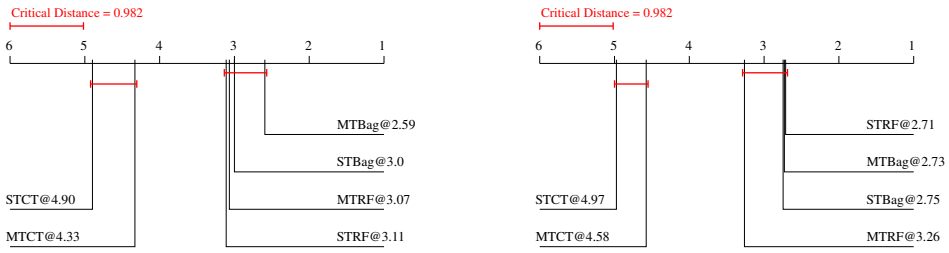


(d) Random forests for single target

Figure 4.6: Average rank diagrams (with the critical distance at a significance level of 0.05) for detection of the saturation points for the prediction of multiple discrete targets. The differences in performance of the algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 4.5.

difference in the performance of the ensemble methods and that all ensemble methods perform statistically significantly better than the corresponding single trees. When the ensembles have 50 trees (Figure 4.7(a)), bagging for predicting multiple targets simultaneously is the best performing method (average rank 2.59): the remaining methods have slightly smaller and very similar average ranks (ranging from 3.0 to 3.11), with random forest for separate prediction of the targets having the largest average rank. The situation is similar with 1000 trees (Figure 4.7(b)), with the difference that now random forests for simultaneous prediction are the worst performing method (average rank 3.26) and the other three methods have essentially the same average ranks (from 2.71 to 2.75), with random forests for separate prediction being the best performing method. This confirms the conclusions of the analysis of with the saturation curves: adding trees helps more for ensembles that predict the targets separately than for ensembles that predict the targets simultaneously.

At the end, we compare the ensembles in terms of efficiency: running times (Figure 4.8(a)) and model sizes (Figure 4.8(b)). Concerning the running time, we can only state that the random forests for predicting multiple targets simultaneously significantly



(a) ensembles with 50 trees (b) ensembles with 1000 trees

Figure 4.7: Average ranks diagrams (with the critical distance at significance level of 0.05) for prediction of multiple discrete targets. The differences in performance of the algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 4.5, with the addition of a single predicting clustering tree for multiple discrete targets – $MTCT$ and predictive clustering trees for each single discrete target – $STCT$.

outperform bagging for predicting the multiple targets separately. As for the size of the models, we can note the following: (1) bagging for predicting multiple targets simultaneously significantly outperforms both ensemble methods for separate prediction of the targets and (2) random forests for predicting multiple targets simultaneously significantly outperform random forests for separate prediction of the targets.

(a) *Time efficiency* (b) *Size of the models*

Figure 4.8: Efficiency of the ensembles for prediction of multiple discrete targets. The size of the ensembles is 50 trees.

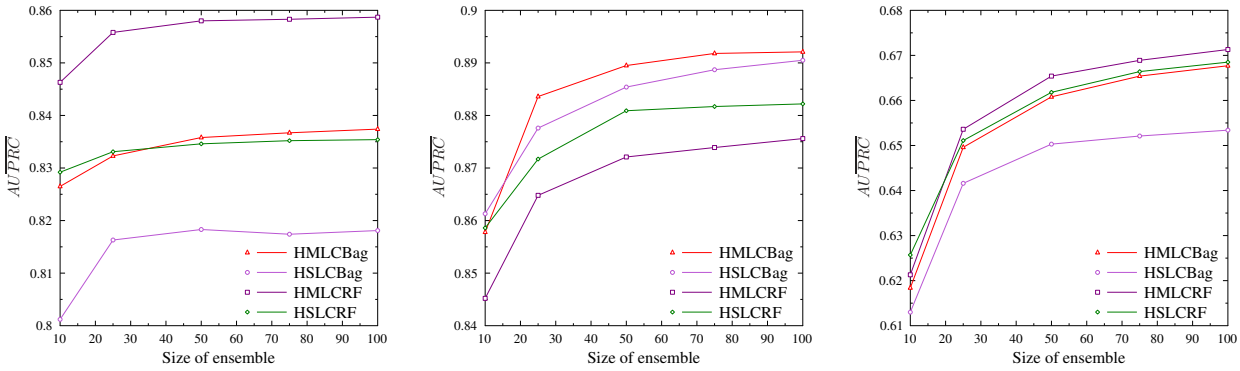
We further investigate the running times and model size ratios. Random forests for predicting multiple targets simultaneously are ∼2.3 times faster to construct and have ∼2.1 times smaller models than random forests for separate prediction of the targets. Also, they are ∼5.6 times faster and have ∼1.14 times larger models than bagging for predicting multiple targets simultaneously. Furthermore, bagging for predicting multiple targets simultaneously is ∼2.5 times faster and has ∼1.9 times smaller models than bagging for separate prediction of multiple targets.

In summary, the predictive performances of the ensemble methods for predicting multiple targets simultaneously and the ones for separate prediction are not statistically signif-

icantly different. However, the ensemble methods for predicting multiple targets simultaneously are better when the number of trees in the ensemble is smaller. Furthermore, they should be preferred if the efficiency of the classifier is an issue. The ensemble methods for simultaneous prediction are faster (especially random forests) and produce smaller models (especially bagging) than the ensemble methods for separate prediction.

4.2.3 Hierarchical multi–label classification

In this subsection, we present the results for the task of hierarchical classification in a similar way as for the task of predicting multiple targets. We asses the performance of the algorithms using the area under the average precision-recall curve ($AU\overline{PRC}$) as suggested by Vens *et al.* (2008). The results are presented with saturation curves (Figure 4.9), statistical tests (Figure 4.11) and efficiency figures (Figure 4.12).



(a) *SCOP-GO* (b) *ImCLEF07D* (c) *Overall*

Figure 4.9: Saturation curves for hierarchical multi–label classification. These curves are obtained by averaging the $AU\overline{PRC}$ values for all of the target variables. Larger $AU\overline{PRC}$ values mean better predictive performance. The algorithm names are abbreviated as follows: random forests for hierarchical multi–label classification – $HMCRF$, random forests for hierarchical single-label classification – $HSCRF$, bagging for hierarchical multi–label classification – $HMCBag$ and bagging for hierarchical single-label classification – $HSCBag$.

The saturation curves for the different domains (functional genomics, image annotation and text classification) show different behavior, thus we discuss the curves for each domain separately. In the domain of functional genomics, the ensembles for HMC outperform the ensembles for HSC when the target hierarchy is organized as a DAG (for instance, see the saturation curve for the *SCOP-GO* dataset in Figure 4.9(a)). Moreover, random forests for HMC are the best performing method. The ensembles for HMC also outperform the

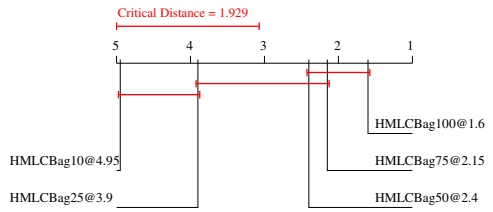
ensembles for HSC on the domain of image annotation/classification (for instance, see the saturation curve for the *ImCLEF07D* dataset in Figure 4.9(b)). On these datasets, bagging for HMC is the best performing method. The situation is different in the text classification domains. Here, the ensembles of HSC outperform the ensembles of HMC. We hypothesize that this is because of the large number of descriptive variables. The performance of ensembles of HMC on text classification datasets should be further investigated.

Next, we relate the performance to the properties of the datasets. First, on the datasets that have on average more than 5 labels per instance ($\overline{\mathcal{L}} > 5$), random forests perform better than bagging in both cases (HMC and HSC). On the datasets with less than 3 labels per instance ($\overline{\mathcal{L}} < 3$), bagging for HMC is better than random forests for HMC. Next, on the datasets with larger hierarchies ($|\mathcal{H}| > 300$), the ensembles for HMC outperform the ensembles of HSC. On the datasets with smaller hierarchies ($|\mathcal{H}| < 100$) random forests perform better than bagging. The ensembles for HMC also outperform the ensembles for HSC when the number of descriptive attributes is smaller than 1000. There are no clear advantages of any one ensemble method on the datasets based on the number of instances available for training.

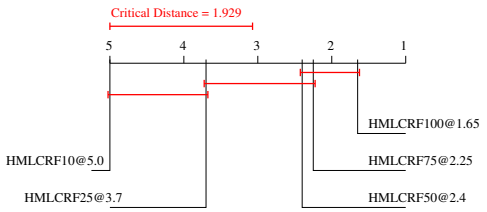
The overall saturation curve (Figure 4.9(c)) shows the performance of the algorithms averaged over all datasets from the three domains. The best performing method is random forests for HMC and the worst performing method is bagging for HSC. To further investigate the differences in performance, we perform statistical analysis for each method separately across all ensemble sizes. We do this to determine the saturation point, i.e, to check when adding extra trees to the ensemble does not statistically significantly improve predictive performance any more. The results from Friedman and Nemenyi tests for assessment of the statistical significance of the difference in performance are shown in Figure 4.10. The ensembles for HMC and random forests for HSC saturate after 50 trees, while bagging for HSC saturates after only 25 trees. We further compare the performance of the ensembles at 50 trees and 100 trees.

The average ranks diagram for the ensembles with 50 trees (Figure 4.11(a)) shows that the performance of the ensembles is not statistically significantly different. Note that the best performing method is random forests for HSC (average rank 2.25) and the worst performing method is bagging for HSC (average rank 2.85). Similarly, there is no statistically significant difference in performance when the ensembles contain 100 trees. Again, bagging for HSC (average rank 2.9) is the worst performing method, but bagging for HMC (average rank 2.2) is now the best performing method. In both cases, the ensemble methods significantly outperform single predictive clustering trees.

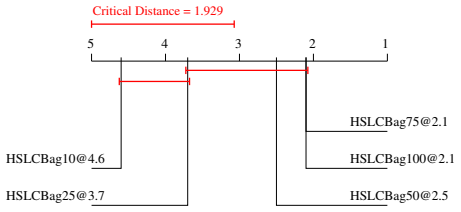
Finally, we compare the algorithms by their efficiency when they contain 50 trees



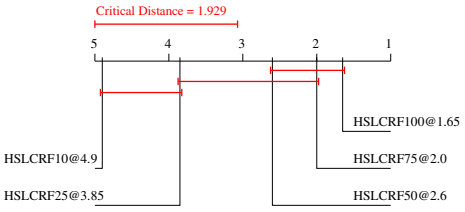
(a) Bagging for HMC



(b) Random forests for HMC

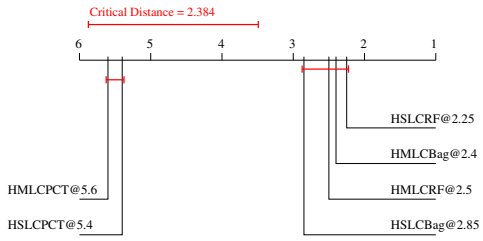


(c) Bagging for HSC

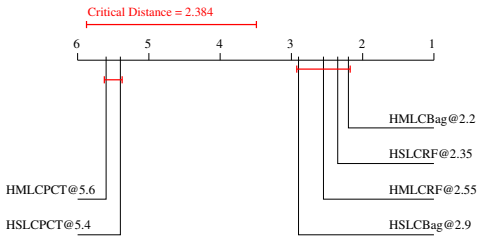


(d) Random forests for HSC

Figure 4.10: Average rank diagrams (with the critical distance at a significance level of 0.05) for detection of the saturation points for hierarchical multi-label classification. The differences in performance of the algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 4.9.



(a) ensemble with 50 trees



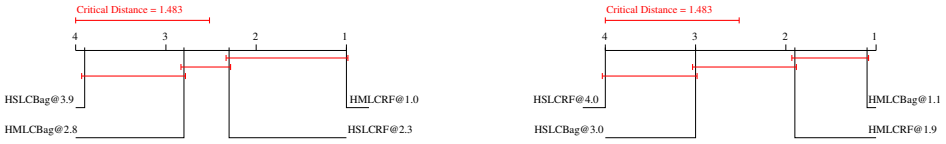
(b) ensemble with 100 trees

Figure 4.11: Average ranks diagrams (with the critical distance at a significance level of 0.05) for hierarchical multi–label classification. The difference in performance for the algorithms connected with a red line is not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 4.9 with the addition of a single predictive clustering tree for hierarchical multi–label classification – *HMCPCT* and predictive clustering trees for hierarchical single–label classification – *HSCPCT*.

(running times in Figure 4.12(a) and model sizes in Figure 4.12(b)). Random forests for HMC are statistically significantly faster than both bagging for HMC and HSC, while random forests for HSC are significantly faster than bagging for HSC. The models of

bagging of HMC are statistically significantly smaller than the models from the ensembles for HSC. The models of random forests for HMC are statistically significantly smaller than the models of the random forests for HSC.

We further investigate the speed up and size of the models ratios. The random forests for HMC are ∼6.4 times faster and have ∼4.6 times smaller models than the random forests for HSC. Similarly, bagging for HMC is ∼6.4 times faster and has ∼3.2 times smaller models than bagging for HSC. Random forests for HMC are ∼7.8 times faster and produce models of comparable size to those of bagging for HMC. All in all, in terms of efficiency, random forests for HMC outperform the rest of the ensemble methods.



(a) *Time efficiency* (b) *Size of the models*

Figure 4.12: Efficiency of the ensembles for hierarchical multi–label classification. The size of the ensembles is 50 trees.

To summarize, the difference in predictive performance between ensembles for HMC and ensembles for HSC is not statistically significant. However, on several datasets, the ensembles for HMC outperform the ensembles for HSC. Moreover, the ensembles for HMC are more efficient than the ensembles for HSC. Finally, the ensembles for HMC lift the predictive performance of a single predictive clustering tree.

4.2.4 Summary of the results

Let us summarize the findings from the empirical evaluation of the proposed methods. To begin with, the results show that ensembles lift the predictive of a single classifier also if the output/target is structured. Next, we construct learning curves for the ensemble methods (for the ensembles predicting both the structured output and the sub-components). The learning curves help to determine the number of base classifiers in an ensemble that offers optimal predictive performance and efficiency of the ensemble. We then compare the performance (predictive power and efficiency) of the ensembles that predict the complete structured output and the ensembles that predict components of the outputs.

We performed the empirical evaluation over a wide range of datasets. In particular, we used 13 datasets with multiple continuous target variables (multi-target regression), 9 datasets with multiple discrete target variables (multi-target classification) and 10 datasets

with hierarchical multi-label classification problems. We summarize the main findings of the experimental evaluation as follows:

- The ensembles for predicting structured outputs (i.e., ensembles of PCTs) lift the predictive performance of a single PCT. The difference in performance is statistically significant at 0.05. Previously this was only shown for the applications where the target is a single continuous or discrete variable. This finding is valid for the three machine learning tasks that we consider in this thesis. This suggests that the non-trivial relations that might exist between the sub-components of the structure are included when combining predictions of several classifiers or when injecting some source of randomness in the learning algorithm.

- The learning curves show that the predictive performance of the ensembles is not increasing significantly after adding the 50-th PCT to the ensemble. This means that constructing an ensemble of 50 trees is a reasonable compromise (for the majority of the domains) between the predictive performance and the efficiency. Furthermore, the learning curves show that on the majority of the domains the ensembles of PCTs have better predictive performance than the ensembles that predict the sub-components. This is especially the case when the ensembles contain fewer PCTs.

- The differences in the predictive performances of ensembles of PCTs and ensembles of trees predicting sub-components of the output are not statistically significant at 0.05 in any of the tasks. However, the ensembles of PCTs often have better predictive performance (i.e., smaller average ranks) than the ensembles of trees predicting the sub-components of the output.

- We assess the efficiency of the proposed methods through the time needed to construct the classifiers and the size of the trees in the ensembles. The ensembles of PCTs are more efficient than ensembles of trees predicting the sub-components of the output on all tasks using both efficiency measures. In particular, random forests of PCTs outperform all other ensembles in terms of time consumption and size of the trees in the ensemble for predicting multiple continuous target variables. Bagging of PCTs has the smallest models when predicting multiple discrete target variables and hierarchical multi–label classification.

5 Further developments

In the previous chapters, we presented an extension for predicting structured outputs of the most widely used ensemble techniques in context of decision trees: bagging and random forests. We considered three typical types of structured outputs: multiple continuous variables, multiple discrete variables and multiple labels that are organized into a hierarchy. All of these support Euclidean distances and hence closed-form prototypes are based on averaging and majority voting.

In this chapter, we further discuss the extensions of the proposed approach for additional types of structured outputs (such as time series). We also discuss additional distances for hierarchical multi-label classification. In both case, closed-form prototype do not exist, which means medoids have to be used as prototypes and the combination of the predictions of individual PCTs in an ensemble need to be modified accordingly.

We next show how the random forests approach can be exploited to obtain feature rankings for structured outputs. We present a case study for biomarker discovery. In this case study, we compare the ranking obtained for structured outputs (in particular, multiple discrete targets) with a ranking obtained for a single target.

The last section of this chapter outlines a novel algorithm for ensemble learning that is based on beam search. The ensemble obtained in this way has two properties: interpretability and controlled diversity. The interpretability of an ensemble is an interesting research topic in the ensemble learning community. Several approaches exist that deal with the problem of obtaining a model that is representative for the whole ensemble (Assche, 2008; Bauer and Kohavi, 1999; Craven, 1996; Domingos, 1998; Ferri *et al.*, 2002; Geurts, 2001; Kargupta *et al.*, 2006; Triviño-Rodriguez *et al.*, 2008). Another interesting research topic is the notion of diversity in the ensembles and its influence/connection to the predictive performance of the ensemble (Bernard *et al.*, 2009; Brown and Kuncheva, 2010; Brown *et al.*, 2005; Carney and Cunningham, 2000; Giacinto and Roli, 2001; Hansen and Salamon, 1990; Kuncheva, 2004; Kuncheva and Whitaker, 2003). All in all, we suggest an approach that unifies the two aforementioned research topics and provide insights how the beam search can be further explored and exploited.

5.1 Predicting other structured outputs

The approaches that we described in Chapter 3 can be easily extended for handling other types of structured outputs. To adequately adjust the algorithms, the only requirement is that a distance can be defined for the given structured output. This means that the variance and prototype functions for induction of PCTs (Chapter 3 and Section 5.3) will now use the new distance measure.

The construction of the ensembles will change in the part of the voting scheme (Section 3.2.6). The new voting scheme will employ the prototype function which uses the new distance. When the distances does not allow for a closed-form prototype, it will return the medoid of the individual predictions. This is different as compared to the voting schemes we used in Chapter 3 which are based on averaging. This is because the distance (for regression and HMLC) is Euclidean and the mean is a closed-form prototype. The feature ranking will additionally require a quality criterion for the prediction of the specific structured outputs. In the following, we will shortly describe several extensions of the proposed algorithms: the use of additional distances for HMLC and predicting time series, as well as calculating prototype and voting for those.

5.1.1 Distances for hierarchical classification

In Chapter 3, we described PCTs for hierarchical multi-label classification. By default, they use weighted Euclidean distance to calculate the variance and the prototype. However, we have also investigated the predictive performance of other distance measures on datasets from functional genomics (Aleksovski *et al.*, 2009).

The distances that can currently be used for hierarchical multi-label classification using PCTs are:

- *weighted Jaccard distance*: The distance between two examples is the ratio between the sum of the weights of their joint annotations and the sum of the weights of all their annotations (Jaccard, 1901; Tan *et al.*, 2005). As in the case of weighted Euclidean distance, the same exponential weighting scheme can be used.

- *SimGIC*: The distance between two examples is the ratio between the sum of the information contents of their joint annotations and the sum of the information contents of all their annotations (Pesquita *et al.*, 2007).

- *ImageCLEF*: The distance takes into account the depth and the difficulty of the predictive problem (the so-called 'branching factor') at which an error has occurred (Tommasi *et al.*, 2010).

The distances were extended for handling hierarchies organized as DAGs similarly as for the weighted Euclidean distance. The variations of the PCT algorithm using the different distances to select tests in the internal nodes were compared over several gene-function prediction datasets. The overall conclusion of the experimental comparison was that there is no statistically significant difference in the performance of the algorithms. The statistical significance was assessed using Friedman test for multiple hypothesis testing (Demšar *et al.*, 2006; Friedman, 1940).

5.1.2 Time series

A time series is a sequence of data points measured at successive time points at uniform or variable time intervals. The selection of a distance/similarity measure for time series depends on the application at hand and the form of the time series (equal/different lengths, sampled at uniform/non-uniform intervals, etc). For an extensive list of the distances for time series see the survey by Liao (2005).

In the CLUS system, four distance measures can be used in the context of predicting time series (Slavkov *et al.*, 2010b): *Euclidean distance*, *Pearson's correlation coefficient*, *Qualitative distance measure* (Todorovski *et al.*, 2002) and Dynamic time warping distance (Sakoe and Chiba, 1978). Depending of the application domain, one can choose which distance measure should be used. The prediction of time series using PCTs has been used in two studies from different domains: molecular biological (clustering time series of gene expression levels) and agriculture (clustering time series of crop and weed cover).

Slavkov *et al.* (2010b) applied the approach to time series data concerning the changes in the expression level of yeast genes in response to a change in environmental conditions. Their evaluation shows that PCTs are able to cluster genes with similar responses, and to predict the time series response of a gene based on the description of the gene. Next, Debeljak *et al.* (2011) use PCTs with the dynamic time warping distance to predict time series of crop and weed cover at agricultural sites throughout the United Kingdom. The time series in this case study are irregular both in terms of length and intervals between points. Both case studies offered interesting and insightful results for the respective domains. This is a unique approach that performs clustering of time series and simultaneously provides descriptions of the clusters.

5.1.3 Prototypes and voting

The ensembles of PCTs we considered in Chapter 3, namely, ensembles of MTRTs and PCTs for HMC, use Euclidean distances for the target datatypes. For these, closed-form

prototypes based on averaging the values of the targets over the examples in a leaf exist and were used. For MTCTs, the majority class for each of the targets is selected as the value of the respective component of the prediction.

The same approach can be and is taken for combining the predictions of individual trees in an ensemble. For continuous targets (and also for HMC), averaging is used. For discrete targets, the class probability distribution vote is used (or one can decide to use majority vote).

For the datatype/distance combinations considered in the above subsections, no closed-form prototypes exist, namely, for hierarchies and time series with distances other than the weighted Euclidean. In this case, to make a prediction in a leaf, we take the medoid over the structured output values in the leaf with respect to the appropriate distance measure.

5.2 Feature ranking for structured outputs

In this section, we describe how the random forest mechanism can be further exploited to calculate the importance of the variables, i.e., to obtain a feature ranking. Breiman (2001a) introduced and described the approach for feature ranking for a single (continuous or discrete) target variable. We extend this approach, so that it can perform feature ranking for arbitrary structured output. To this end, we use predictive clustering trees (see Chapter 3) and adequate error measures for the given structured outputs.

Here, we first present the algorithm itself (Table 5.1). We then describe several error measures that can be used for structured outputs. Finally, we present a case study where we use feature ranking for biomarker discovery.

5.2.1 Feature ranking using random forests

The proposed approach for feature ranking using random forests is presented in Table 5.1. It is based on internal out-of-bag estimates of the error and noising of the descriptive variables. The rationale behind this approach is that if a variable is important for the target, then noising its values should produce an increase in the error. To create each tree from the forest, the algorithm first creates a bootstrap replicate of the training set (line 4, from the $Induce\_RF$ procedure, Table 5.1). The samples that are not selected for the bootstrap replicate are called out-of-bag (OOB) samples (line 7, procedure $Induce\_RF$). These samples are used to evaluate the performance of each tree from the forest.

Suppose that there are D descriptive variables. After each tree from the forest is built, the values of the descriptive attributes for the OOB samples are randomly per-

muted attribute-by-attribute thus obtaining D noised/permuted OOB samples (line 3 from $Update\_Imp$ procedure). The predictive performance of each tree T_i is evaluated on the original OOB data ($Err_{OOB_i} = Evaluate(T_i, E_{OOB_i})$) and the permuted versions of the OOB data ($Err_{ji} = Evaluate(T_i, Randomize(E_{OOB_i}, j))$). Tthe importance of the j-th variable (I_j) is then calculated as the relative increase of the error that is obtained when its values are randomly permuted (Equation 5.1). The importance is averaged over all trees in the forest. The variable importance is calculated using the following equation:

$$I_j = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{Err_{ji} - Err_{OOB_i}}{Err(OOB_i)} \tag{5.1}$$

where k is the number of bootstrap replicates (or size of the random forest) and I_j is the importance of the j-th descriptive variable ($0 < j \leq D$).

Table 5.1: The algorithm for feature ranking via random forests. E is the set of training examples, k is the number of trees in the forest, D is the number of descriptive variables and $f(D)$ is the size of the feature subset that is considered at each node during tree construction.

procedure Induce_RF($E, k, f(D)$)
returns Forest, Importances

1: $F = \emptyset$
2: $I = \emptyset$
3: **for** $i = 1$ **to** k **do**
4: $\quad E_i = \text{Bootstrap\_sample}(E)$
5: $\quad T_i = PCT(E_i, f(D))$
6: $\quad F = F \bigcup T_i$
7: $\quad E_{OOB} = E \setminus E_i$
8: $\quad \text{Update\_Imp}(E_{OOB}, T_i, I)$
9: $I = \text{Average}(I, k)$
10: **return** F, I

procedure Update_Imp(E_{OOB}, T, I)

1: $Err_{OOB} = \text{Evaluate}(T, E_{OOB})$
2: **for** $j = 1$ **to** D **do**
3: $\quad E_j = \text{Randomize}(E_{OOB}, j)$
4: $\quad Err_j = \text{Evaluate}(T, E_j)$
5: $\quad I_j = I_j + \frac{1}{k} \cdot \frac{Err_j - Err_{OOB}}{Err_{OOB}}$
6: **return**

For each type of structured output, the algorithm requires an appropriate error measure. To begin with, we use the average misclassification rate for multi-target classification, where the target structure is a tuple of discrete variables. In the case of predicting a tuple of continuous variables (multi-target regression), we use the average relative root mean squared error ($\overline{RRMSE}$). If the target is a time series, we use the root mean squared error ($RMSE_{TS}$), where the error is calculated according to a distance measure on time series data. In the case of hierarchical multi-label classification, we propose to use ($1 - \overline{AUPRC}$)

as error measure. All in all, based on the application at hand and the type of structured output, one can easily update these measures to be more suitable for performing the task at hand.

The proposed approach for feature ranking generates a single ordered list of features valid for the structured output as a whole. A less desirable alternative is to generate several rankings, one for each component of the structured output (if it is possible to decompose the output at all), and then use some complex aggregation function to produce a single ranking valid for the complete structured output (for example, see (Jong *et al.*, 2004; Saeys *et al.*, 2008; Slavkov *et al.*, 2010a)).

To compare rankings, e.g., two rankings generated with the alternative approaches mentioned above, one can use testing error curves (Slavkov *et al.*, 2010a). Testing error curves are constructed as follows. Using the feature ranking, $|D|$ classifiers (or in general predictive models) are constructed (D is the number of descriptive attributes). The first classifier is constructed using only the top ranked feature; the second classifier is constructed using the two top ranked features and so on. The curve plots on the x-axis the number of features and on the y-axis the misclassification rate (or in general error).

5.2.2 Biomarker discovery using multi-target ranking

We applied the above approach to the problem of biomarker discovery for neuroblastoma, a type of embryonal tumor (Kocev *et al.*, 2008). We used the data from the micro array study performed by Schramm *et al.* (2004) on 63 patients (samples). In this study, the main interest is to find a set of biomarkers for the outcome of the disease (relapse or no event). However, there are additional clinical parameters that are available, in addition to the outcome, such as MYCN gene amplification and 1p chromosome deletion. It is known that these genomic alterations are connected to the disease outcome.

Figure 5.1 depicts two testing error curves, one for the feature ranking when all three variables are used as targets and one for the ranking when the target is only the disease outcome. We can note from the curves that the multi-target ranking is better than the one when only a single variable is used. To begin with, the multi-target classifiers that are constructed using the top most ranked features exhibit better predictive performance than the classifiers for the single target ranking. Note that this is especially important for the domain of biomarker discovery, where the users are interested in the top 10-20 ranked features/genes, so they can perform wet-lab experiments using the results of the ranking. Furthermore, the Wilcoxon test that considers the complete testing error curves shows that the classifiers from the multi-target ranking outperform the classifiers from the single target ranking with $p < 4 \cdot 10^{-5}$. The Wilcoxon test on the testing error curves for the

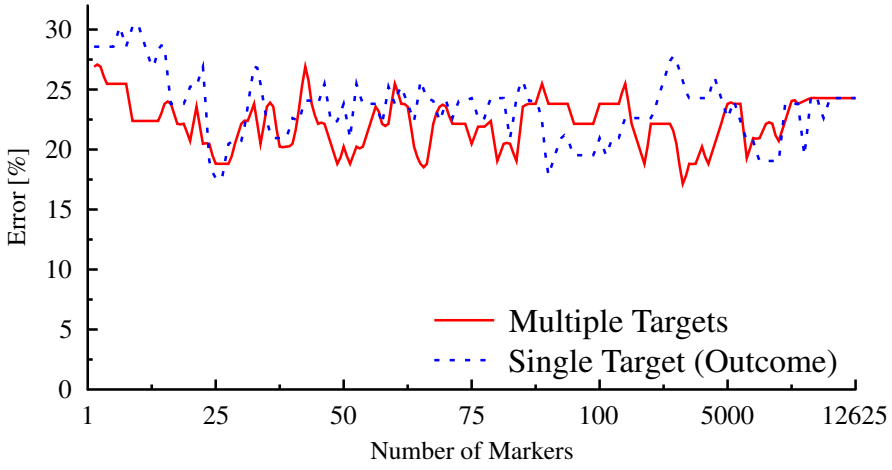


Figure 5.1: Two testing error curves for feature ranking: one for all clinical parameters simultaneously and one for disease outcome. The error rate is measured on the target variable 'disease outcome'.

other two variables, MYCN gene amplification and 1p chromosome deletion, showed that there is no statistically significant difference between the performance of the multi-target ranking and the performance of the single-target ranking. All in all, the proposed approach can exploit the mutual information/dependence of the multiple targets and perform better feature ranking (i.e., provide more reliable set of biomarkers).

The proposed approach has several advantages over the rankings obtained by learning separate rankings for the components of the output. To begin with, it is general in terms of the type of the output: it can handle various types of structured outputs and it can easily be extended to arbitrary types of structured output. It can exploit the underlying dependencies and relations that may exist between the components of the outputs. Furthermore, if another variable/component is added to the structured output, then for learning of separate rankings this will mean learning an additional ranking for the added variable/component. On the other hand, the running time of the proposed approach of overall ranking will increase only slightly. All in all, the proposed approach is efficient, general and can be extended for arbitrary types of structured output.

5.3 Construction of ensembles of PCTs using beam search

Constraint-based data mining (Džeroski *et al.*, 2010) is concerned with developing data mining algorithms that can take into account user specified constraints. These constraints

make the process of data mining more declarative and increase the influence of the user on the data mining results. Constraints for predictive models can involve error/accuracy of the model, its size and its syntactic form.

The PCT approach has been also adapted to handle such constraints (Struyf and Džeroski, 2006). It handles constraints on trees by first building a maximal tree, then pruning it with an adapted version of the dynamic programming algorithm of Garofalakis *et al.* (2003). This, however, may fail to find a tree satisfying a given set of constraints even when one exists, due to the myopia of greedy search.

Here, we propose a new induction algorithm for PCTs (and trees in general) that uses beam search (we call this implementation CLUS-BS) (Kocev *et al.*, 2007a). The CLUS-BS approach has three main advantages over the TDI algorithm. To begin with, it return set of PCTs, instead of a single PCT. This is useful in some domains, where the domain experts require multiple trees/solutions for the problem at hand. Next, many useful constraints can be pushed into the induction algorithm. For instance, size constraints, such as 'return a tree with at most 15 nodes', can be handled during the induction of the tree, i.e., during the refinement of the trees from the beam and not in post-pruning, while the standard approach handles this mostly during post-pruning (Garofalakis *et al.*, 2003). Finally, this approach is less susceptible to myopia than the standard greedy search.

However, the CLUS-BS approach tends to return trees that are similar to each-other[1], both syntactically (similar attributes appear in the internal nodes of the trees) and semantically (the trees make equal predictions for the same instances). To overcome this, we introduce an additional term in the heuristic score that calculates the similarity of the tree to the other trees that are already in the beam. In this way, the induced beam will contain trees that are less similar to each other and the user can control the level of diversity in the beam (we call this implementation CLUS-BS-S).

The trees obtained using beam search (especially using CLUS-BS-S which favours their diversity) can be regarded as an ensemble. Thus, CLUS-BS-S can be used for ensemble learning where each tree from the beam can vote to obtain a joint prediction. Moreover, the best ranked model from the beam can be selected as a representative for the whole ensemble, and, thus, CLUS-BS and CLUS-BS-S can produce 'interpretable' ensembles. Furthermore, using the diversity measure, we can investigate the connection between the diversity of an ensemble of trees and its predictive performance. The latter question has received a significant amount of attention from the ensemble learning community (Brown

[1]Note that this is to be expected having in mind the algorithm presented below in Table 5.2 and the heuristic score from Equation 5.2. If a given tree has good predictive performance, then its refinements will most probably also have good predictive performance.

and Kuncheva, 2010; Brown *et al.*, 2005; Carney and Cunningham, 2000; Hansen and Salamon, 1990; Kuncheva, 2004; Kuncheva and Whitaker, 2003).

In the remainder of this Section, we first describe the beam search induction algorithm. Then, we present the heuristic score that we use to evaluate the trees and we show how the similarity measure can be included in the score. Next, we discuss the results of the experimental evaluation of the proposed approach. At the end, we conclude and give some directions for further work.

5.3.1 Beam search induction of PCTs

We propose a new approach for induction of decision trees that uses a beam search strategy (Kocev *et al.*, 2007a). The algorithm is outlined in Table 5.2. The beam is a set of trees (PCTs) that are ordered by their heuristic value, which is related to their accuracy/error and size. The algorithm starts with a beam that contains precisely one PCT: a leaf covering all the training data E.

Each iteration of the main loop creates a new beam by refining the PCTs in the current beam. That is, the algorithm iterates over the trees in the current beam and computes for each PCT its set of refinements (Fig. 5.2). A refinement is a copy of the given PCT in which one particular leaf is replaced by a stub, which is depth one sub-tree (i.e., an internal node with an attribute-value test and two leaves).

Note that a PCT can have many refinements: a PCT with L leaves yields $L \cdot M$ refined trees, with M the number of possible tests that can be put in a new node. In CLUS-BS, M is equal to the number of attributes: CLUS-BS considers for each attribute only the test with the best heuristic value. Note that the number of possible tests on a numeric attribute A is typically huge: one test $A < a_i$, for each possible split point a_i. CLUS-BS only constructs one refined tree for the split that yields the best heuristic value. This approach limits the number of refinements of a given PCT and increases the diversity of the trees in the beam.

For each generated refinement, CLUS-BS computes its heuristic value. The heuristic function differs from the heuristic used in the TDI algorithm from Section 3.1. The heuristic in the latter is local, i.e., it only depends on the instances local to the node that is being constructed. In CLUS-BS, the heuristic is global and measures the quality of the entire tree. The reason is that beam search needs to compare trees of arbitrary structure, whereas TDI only needs to compare trees that differ in one node, i.e., to rank different tests for the same tree node.

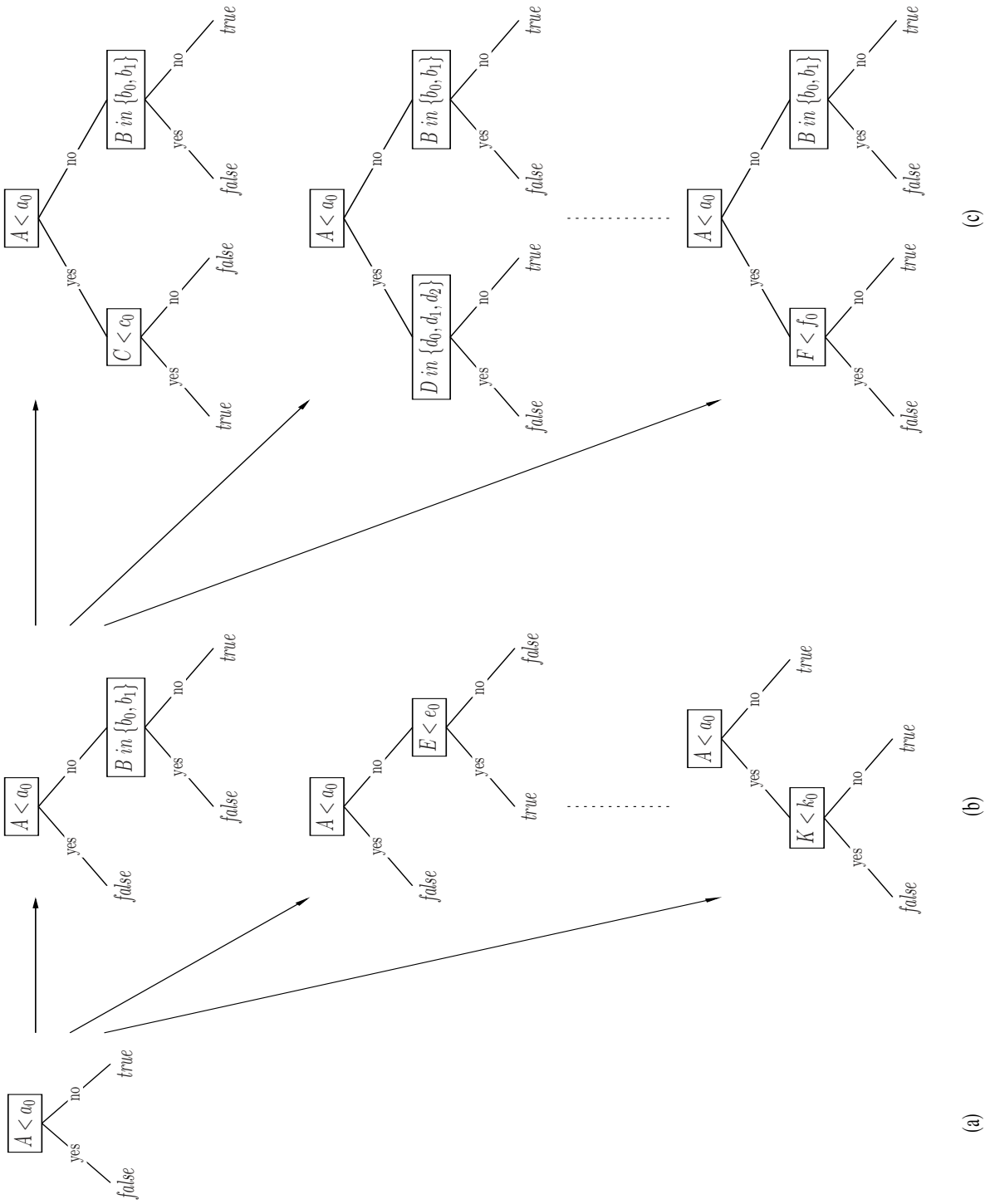


Figure 5.2: Refining the trees in the beam. (a) A tree in the beam; (b) The refinements of tree (a); (c) The refinements of the top-most tree in (b). Note that the refinements (c) are only computed in a subsequent iteration of the search after the top-most tree of (b) has entered the beam.

Table 5.2: The CLUS-BS beam search algorithm for induction of predictive clustering trees.

procedure CLUS-BS(E,k)

1: $i = 0$
2: T_{leaf} =leaf(centroid(I))
3: $h = \text{Heuristic}(T_{\text{leaf}}, E)$
4: $\text{beam}_0 = \{(h, T_{\text{leaf}})\}$
5: **repeat**
6: $\quad i = i + 1$
7: $\quad \text{beam}_i = \text{beam}_{i-1}$
8: $\quad$ **for each** $T \in \text{beam}_{i-1}$ **do**
9: $\quad\quad R = \text{Refine}(T, E)$
10: $\quad\quad$ **for each** $T_{\text{cand}} \in R$ **do**
11: $\quad\quad\quad h = \text{Heuristic}(T_{\text{cand}}, E)$
12: $\quad\quad\quad h_{\text{worst}} = \max_{T \in \text{beam}_i} \text{Heuristic(T, E)}$
13: $\quad\quad\quad T_{\text{worst}} = \text{argmax}_{T \in \text{beam}_i} \text{Heuristic(T, E)}$
14: $\quad\quad\quad$ **if** $h < h_{\text{worst}}$ **or** $|\text{beam}_i| < k$ **then**
15: $\quad\quad\quad\quad \text{beam}_i = \text{beam}_i \cup \{(h, T_{\text{cand}})\}$
16: $\quad\quad\quad$ **if** $|\text{beam}_i| > k$ **then**
17: $\quad\quad\quad\quad \text{beam}_i = \text{beam}_i \setminus \{(h_{\text{worst}}, T_{\text{worst}})\}$
18: **until** $\text{beam}_i = \text{beam}_{i-1}$
19: **return** beam_i

procedure Refine(T, E)

1: $R = \emptyset$
2: **for each** leaf $l \in T$ **do**
3: $\quad E_l = \text{Instances}(E, l)$
4: $\quad$ **for each** attribute a **do**
5: $\quad\quad t =$ best test on a
6: $\quad\quad \{E_1, E_2\} = \text{Partition}(t, E_l)$
7: $\quad\quad l_1 = \text{leaf}(\text{centroid}(E_1))$
8: $\quad\quad l_2 = \text{leaf}(\text{centroid}(E_2))$
9: $\quad\quad n = \text{node}(t, \{l_1, l_2\})$
10: $\quad\quad T_r =$ replace l by n in T
11: $\quad\quad R = R \cup \{T_r\}$
12: **return** R

The heuristic that we propose to use is:

$$h(T, E) = \left(\sum_{\text{leaf} \in T} \frac{|E_{\text{leaf}}|}{|E|} Var(I_{\text{leaf}}) \right) + \alpha \cdot \text{size}(T) \,, \tag{5.2}$$

with E all training data and E_{leaf} the examples sorted into a specific leaf. It has two components: the first one is the average variance of the leaves of the PCT weighted by size, and the second one is a size penalty. The latter biases the search to smaller trees and can be seen as a soft version of a size constraint. The size function that we use throughout the paper counts the total number of nodes in the PCT (both the internal nodes and the leaves).

After the heuristic value of a tree is computed, CLUS-BS compares it to the value of the worst tree in the beam. If the new tree is better, or if there are fewer than k trees in the beam (k is the beam width), then CLUS-BS adds the new PCT to the beam: if this

causes the beam width to exceed the prescribed beam size, CLUS-BS removes the worst tree from the beam. The algorithm ends when the beam no longer changes. This occurs either if none of the refinements of a tree in the beam is better than the current worst tree, or if none of the trees in the beam yields any valid refinements. This is the point in the algorithm where the user constraints can be used to prune the search: a refinement is valid in CLUS-BS if it does not violate any of these constraints.

Note that Equation 5.2 is identical to the heuristic used in the TDI algorithm from Section 3.1 if we assume that there are no constraints, $\alpha = 0$ and $k = 1$. In this case, the tree computed by CLUS-BS will be identical to the tree constructed with TDI. The only difference with TDI is the order in which the leaves are refined: TDI refines depth-first, whereas CLUS-BS with a beam width of one refines best-first.

Preliminary experiments have indicated a possible disadvantage of the proposed approach for induction of PCTs. Namely, the beam tends to fill up with small variations of the same PCT, i.e., trees that differ only in a single node. To alleviate this, we modify the heuristic score (Equation 5.2) to include also a similarity term. We discuss this similarity term in the next section.

5.3.2 Diversity in the beam

The diversity of the predictive models in ensembles is of recognized importance in the area of ensemble learning. Unfortunately, there is no 'uniquely agreed definition' (Brown and Kuncheva, 2010) of diversity. Many different diversity measures have been proposed (Kuncheva and Whitaker, 2003) with one single goal: to increase the predictive performance of the ensembles by balancing the accuracy of the base classifiers with their diversity. Several studies have been performed concerning the clarification and quantification of the role of the diversity in ensemble learning (Brown and Kuncheva, 2010; Brown *et al.*, 2005; Carney and Cunningham, 2000; Kuncheva, 2004; Kuncheva and Whitaker, 2003).

However, there is no unifying theory behind the different diversity measures or recommendations for which measure to use under what circumstances. Here, we propose to use Euclidean measures between the tree predictions for all of the machine learning tasks. This approach is applicable in a straightforward manner for the regression tasks. For the classification tasks, we propose to use the average distance between the probability distributions of the classes predicted for each example.

We propose to calculate the distance between the prediction of the trees as follows:

$$d(T_1, T_2, E) = \frac{1}{\eta} \cdot \sqrt{\frac{\sum_{t \in E} d_p(p(T_1, t), p(T_2, t))^2}{|I|}}\,, \tag{5.3}$$

with η a normalization factor, $|E|$ the number of training instances, $p(T_j, t)$ the prediction of tree T_j for instance t, and d_p a distance function between predictions. In Equation 5.3, η and d_p depend on the learning task. For regression tasks, d_p is the absolute difference between the predictions, and $\eta = M - m$, with $M = \max_{t \in E, j \in \{1,2\}} p(T_j, t)$ and $m = \min_{t \in E, j \in \{1,2\}} p(T_j, t)$. This choice of η ensures that $d(T_1, T_2, E)$ is in the interval $(0, 1)$. For classification tasks, the distance is calculated similarly as for the regression with d_p is now the sum of absolute differences between the probabilities for each class predicted for the instance by the two trees.

In addition, we also consider disagreement measure for classification. Here, the η parameter is set to 1 and $d_p = \delta$ with

$$\delta(a, b) = \begin{cases} 1 \text{ if } a \neq b \\ 0 \text{ if } a = b \end{cases} \tag{5.4}$$

The proposed distance measure between the predictions of the PCTs can be easily extended for predicting structured outputs. For predicting multiple targets, both discrete and continuous, the average distance per target variable can be used. In the context of hierarchical multi-label classification, a similar (weighted) average can be calculated for each of the nodes in the hierarchy. Some other distances for hierarchies of labels can also be used (Aleksovski *et al.*, 2009).

Using these definitions of distances between trees, the heuristic score for the trees (updated version of Equation 5.2) can be calculated as follows:

$$h_s(T, \text{beam}, E) = \left(\sum_{\text{leaf} \in T} \frac{|E_{\text{leaf}}|}{|E|} Var(E_{\text{leaf}}) \right) + \alpha \cdot \text{size}(T) + \beta \cdot \text{sim}(T, \text{beam}, E) \tag{5.5}$$

where the first two terms are the same as in the Equation 5.2, β is a user defined parameter that controls the influence of the beam diversity on the total heuristic score and $\text{sim}(T, \text{beam}, E)$ is the similarity score. Note that the heuristic score of each tree that is already in the beam is updated when new tree candidate (T_{cand}) is produced. The updating doesn't require extra processing time, since in any case we need to calculate the distance of each tree from the beam to the candidate tree. The similarity score of each tree T (including the candidate tree T_{cand}) is calculated as:

$$\text{sim}(T, \text{beam}, E) = 1 - \frac{d(T, T_{cand}, E) + \sum_{T_i \in \text{beam}} d(T, T_i, E)}{|\text{beam}|} \tag{5.6}$$

where T is a tree in the beam or candidate tree (T_{cand}), E is the training set and $d(T_i, T_j, E)$ is the distance as defined in Equation 5.3.

Since the heuristic value of a tree now also depends on the other trees in the beam, it changes when a new tree is added. Therefore, each time that CLUS-BS-S considers a new candidate tree, it recomputes the heuristic value of all trees already in the beam using Equation 5.5. The heuristic score for the trees already in the beam is updated only with the term for the similarity, while the term for the predictive performance remains the same. To make the calculations more efficient, one can exploit some properties of the distance measures, such as symmetry $d(T_a, T_b, E) = d(T_b, T_a, E)$ and reflexiveness $d(T_a, T_a, E) = 0$.

5.3.3 Empirical evaluation

We experimentally evaluated the proposed approaches (CLUS-BS and CLUS-BS-S) using 16 datasets (8 classification and 8 regression) from the UCI repository (Asuncion and Newman, 2007). We used the disagreement measure for the classification datasets and the absolute difference between the predictions for the regression datasets (as described above). We set the beam size k to 10, the soft-size constraint influence α to 0.00001 and the influence of the diversity β to 1. The performance of the algorithms was compared over a range of hard size constraints varying from 5 to 51 and no size constraints. The performance of the algorithms was assessed by 10-fold cross-validation. A more detailed description of the experiments, results and discussion can be found in (Kocev *et al.*, 2007a).

The results show that CLUS-BS yields models of comparable accuracy to a standard TDI algorithm. CLUS-BS wins[1] on 5 classification and 3 regression tasks. TDI wins on 2 classification and no regression tasks. This confirms that CLUS-BS yields more accurate models, which can be explained by the fact that it is less susceptible to myopia. There is no clear correlation between the number of wins and the value of the size constraint.

CLUS-BS-S wins over TDI on 6 classification and 4 regression tasks and loses on 13 classification and 1 regression tasks. CLUS-BS-S performs, when compared to CLUS-BS, worse on classification data than on regression data. This is because the heuristic (used in CLUS-BS-S) trades off accuracy for diversity. If a given tree in the beam is accurate, then new trees will be biased to be less accurate because the similarity score favors trees with different predictions. For classification problems, this effect is more pronounced because a '0/1' distance between predictions is used, whereas in the regression case a continuous distance function is used. The latter makes it 'easier' to have different predictions that are still reasonably accurate. Also, this effect is stronger for larger size constraints (the majority of the losses of CLUS-BS-S are for SC31, SC51 and NoSC), because the relative

[1] The statistical significance of the results was assessed using the paired t-test. A win was considered statistically significant if the corresponding p value was smaller than 0.05.

contribution of the similarity score to the heuristic is greater for larger size constraints. The losses are in the range of 1-2% accuracy, so for the majority of domains this is not a serious problem.

The results regarding the diversity in the beam show that CLUS-BS-S trades off accuracy for beam diversity. The beam diversity for CLUS-BS-S is always larger than that of CLUS-BS. Moreover, the variance of the accuracies of the trees in the beam increases with the beam diversity. Additionally, the trees produced by CLUS-BS-S not only produce different predictions, but are also syntactically different from the trees constructed with CLUS-BS.

We plan to further extend this work along several dimensions. To begin with, we will consider introduction of the diversity during the test selection in the tree building process, i.e., during the generation of the refinements. This can be done in a computationally efficient way if the distance measures are Euclidean. Second, we will investigate the influence of the beam size on the performance. Next, we will perform experiments for different values of the β parameter to gain more insight into the trade-off between the predictive performance and beam similarity. Finally, we will combine the trees in the beam in an ensemble and comment on the influence of the diversity of the trees in the ensemble on the performance of the ensemble. Moreover, the ensemble that is obtained in this way can be interpreted by selecting the top ranked tree (since in the beam the trees are ordered by their performance). All in all, the proposed approach will offer further understanding about the influence of the diversity in the ensemble to its accuracy and ensemble interpretability.

6 Case studies

In this chapter, we present three case studies that use ensembles for predicting structured outputs. The case studies are from three domains: ecological modelling (modelling vegetation condition), image annotation (annotation of medical X-ray images) and functional genomics (predicting the functions of a gene). In these case studies, two machine learning tasks are addressed: predicting multiple continuous variables (vegetation condition) and hierarchical multi–label classification (image annotation and functional genomics).

In addition to these case studies, we have used ensembles for predicting structured outputs to construct habitat models for the diatoms in lake Prespa, Macedonia (Kocev *et al.*, 2010). The habitat for the diatoms was described using several environmental variables, and the communities were described by the abundance of diatom species at the given sites. The predictive performance of the obtained habitat models (PCTs for predicting multiple continuous variables) was not high: We used ensembles to test whether the performance of the PCTs can be significantly lifted. Although the ensembles do lift the predictive performance of the PCTs in this setting, the conclusion was that the predictive performance is limited by the size of the dataset and the selection of the descriptive (environmental) variables and not by the learning paradigm (in our case PCTs).

The case studies presented here demonstrate the wide range of possible applications of the proposed algorithms and extensions. We show that the ensembles for predicting structured outputs have competitive predictive performance (and even better in come cases) as compared to the state-of-the-art approaches used in the respective application domains. In addition, the ensembles for predicting structured outputs are more efficient, having smaller running times and producing smaller models.

In the next sections, we present the three applications as follows. First, in Section 6.1, we describe the use of PCTs and ensembles of PCTs for prediction of the vegetation condition in the state of Victoria, Australia, from GIS and remote-sensed data. Next, in Section 6.2, we present the application of PCT ensembles to the annotation of medical X-ray images. Finally, in Section 6.3, we compare ensembles (in particular bagging) of PCTs for predicting the functions of a gene to state-of-the-art approaches to predicting gene function used in functional genomics.

6.1 Predicting vegetation condition

In this section, we present a study concerned with modelling the condition of remnant indigenous vegetation. To this end, we use ensembles for predicting structured outputs (in particular, predicting multiple continuous variables). The condition of the vegetation is described by multiple (habitat hectares) scores that reflect the structural and compositional attributes of a wide variety of plant communities at a given site. Multiple sites were manually assessed, in terms of these scores, and subsequently described with GIS and remote-sensed data.

From the data, we learned a (pruned) PCT and ensembles of PCTs. We compare their performance with that of linear regression, regression trees (that predict individual numeric variables) and ensembles of regression trees. The pruned PCT was constructed to extract knowledge from the data. The goal was to better understand the resilience of some indigenous vegetation types and the relative importance of biophysical and landscape attributes that influence their condition.

From the learned models, we can conclude that the most important variables influencing all scores are those related to tree cover. This holds also for scores that do not depend directly on the presence of tree cover. Land cover is also of high importance, with dense forest cover yielding high scores. Finally, climate (including the variability of weather conditions) also plays an important role.

The ensembles of PCTs were used to generate maps of the condition of the indigenous vegetation: They were selected because of their high predictive power and efficiency. We compared their performance with the performance of the ensembles of regression trees. In terms of predictive performance, the difference between the two methods was not statistically significant at the confidence level 0.05. However, if we also consider the efficiency (time needed to construct the classifier and size of the underlying models), the random forests of PCTs should be preferred.

The usefulness of models of vegetation condition is twofold. First, they provide an enhanced knowledge and understanding of the condition of different indigenous vegetation types, and identify possible biophysical and landscape attributes that may contribute to vegetation decline. Second, these models may be used to map the condition of indigenous vegetation across extensive areas (in this case study, we generated a map for the whole area of Victoria state, Australia) with some predictive confidence using easily obtained remotely acquired data together with adequate field data, these maps can be used in support of biodiversity planning, management and investment decisions.

6.2 Hierarchical annotation of medical images

Hierarchical multi-label classification (HMC) problems are encountered increasingly often in image annotation. However, flat classification machine learning approaches are predominantly applied in this area, in particular collections of SVMs. In this case study, we propose to exploit the annotation hierarchy in image annotation by using ensembles of PCTs for HMC.

We apply the ensembles of PCTs for HMC to two benchmark tasks for hierarchical annotation of medical (X-ray) images and an additional task for photo annotation. We compare it to a collection of SVMs (trained with a χ^2 kernel), the best-performing and most-frequently used approach to (hierarchical) image annotation. Our approach achieves better results than the competition on all of these: For the two medical image datasets, these are the best results reported in the literature so far[1]. Our approach has superior performance, both in terms of accuracy/error and especially in terms of efficiency.

We explore the relative performance of ensembles of PCTs for HMC and collections of SVMs under a variety of conditions. Along one dimension, we consider three different datasets. Along another dimension, we consider two ensemble approaches, bagging and random forests. Furthermore, we consider several state-of-the-art feature extraction approaches and combinations thereof. Finally, we consider two types of feature fusion, i.e., low- and high-level fusion.

Ensembles of PCTs for HMC perform consistently better than SVMs over the whole range of conditions explored above. The two ensemble approaches perform better than SVM collections on all three tasks, with random forests being more efficient than bagging (and the most efficient overall). The relative performance holds for different image descriptors and their combinations. The relative performance also holds for both low-level and high-level fusion of the image descriptors, the former yielding slightly better performance. We can thus conclude that for the task of hierarchical image annotation, ensembles of PCTs for HMC are a superior alternative to using collections of SVMs.

At the end, we emphasize the scalability of our approach. Decision trees are one of the most efficient machine learning approaches and can handle large numbers of examples. The ensemble approach of random forests scales very well for large numbers of features. Finally, trees for HMC scale very well as the complexity of the annotation hierarchy increases, being able to handle very large hierarchies organized as trees or directed acyclic graphs. Combining these, our approach is scalable along all three dimensions.

[1]Annotation results for these images can be found at the ImageCLEF competition web site (`http://www.imageclef.org/2009/medanno`) for the *Medical Image Annotation Task* or in the edited volume describing the competitors ((Tommasi *et al.*, 2010) and the references thereof).

6.3 Predicting gene function

The completion of several genome projects in the past decade has generated the full genome sequence of many organisms. Identifying open reading frames (ORFs) in the sequences and assigning biological functions to them has now become a key challenge in modern biology. This last step is often guided by automatic discovery processes which interact with the laboratory experiments.

This case study considers three model organisms: *Saccharomyces cerevisiae* (yeast), *Arabidopsis thaliana* (cress) and *Mus musculus* (mouse) which are well studied organisms in biology. It is still a challenge, however, to develop methods that assign biological functions to the ORFs in these genomes automatically. Different machine learning methods have been proposed to this end, but it remains unclear which method is to be preferred in terms of predictive performance, efficiency and usability.

Here, we present the use of predictive clustering trees for HMC in functional genomics, i.e., to predict gene functions for each of the three organisms. The learner produces a single tree that predicts, for a given gene, its biological functions from a function classification scheme, such as FunCat or the Gene Ontology. Preliminary studies in using PCTs for HMC to predict gene function were conducted by Struyf *et al.* (2005) and Blockeel *et al.* (2006), but were of limited scope: smaller number of datasets, organisms and classification schemes for gene functions were used.

The study also presents a tree-based ensemble learner for HMC. While tree-based ensembles for multi-target prediction were published earlier (Kocev *et al.*, 2007b), this is the first publication describing ensembles of trees for HMC and their implementation CLUS-ENS-HMC. The empirical evidence shows that this learner outperforms several state-of-the-art methods on the datasets from the three model organisms.

This case study reveals several advantages of using the proposed approach over other approaches for prediction of gene functions. To begin with, we show that PCTs for HMC outperforms an existing decision tree learner (C4.5H/M, (Clare, 2003)) in terms of predictive performance. Next, we show that the predictive performance boost, obtained in regular classification tasks by using ensembles, carries over to the HMC context. Then, by constructing an ensemble of PCTs, our method outperforms a statistical learner based on SVMs for *Saccharomyces cerevisiae*, both in predictive performance and in efficiency. Finally, this ensemble learner is competitive to statistical and network based methods for *Mus musculus* data. To summarize, individual PCTs for HMC can give additional biological insight in the predictions, while ensembles of PCTs for HMC yields state-of-the-art quality (predictive performance) for gene function prediction.

6.4 Summary of the case studies

We applied the developed ensembles of PCTs to three application domains. In the case studies, the ensembles of PCTs were compared to the state-of-the-art approaches used in the respective domains. We summarize the conclusions from the case studies as follows:

- *Prediction of vegetation condition*: We used two scenarios for assessing the condition of the indigenous vegetation using easily obtained remote sensed data. The first scenario was concerned with knowledge extraction: we constructed a pruned PCT for predicting multiple continuous targets. The PCT helped to better understand the resilience of some indigenous vegetation types and the relative importance of the biophysical and landscape attributes that influence their condition. For the second scenario, in which high predictive power was required, we constructed ensembles (especially random forests) of PCTs to generate maps of the condition of the indigenous vegetation across the Victoria state, Australia. These maps can support biodiversity planning, management and investment decisions.

- *Hierarchical annotation of medical images*: We applied the ensembles of PCTs for HMC on two benchmark tasks for hierarchical annotation of medical (X-Ray) images and an additional task for general photo annotation. The ensembles of PCTs outperformed, on all three tasks, a collection of SVMs with χ^2 kernel (the best-performing and most-frequently used approach in image annotation). Moreover, for the medical images, the ensembles of PCTs produced the best results reported in the literature. Ensembles of PCTs (especially random forests) are also more efficient than the collection of SVMs.

- Prediction of gene functions: We used ensembles of PCTs for prediction of the gene function in three organisms: *Saccharomyces cerevisiae*, *Arabidopsis thaliana* and *Mus musculus*. The genes were annotated with functions from the FunCat catalogue of functions (tree-shaped hierarchy) and the Gene ontology (DAG shaped hierarchy). The extensive experimental evaluation showed that bagging of PCTs outperforms a statistical learner based on SVMs for the *Saccharomyces cerevisiae* genes, both in terms of predictive performance and efficiency. For the two other organisms bagging of PCTs is competitive to the state-of-the-art approaches in the area of functional genomics.

7 Conclusions and further work

In this thesis, we have developed and evaluate methods for learning ensembles for predicting structured outputs. Each of the proposed methods constructs a single model to make a prediction for the whole structure simultaneously. The proposed methods are general with respect to the type of the output: they can handle multiple target variables and hierarchically structured classes (tree-shaped and DAGs). They are also scalable to a wide range of datasets with different numbers of examples and descriptive variables and different types and sizes of structured outputs.

In the remainder of this chapter, we first summarize the results of the empirical evaluation of the proposed method and the case studies. Then, we discuss how the proposed methods can be further improved and applied.

7.1 Conclusions

The methods we propose in this thesis further extend the predictive clustering framework in the context of ensemble learning. They contribute in the areas of ensemble learning, predicting structured outputs and the respective application domains of the case studies: vegetation condition assessment, image annotation and functional genomics.

First, we have developed methods for learning ensembles for predicting structured outputs. The methods are extending the predictive clustering framework in the context of ensemble learning.

The random forests of PCTs, as a side-product, can provide also a feature ranking. In this thesis, we suggested that this can be used to obtain feature ranking for arbitrary structured outputs. The feature ranking obtained this way exploit some underlying connections and relations that exist between the sub-components of the outputs. We show this on a small case study for bio-marker discovery where the proposed approach offers better feature ranking than the feature ranking for the sub-components.

We also proposed a novel ensemble learning algorithm that is based on the beam–search strategy. This algorithm tackles two issues that are actively researched by the community: ensemble diversity and ensemble interpretability. With the proposed algorithm,

we can explicitly control the diversity of the trees that are in the ensemble. Thus, we can investigate the influence of the diversity of an ensemble on its predictive performance. Furthermore, the beam-search keeps the trees sorted by a heuristic score. The best tree from the heuristic score can be thus used as a representative for the ensemble. The ensemble constructed using the proposed approach will be diverse and interpretable.

We applied the developed ensembles of PCTs to three application domains. In the case studies, the ensembles of PCTs were compared to the state-of-the-art approaches used in the respective domains. We summarize the conclusions from the case studies as follows:

- Predicting the vegetation condition: The obtained PCT models contributed for better understanding of the resilience of some indigenous vegetation types and the relative importance of the biophysical and landscape attributes that influence their condition. In addition, the ensembles of PCTs were used to generate maps of the condition of the indigenous vegetation across the Victoria state, Australia, for support the biodiversity planning, management and investment decisions.

- Hierarchical classification of medical images: The ensembles of PCTs for HMC outperfromed a collection of SVMs (the most-frequently used classifier in image annotation). The annotation results produced by the PCT ensembles are the best results reported in the literature for the used medical X-ray images database. Ensembles of PCTs (especially random forests) are also more efficient than the collection of SVMs.

- Prediction of gene functions: We applied bagging of PCTs for prediction of the gene function in three organisms: *Saccharomyces cerevisiae*, *Arabidopsis thaliana* and *Mus musculus*. The genes were annotated with functions from the FunCat catalogue of functions (tree-shaped hierarchy) and the Gene ontology (DAG shaped hierarchy). The extensive experimental evaluation showed that bagging of PCTs is competitive to the state-of-the-art approaches in the area of functional genomics on all three organisms.

7.2 Further work

In this thesis, we presented several methods for learning ensembles that can be used for prediction of three types of structured outputs: multiple continuous variables, multiple discrete variables and hierarchical multi–label classification. One line of further work is to extend the proposed approach for other types of structured outputs (e.g., the ones

we discuss in Section 5.1). Also, other distance measures for structured types can be implemented, thus making the algorithms more flexible and applicable to new domains.

Another line of further work is to evaluate the feature ranking approach for structured outputs (discussed in Section 5.1) on a larger scale. The small case study presented here (dealing with multi-target classification) showed that this approach is interesting: It needs further investigation in scenarios where the output is multiple continuous variables or classes organized in a hierarchy or time series.

A third line of further work is to investigate the beam search tree induction in the context of learning a diverse and interpretable ensemble. This ensemble learning method should be first evaluated in a large study. Then, it can be extended for predicting structured outputs.

Finally, the proposed approach can find many further uses in the application domains already considered here and broader. In image annotation, it can be used for visual codebook construction and large scale image retrieval, as described below. In assessing the state of the environment from remote sensing, other applications are also possible, such as simultaneous prediction of several forest properties (forest height and density) (Stojanova *et al.*, 2010). Many further applications are possible in relating environmental parameters to community structure: Besides considering other ecosystems than lakes, community structure can be viewed as a sub-hierarchy of the taxonomy of living organisms, and the corresponding learning problems as a problem of HMC.

For the construction of a visual codebook, in the area of image annotation, typically k-means clustering is used. Marée *et al.* (2007); Moosmann *et al.* (2008) proposed to use decision trees for predicting a single target variable to this aim, since the decision trees are much faster and more efficient than k-means clustering. Their approach, in addition to better efficiency, offers also better predictive performance. Predictive clustering trees (and ensembles thereof) can be used for visual codebook construction since they can exploit the dependencies between the multiple image classes and thus offer even more discriminative codebooks.

Marée *et al.* (2009) suggested to further exploit decision trees in the context of image retrieval. Typically, in image retrieval, the hierarchical search structure is constructed using approximate or hierarchical k-means algorithm (Philbin *et al.*, 2007). However, predictive clustering trees can be also used to represent such hierarchical search structures. The suggested approach will offer faster image retrieval because the construction of a predictive clustering tree is much faster than k-means clustering.

8 References

ADIAC (2008). Automatic diatom identification and classification. http://rbg-web2.rbge.org.uk/ADIAC/.

Aleksovski, D., Kocev, D., and Džeroski, S. (2009). Evaluation of distance measures for hierarchical multi-label classification in functional genomics. In *ECML/PKDD 2009 Workshop on Learning from Multi-Label Data*, pages 5–16.

Ali, K. and Pazzani, M. (1996). Error reduction through learning multiple descriptions. *Machine Learning*, **24**(3), 173–202.

Allwein, E. L., Schapire, R. E., and Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, **1**, 113–141.

Ando, R. K., Zhang, T., and Bartlett, P. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, **6**, 1817–1853.

Argyriou, A., Evgeniou, T., and Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, **73**, 243–272.

Assche, A. V. (2008). *Improving the applicability of ensemble methods in data mining*. Ph.D. thesis, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium.

Asuncion, A. and Newman, D. (2007). UCI - machine learning repository. http://www.ics.uci.edu/ mlearn/MLRepository.html.

Bakır, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., and Vishwanathan, S. V. N. (2007). *Predicting structured data*. Neural Information Processing. The MIT Press.

Bakker, B. and Heskes, T. (2003). Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, **4**, 83–99.

Banfield, R. E., Hall, L. O., Bowyer, K. W., and Kegelmeyer, W. P. (2007). A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(1), 173–180.

Barutcuoglu, Z., Schapire, R. E., and Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, **22**(7), 830–836.

Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, **36**(1), 105–139.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, **12**, 149–198.

Ben-David, S. and Borbely, R. S. (2008). A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine Learning*, **73**(3), 273–287.

Bernard, S., Heutte, L., and Adam, S. (2009). On the selection of decision trees in random forests. In *IJCNN'09: Proceedings of the 2009 international joint conference on Neural Networks*, pages 790–795. IEEE Press.

Berthold, M. R. and Hand, D. J., editors (2003). *Intelligent Data Analysis: An Introduction*. Springer Verlag.

Bishop, C. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

Blockeeel, H., Džeroski, S., and Grbović, J. (1999). Simultaneous prediction of multiple chemical parameters of river water quality with tilde. In *Proceedings of the 3rd European Conference on PKDD - LNAI 1704*, pages 32–40. Springer.

Blockeel, H. (1998). *Top-down induction of first order logical decision trees*. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium.

Blockeel, H. and Struyf, J. (2002). Efficient algorithms for decision tree cross-validation. *Journal of Machine Learning Research*, **3**, 621–650.

Blockeel, H., Raedt, L. D., and Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann.

Blockeel, H., Bruynooghe, M., Džeroski, S., Ramon, J., and Struyf, J. (2002). Hierarchical multi–classification. In *KDD-2002 Workshop Notes: MRDM 2002, Workshop on Multi-Relational Data Mining*, pages 21–35.

Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S., and Clare, A. (2006). Decision trees for hierarchical multilabel classification: A case study in functional genomics. In *Knowledge Discovery in Databases: PKDD 2006 - LNCS 4213*, pages 18–29. Springer Berlin / Heidelberg.

Boutell, M., Luo, J., Shen, X., and Brown, C. (2004). Learning multi-label scene classification. *Pattern Recognition*, **37**(9), 1757–1771.

Bratko, I. (2000). *Prolog Programming for Artificial Intelligence*. Addison Wesley, 3rd edition.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, **24**(2), 123–140.

Breiman, L. (1996b). Bias, variance and arcing classifiers. Technical Report TR 460, Statistics department, University of Berkeley, CA.

Breiman, L. (2001a). Random forests. *Machine Learning*, **45**(1), 5–32.

Breiman, L. (2001b). Using iterated bagging to debias regressions. *Machine Learning*, **45**(3), 261–277.

Breiman, L. and Friedman, J. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **59**(1), 3–54.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. J. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC.

Brown, G. and Kuncheva, L. (2010). GOOD and BAD diversity in majority vote ensembles. In *Proc. Multiple Classifier Systems (MCS'10) – LNCS 5997*, pages 124–133. Springer–Verlag.

Brown, G., Wyatt, J., Harris, R., and Yao, X. (2005). Diversity creation methods: A survey and categorisation. *Journal of Information Fusion*, **6**(1), 5–20.

Brown, P. J. and Zidek, J. V. (1980). Adaptive multivariate ridge regression. *The Annals of Statistics*, **8**(1), 64–74.

Cai, F. and Cherkassky, V. (2009). Svm+ regression and multi-task learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 418–424.

Caponnetto, A., Micchelli, C. A., Pontil, M., and Ying, Y. (2008). Universal multi–task kernels. *Journal of Machine Learning Research*, **9**, 1615–1646.

Carney, J. G. and Cunningham, P. (2000). Tuning diversity in bagged ensembles. *International Journal of Neural Systems*, **10**(4), 267–279.

Caruana, R. (1997). Multitask learning. *Machine Learning*, **28**, 41–75.

Ceci, M. and Malerba, D. (2007). Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, **28**, 37–78.

Chen, Y. and Xu, D. (2004). Global protein function annotation through mining genome-scale data in yeast saccharomyces cerevisiae. *Nucleic Acids Research*, **32**(21), 6414–6424.

Clare, A. (2003). *Machine learning and data mining for yeast functional genomics*. Ph.D. thesis, University of Wales Aberystwyth, Aberystwyth, Wales, UK.

Clare, A. and King, R. D. (2003). Predicting gene function in Saccharomyces cerevisiae. *Bioinformatics*, **19**(S2), 42–49.

Costa, E., Lorena, A., Carvalho, A., Freitas, A., and Holden, N. (2007). Comparing several approaches for hierarchical classification of proteins with decision trees. In *Advances in bioinformatics and computational biology – LNBI 4643*, pages 126–137. Springer-Verlag Berlin/Heidelberg.

Craven, M. W. (1996). *Extracting comprehensible models from trained neural networks*. Ph.D. thesis, University of Wisconsin – Madison, Wisconsin, USA.

Debeljak, M., Squire, G. R., Kocev, D., Hawes, C., Young, M. W., and Džeroski, S. (2011). Analysis of time series data on agroecosystem vegetation using predictive clustering trees. *Ecological Modelling*, **x**(y), To appear.

Demšar, D., Debeljak, M., Džeroski, S., and Lavigne, C. (2005). Modelling pollen dispersal of genetically modified oilseed rape within the field. In *The Annual Meeting of the Ecological Society of America*.

Demšar, D., Džeroski, S., Larsen, T., Struyf, J., Axelsen, J., Bruns-Pedersen, M., and Krogh, P. H. (2006). Using multi-objective classification to model communities of soil. *Ecological Modelling*, **191**(1), 131–143.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, **7**, 1–30.

Dietterich, T. G. (2000a). Ensemble methods in machine learning. In *Proc. of the 1st International Workshop on Multiple Classifier Systems - LNCS 1857*, pages 1–15. Springer.

Dietterich, T. G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, **40**(2), 139–157.

Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2008). Hierchical annotation of medical images. In *Proceedings of the 11th International Multiconference - Information Society IS 2008*, pages 174–181. IJS, Ljubljana.

Domingos, P. (1998). Knowledge discovery via multiple models. *Intelligent Data Analysis*, **2**(1-4), 187–202.

Domingos, P. (2000). A unified bias-variance decomposition and its applications. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 231–238.

Domingos, P. and Pazzani, M. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, **29**(2), 103–130.

Džeroski, S. (2007). Towards a general framework for data mining. In S. Džeroski and J. Struyf, editors, *Knowledge Discovery in Inductive Databases, 5th International Workshop, KDID 2006, Revised Selected and Invited Papers*, volume 4747, pages 259–300.

Džeroski, S. and Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, **54**(3), 255–273.

Džeroski, S., Demšar, D., and Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, **13**(1), 7–17.

Džeroski, S., Panov, P., and Ženko, B. (2009). Ensemble methods in machine learning. In *Encyclopedia of complexity and systems science*, pages 5317–5325. Springer New York.

Džeroski, S., Goethals, B., and Panov, P. (2010). *Inductive databases and constraint-based data mining*. Springer.

Eisner, R., Poulin, B., Szafron, D., Lu, P., and Greiner, R. (2005). Improving protein function prediction using the hierarchical structure of the gene ontology. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–10.

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press.

Evgeniou, T., Micchelli, C. A., and Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, **6**, 615–637.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, **39**, 27–34.

Ferri, C., Hernández-Orallo, J., and Ramírez-Quintana, M. J. (2002). From ensemble methods to comprehensible models. In *Discovery Science – LNCS 2534*, pages 223–234. Springer Berlin/Heidelberg.

Freund, Y. and Mason, L. (1999). The alternating decision tree learning algorithm. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 124–133. Morgan Kaufmann Publishers Inc.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proc. of the Thirteenth International Conference on Machine Learning - ICML*, pages 148–156. Morgan Kaufman.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, **11**, 86–92.

Garofalakis, M., Hyun, D., Rastogi, R., and Shim, K. (2003). Building decision trees with constraints. *Data Mining and Knowledge Discovery*, **7**(2), 187–214.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, **4**(1), 1–58.

Geurts, P. (2001). Dual perturb and combine algorithm. In *Proceedings of AISTATS 2001, 8th International Workshop on Artificial Intelligence and Statistics*, pages 196–201.

Geurts, P., Ernst, D., and Wehenkel, L. (2006a). Extremely randomized trees. *Machine Learning*, **36**(1), 3–42.

Geurts, P., Wehenkel, L., and D'Alché-Buc, F. (2006b). Kernelizing the output of tree–based methods. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 345–352. ACM.

Giacinto, G. and Roli, F. (2001). An approach to the automatic design of multiple classifier systems. *Pattern Recognition Letters*, **22**(1), 25–33.

Gjorgjioski, V., Džeroski, S., and White, M. (2008). Clustering analysis of vegetation data. Technical Report 10065, Jožef Stefan Institute.

Greene, W. H. (2007). *Econometric analysis*. Prentice Hall, 6th edition.

Guan, Y., Myers, C. L., Hess, D. C., Barutcuoglu, Z., Caudy, A. A., and Troyanskaya, O. G. (2008). Predicting gene function in a hierarchical context with an ensemble of classifiers. *Genome biology*, **9**(S1), S3+.

Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**(10), 993–1001.

Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. Chapman & Hall.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(8), 832–844.

Huang, Y. S. and Suen, C. Y. (1995). A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**, 90–94.

Ianakiev, K. and Govindaraju, V. (2000). Architecture for classifier combination using entropy measures. In *Multiple Classifier Systems - LNCS 1857*, pages 340–350. Springer Berlin/Heidelberg.

Iman, R. L. and Davenport, J. M. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics - Theory and Methods*, **9**(6), 571–595.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, (37), 547–579.

Jong, K., Mary, J., Cornuéjols, A., Marchiori, E., and Sebag, M. (2004). Ensemble feature ranking. In *ECML PKDD '04: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases – LNCS 3202*, pages 267–278. Springer-Verlag.

Kampichler, C., Džeroski, S., and Wieland, R. (2000). Application of machine learning techniques to the analysis of soil ecological data bases: relationships between habitat features and collembolan community characteristics. *Soil Biology and Biochemistry*, **32**(2), 197–209.

Karalič, A. (1995). *First Order Regression*. Ph.D. thesis, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia.

Kargupta, H., Park, B.-H., and Dutta, H. (2006). Orthogonal decision trees. *IEEE Transactions on Knowledge and Data Engineering*, **18**(8), 1028–1042.

Kittler, J., Hatef, M., Duin, R. P. W., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(3), 226–239.

Klimt, B. and Yang, Y. (2004). The enron corpus: A new dataset for email classification research. In *ECML '04: Proceedings of the 18th European Conference on Machine Learning – LNCS 3201*, pages 217–226. Springer Berlin / Heidelberg.

Kocev, D., Struyf, J., and Džeroski, S. (2007a). Beam search induction and similarity constraints for predictive clustering trees. In *Proc. of the 5th International Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 4747*, pages 134–151.

Kocev, D., Vens, C., Struyf, J., and Džeroski, S. (2007b). Ensembles of multi–objective decision trees. In *ECML '07: Proceedings of the 18th European Conference on Machine Learning – LNCS 4701*, pages 624–631. Springer Berlin / Heidelberg.

Kocev, D., Slavkov, I., and Džeroski, S. (2008). More in better: ranking with multiple targets for biomarker discovery. In *Proc. 2nd Intl Wshp on Machine Learning in Systems Biology*, page 133.

Kocev, D., Džeroski, S., White, M., Newell, G., and Griffioen, P. (2009). Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling*, **220**(8), 1159–1168.

Kocev, D., Naumoski, A., Mitreski, K., Krstić, S., and Džeroski, S. (2010). Learning habitat models for the diatom community in lake prespa. *Ecological Modelling*, **221**(2), 330–337.

Kong, E. B. and Dietterich, T. G. (1995). Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321.

Kuncheva, L. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience.

Kuncheva, L. and Whitaker, C. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, **51**, 181–207.

Langley, P. (1996). *Elements of machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Lee, H., Tu, Z., Deng, M., Sun, F., and Chen, T. (2006). Diffusion kernel–based logistic regression models for protein function prediction. *OMICS: A Journal of Integrative Biology*, **10**(1), 40–55.

Lehmann, T., Schubert, H., Keysers, D., Kohnen, M., and Wein, B. (2003). The irma code for unique classification of medical images. In *Medical Imaging 2003: PACS and Integrated Medical Information Systems: Design and Evaluation*, pages 440–451.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, **5**, 361–397.

Liao, T. W. (2005). Clustering of time series data–a survey. *Pattern Recognition*, **38**(11), 1857–1874.

Marée, R., Geurts, P., and Wehenkel, L. (2007). Random subwindows and extremely randomized trees for image classification in cell biology. *BMC Cell Biology*, **8**(Suppl 1), S2.

Marée, R., Geurts, P., and Wehenkel, L. (2009). Content-based image retrieval by indexing random subwindows with randomized trees. *IPSJ Transactions on Computer Vision and Applications*, **1**, 46–57.

Mason, L., Bartlett, P. L., and Baxter, J. (2000). Improved generalization through explicit optimization of margins. *Machine Learning*, **38**(3), 243–255.

McCarthy, J., Minsky, M., Rochester, N., and Shannon, C. (1955). A proposal for the Dartmouth summer research project on artificial intelligence. http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html.

Merz, C. J. (1999). Using correspondence analysis to combine classifiers. *Machine Learning*, **36**(1), 33–58.

Micchelli, C. A. and Pontil, M. (2004). Kernels for multi–task learning. In *Advances in Neural Information Processing Systems 17 - Proceedings of the 2004 Conference*, pages 921–928.

Mitchell, T. (1997). *Machine learning*. McGraw Hill.

Moosmann, F., Nowak, E., and Jurie, F. (2008). Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **30**(9), 1632–1646.

Mostafavi, S., Ray, D., Warde-Farley, D., Grouios, C., and Morris, Q. (2008). Genemania: a real-time multiple association network integration algorithm for predicting gene function. *Genome biology*, **9**(S1), S4+.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons*. Ph.D. thesis, Princeton University, Princeton, NY, USA.

Obozinski, G., Lanckriet, G., Grant, C., Jordan, M. I., and Noble, W. S. (2008). Consistent probabilistic outputs for protein function prediction. *Genome Biology*, **9**(S1), S6+.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, **11**, 169–198.

Panov, P. and Džeroski, S. (2007). Combining bagging and random subspaces to create better ensembles. In *Advances in Intelligent Data Analysis VII - LNCS 4723*, pages 118–129. Springer Berlin/Heidelberg.

Pérez, J. M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I., and Martín, J. I. (2004). Behavior of consolidated trees when using resampling techniques. In *Pattern Recognition in Information Systems, Proceedings of the 4th International Workshop on Pattern Recognition in Information Systems*, pages 139–148. INSTICC Press.

Pesquita, C., Faria, D., Bastos, H., Falcão, A. O., and Couto, F. M. (2007). Evaluating go-based semantic similarity measures. In *10th Annual Bio-Ontologies Meeting (Bio-Ontologies 2007)*, pages 37–40.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 1–8.

Quinlan, R. J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1 edition.

Raviv, Y. and Intrator, N. (1996). Bootstrapping with noise: An effective regularization technique. *Connection Science*, **8**, 355–372.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009). Classifier chains for multi-label classification. In *Machine Learning and Knowledge Discovery in Databases - LNCS 5782*, pages 254–269. Springer Berlin/Heidelberg.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2006). Kernel–based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, **7**, 1601–1626.

Saeys, Y., Abeel, T., and Peer, Y. (2008). Robust feature selection using ensemble feature selection techniques. In *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases – LNCS 5212*, pages 313–325. Springer-Verlag.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **26**(1), 43–49.

Schapire, R., Freund, Y., Bartlett, P., and Lee, W. S. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, **26**(5), 322–330.

Schramm, A., Schulte, J. H., Klein-Hitpass, L., Havers, W., Sieverts, H., Berwanger, B., Christiansen, H., Warnat, P., Brors, B., Eils, J., Eils, R., and Eggert, A. (2004). Prediction of clinical outcome and biological characterization of neuroblastoma by expression profiling. *Oncogene*, **24**, 7902–7912.

Seni, G. and Elder, J. F. (2010). *Ensemble methods in data mining: Improving accuracy through combining predictions*. Morgan & Claypool Publishers.

Silla, C. and Freitas, A. (2010). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, pages 1–42.

Skrjanc, M., Grobelnik, M., and Zupanic, D. (2001). Insights offered by data-mining when analyzing media space data. *Informatica (Slovenia)*, **25**(3), 357–363.

Slavkov, I., Ženko, B., and Džeroski, S. (2010a). Evaluation method for feature rankings and their aggregations for biomarker discovery. In *Proc. 3rd Intl Wshp on Machine Learning in Systems Biology, JMLR: Workshop and Conference Proceedings 8*, pages 14–29. Microtome Publishing.

Slavkov, I., Gjorgjioski, V., Struyf, J., and Džeroski, S. (2010b). Finding explained groups of time-course gene expression profiles with predictive clustering trees. *Molecular BioSystems*, **6**(4), 729–740.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, **45**(4), 427–437.

Sokolova, M., Japkowicz, N., and Szpakowicz, S. (2006). Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *AI 2006: Advances in Artificial Intelligence - LNCS 4304*, pages 1015–1021. Springer Berlin / Heidelberg.

Stojanova, D. (2009). *Estimating Forest Properties from Remotely Sensed Data by using Machine Learning*. Master's thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., and Džeroski, S. (2010). Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics*, **5**(4), 256–266.

Struyf, J. and Džeroski, S. (2006). Constraint based induction of multi-objective regression trees. In *Proc. of the 4th International Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 3933*, pages 222–233. Springer.

Struyf, J., Džeroski, S., Blockeel, H., and Clare, A. (2005). Hierarchical multi-classification with predictive clustering trees in functional genomics. In *Progress in Artificial Intelligence - LNCS 3808*, pages 272–283. Springer Berlin/Heidelberg.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison Wesley.

Thrun, S. and Pratt, L. (1998). *Learning to learn*. Kluwer Academic Publishers.

Tian, W., Zhang, L. V., Taşan, M., Gibbons, F. D., King, O. D., Park, J., Wunderlich, Z., Cherry, J. M., and Roth, F. P. (2008). Combining guilt–by–association and guilt–by–profiling to predict saccharomyces cerevisiae gene function. *Genome biology*, **9**(S1), S7+.

Todorovski, L., Cestnik, B., Kline, M., Lavrač, N., and Džeroski, S. (2002). Qualitative clustering of short time-series: A case study of firms reputation data. In *ECML/PKDD'02 Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning*, pages 141–149.

Tommasi, T., Caputo, B., Welter, P., Güld, M., and Deserno, T. (2010). Overview of the clef 2009 medical image annotation track. In *Multilingual Information Access Evaluation II. Multimedia Experiments – LNCS 6242*, pages 85–93. Springer Berlin/Heidelberg.

Triviño-Rodriguez, J., Ruiz-Sepúlveda, A., and Morales-Bueno, R. (2008). How an ensemble method can compute a comprehensible model. In *Data Warehousing and Knowledge Discovery – LNCS 5182*, pages 368–378. Springer Berlin/Heidelberg.

Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. (2008). Multilabel classification of music into emotions. In *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 325–330.

Valentini, G. (2003). *Ensemble methods based on bias-variance analysis*. Ph.D. thesis, Universita' di Genova, Genova, Italy.

Valentini, G. and Re, M. (2009). Weighted true path rule: a multilabel hierarchical algorithm for gene function prediction. In *Proceedings of the 1st International Workshop on Learning from Multi-Label Data*, pages 133–146.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., and Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, **73**(2), 185–214.

Ženko, B. (2007). *Learning predictive clustering rules*. Ph.D. thesis, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia.

Wernecke, K. D. (1992). A coupling procedure for discrimination of mixed data. *Biometrics*, **48**(2), 497–506.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. (2007). Multi-task reinforcement learning: a hierarchical bayesian approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 1015–1022. ACM.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, **5**(1), 241–259.

Yang, Q. and Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, **5**(4), 597–604.

Appendix 1: Complete results

We give complete results for the three considered tasks: predicting of multiple continuous targets, predicting of multiple discrete targets and hierarchical multi-label classification. For each task, we give the saturation curves, statistical tests for the predictive performance and efficiency over all ensemble sizes.

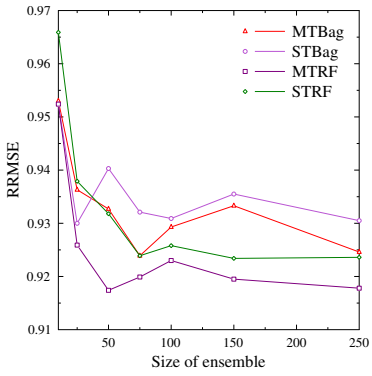
The results are from the Friedman test for multiple hypothesis testing and post-hoc Nemenyi test. The average rank diagrams are obtained using the critical distance at a significance level of 0.05. The differences in performance of the algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The algorithm names are abbreviated as follows:

- **MTRT**: Multi-target regression tree
- **STRT**: Single-target regression tree
- **MTRF**: Multi-target random forest
- **STRF**: Single-target random forest
- **MTBag**: Multi-target bagging
- **STBag**: Single-target bagging
- **HMCPCT**: PCT for HMC
- **HSCPCT**: PCT for HSC
- **HMCRF**: Random forest of PCTs for HMC
- **HSCRF**: Random forest of PCTs for HSC
- **HMCBag**: Bagging of PCTs for HMC
- **HSCBag**: Bagging of PCTs for HSC

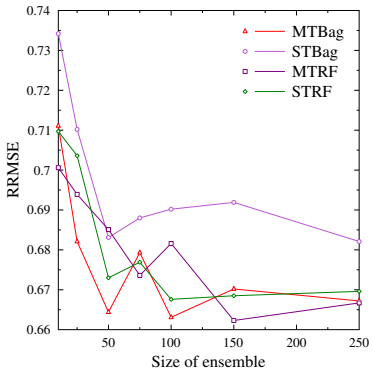
8.1 Prediction of multiple continuous targets

8.1.1 Saturation curves

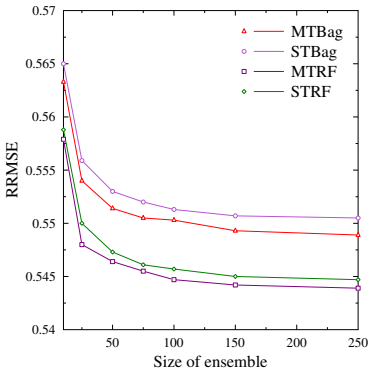
Collembola



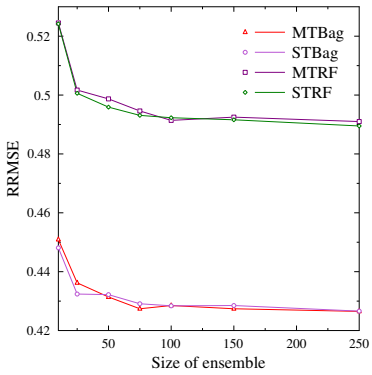
EDM



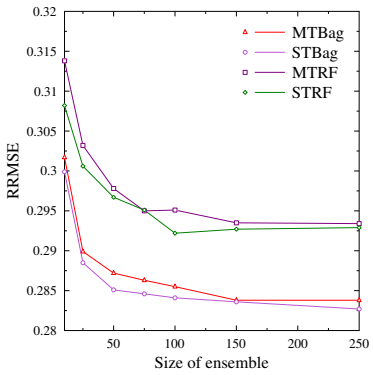
Forestry-Kras



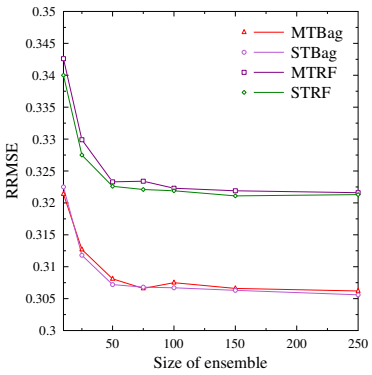
Forestry-Slivnica-LandSat



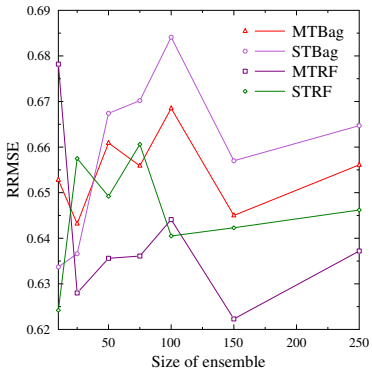
Forestry-Slivnica-IRS



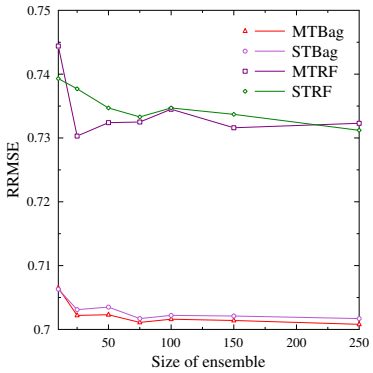
Forestry-Slivnica-SPOT



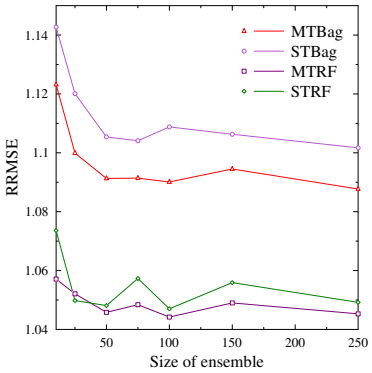
Sigmea real



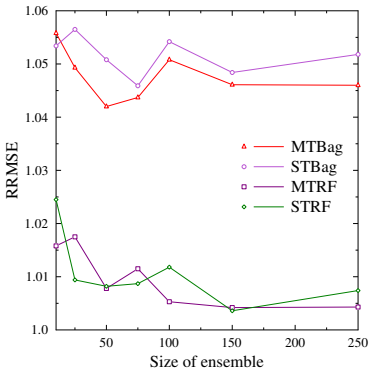
Soil quality



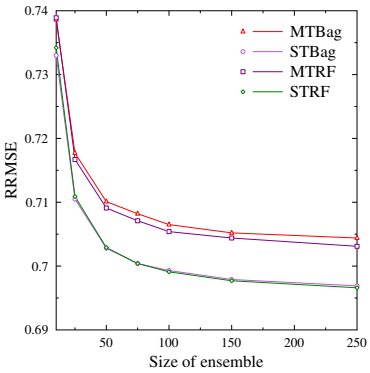
Solar-flare 1



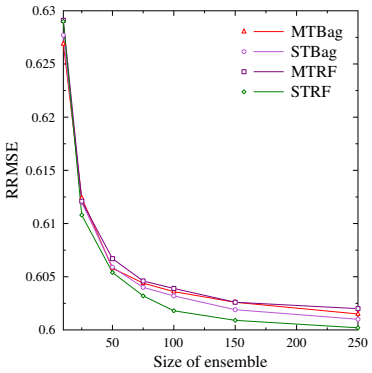
Solar-flare 2



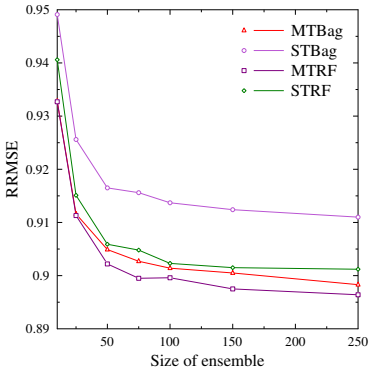
Vegetation clustering



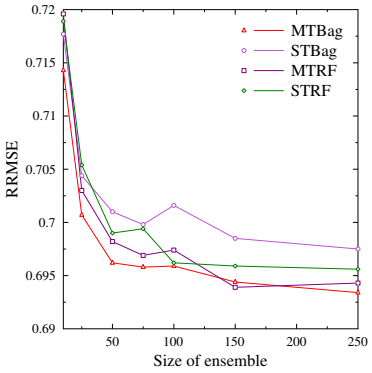
Vegetation condition



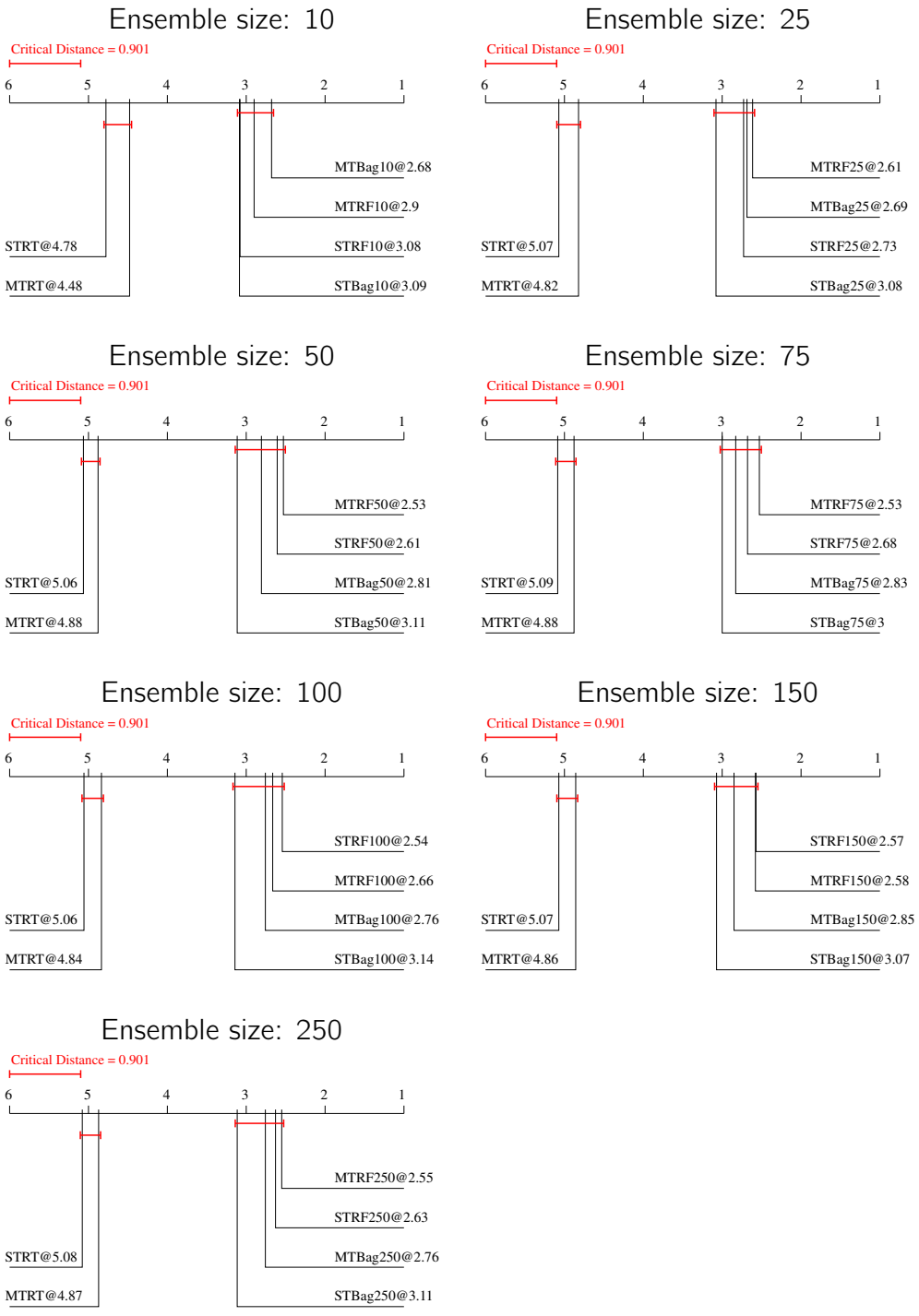
Water quality



Overall



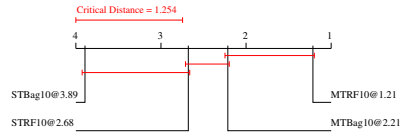
8.1.2 Statistical tests for predictive performance



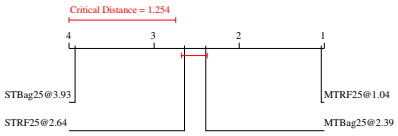
8.1.3 Statistical tests for efficiency

Time consumption

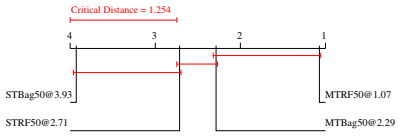
Ensemble size: 10



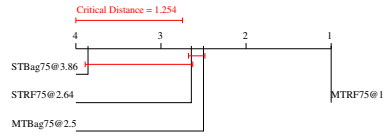
Ensemble size: 25



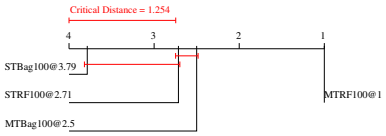
Ensemble size: 50



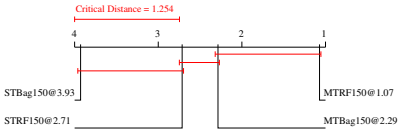
Ensemble size: 75



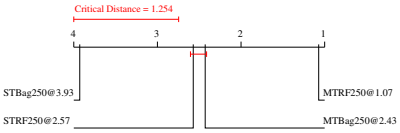
Ensemble size: 100



Ensemble size: 150

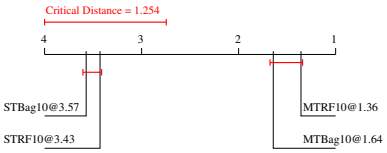


Ensemble size: 250

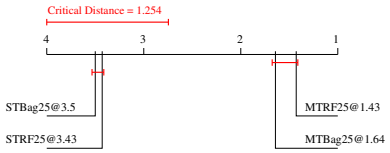


Model size

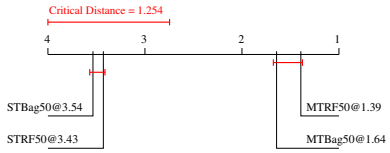
Ensemble size: 10



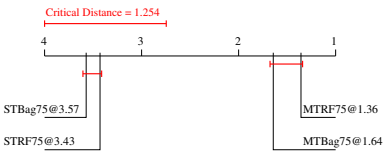
Ensemble size: 25



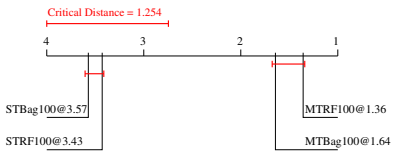
Ensemble size: 50



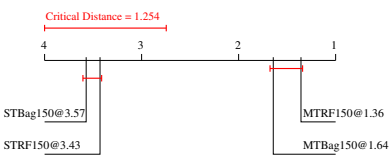
Ensemble size: 75



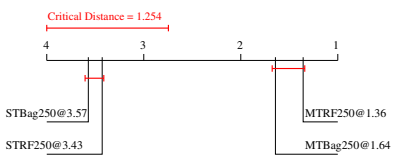
Ensemble size: 100



Ensemble size: 150

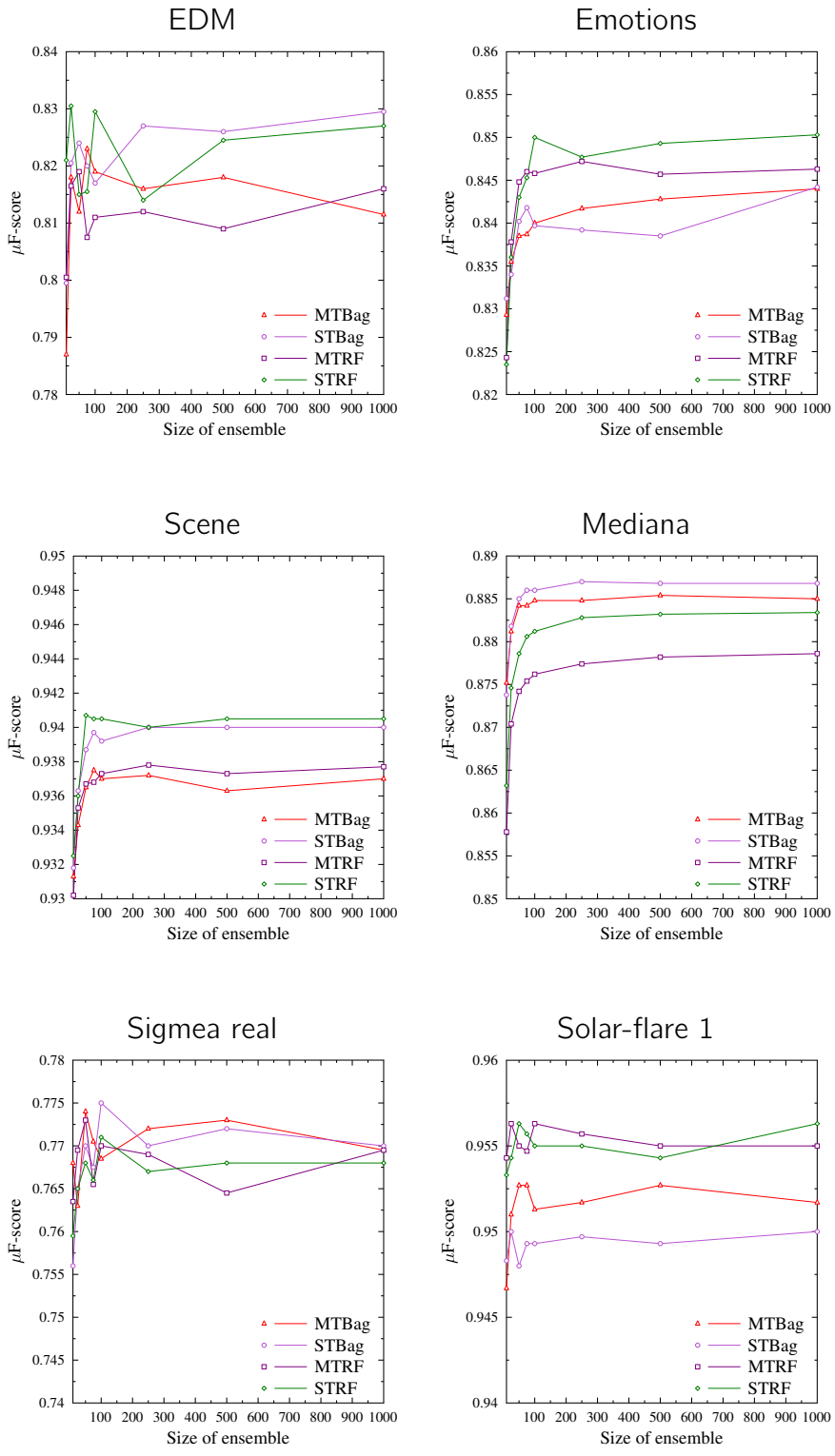


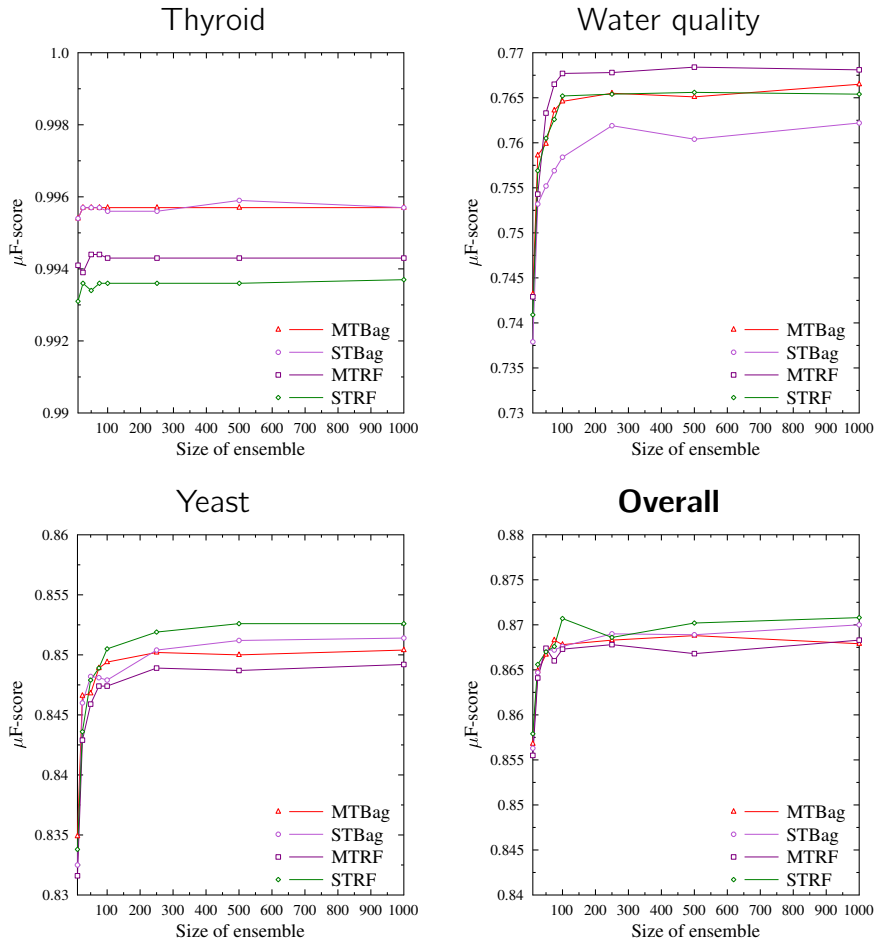
Ensemble size: 250



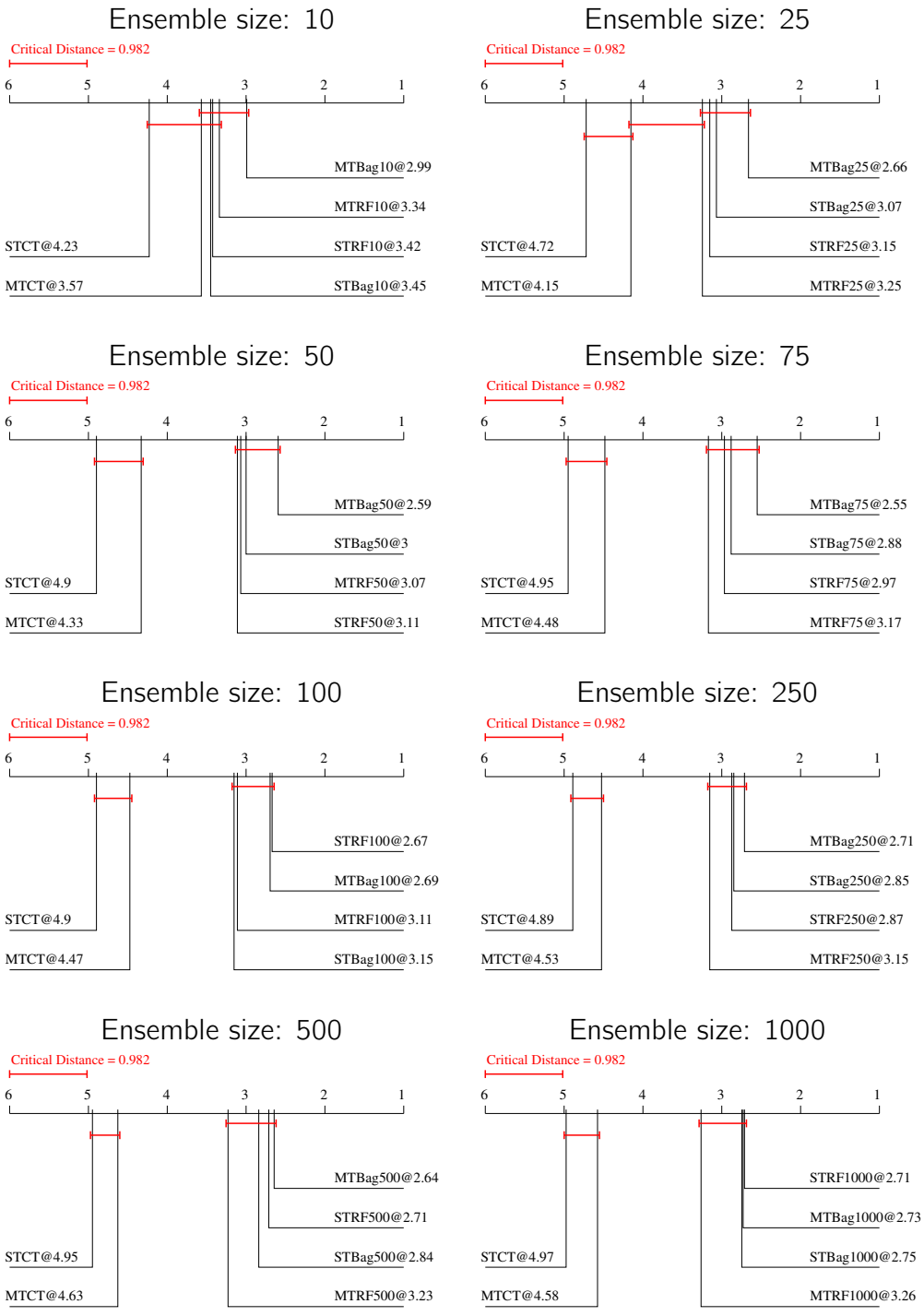
8.2 Prediction of multiple discrete targets

8.2.1 Saturation curves




Thyroid
Water quality
Yeast
Overall
μF-score
Size of ensemble
MTBag
STBag
MTRF
STRF

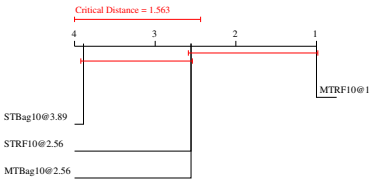
8.2.2 Statistical tests for predictive performance



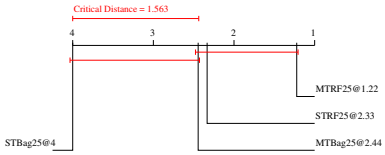
8.2.3 Statistical tests for efficiency

Time consumption

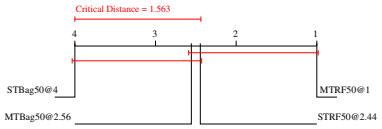
Ensemble size: 10



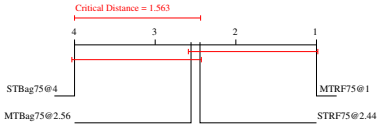
Ensemble size: 25



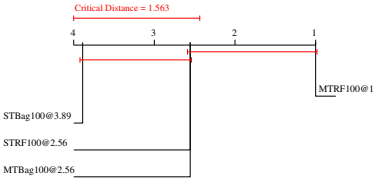
Ensemble size: 50



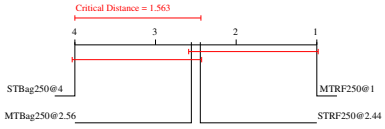
Ensemble size: 75



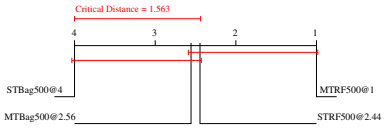
Ensemble size: 100



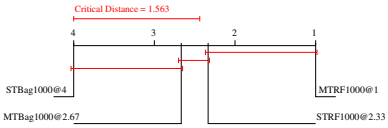
Ensemble size: 250



Ensemble size: 500

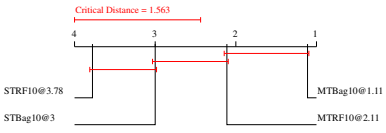


Ensemble size: 1000

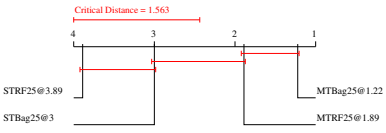


Model size

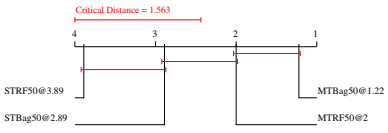
Ensemble size: 10



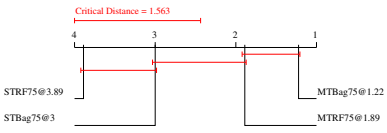
Ensemble size: 25



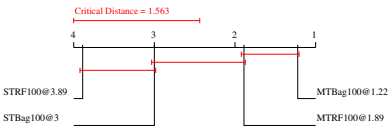
Ensemble size: 50



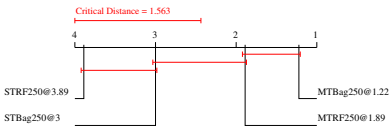
Ensemble size: 75



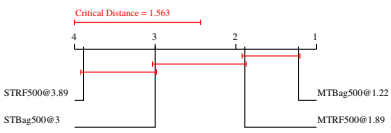
Ensemble size: 100



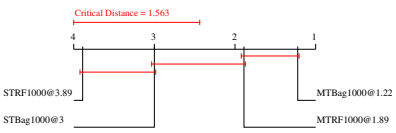
Ensemble size: 250



Ensemble size: 500

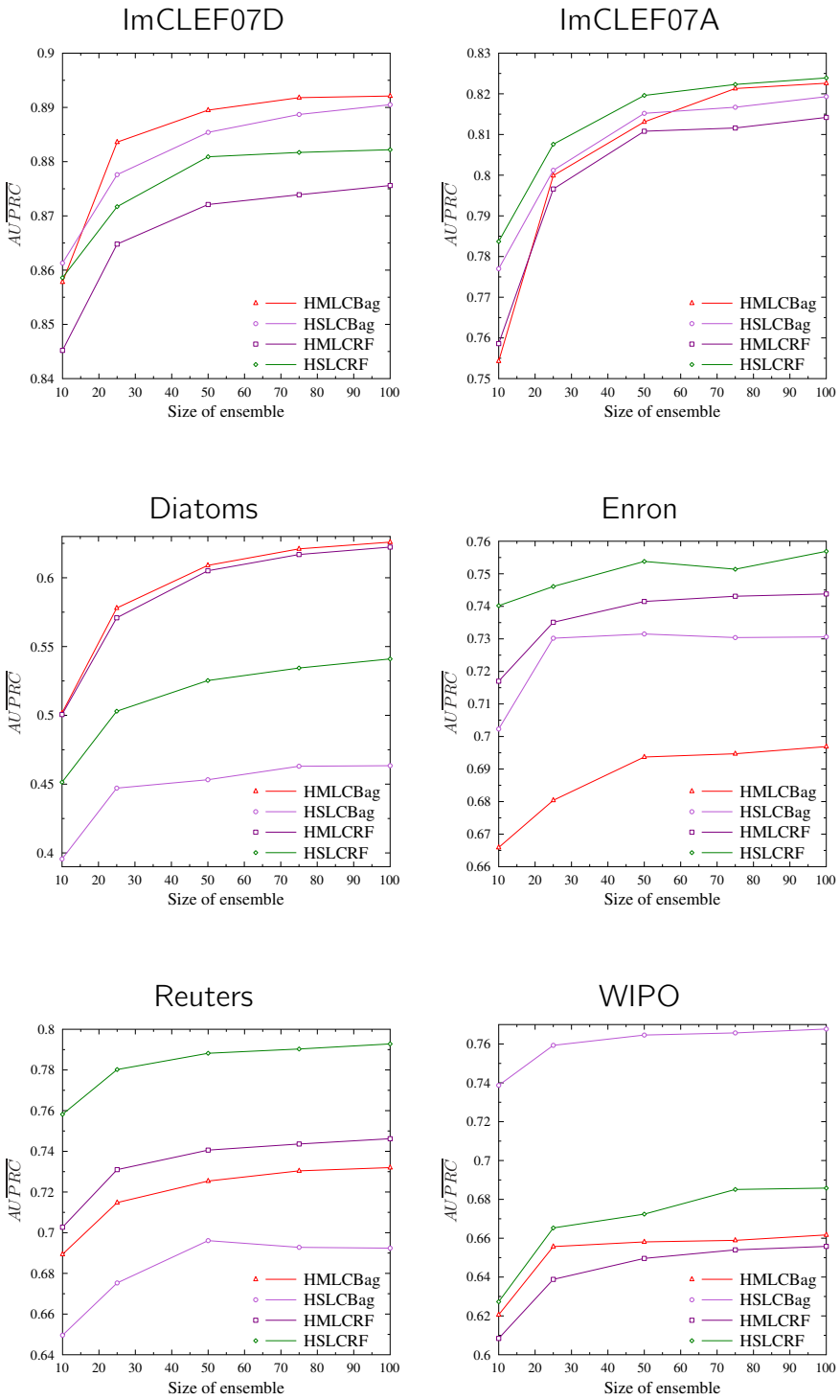


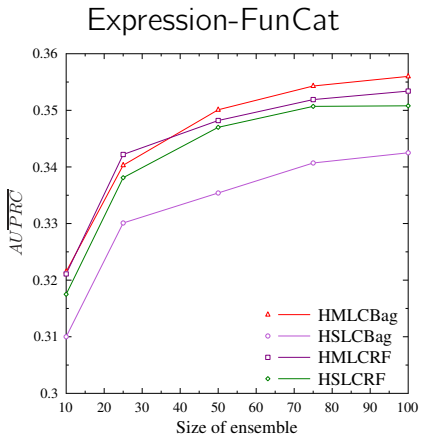
Ensemble size: 1000

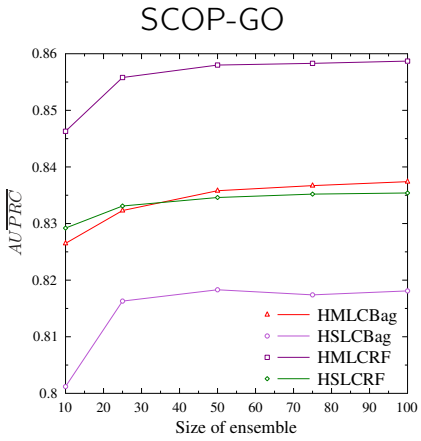


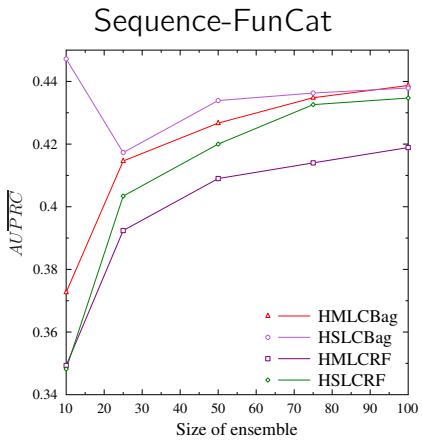
8.3 Hierarchical multi-label classification

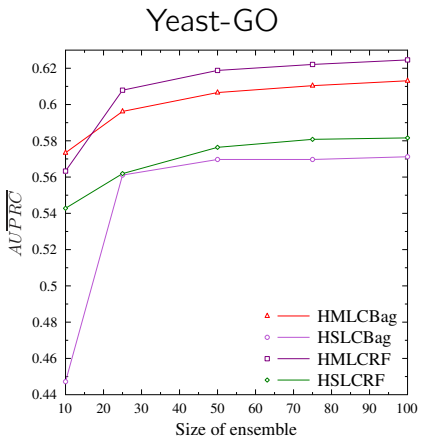
8.3.1 Saturation curves

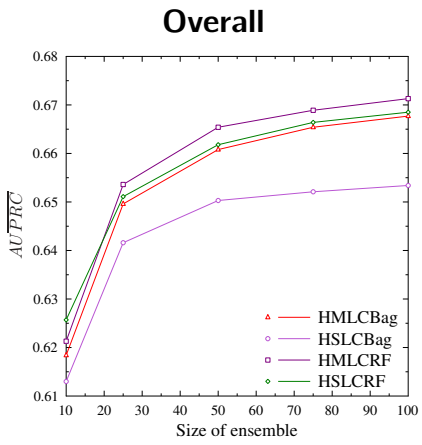



Expression-FunCat
$\overline{AUPRC}$
Size of ensemble
HMLCBag
HSLCBag
HMLCRF
HSLCRF

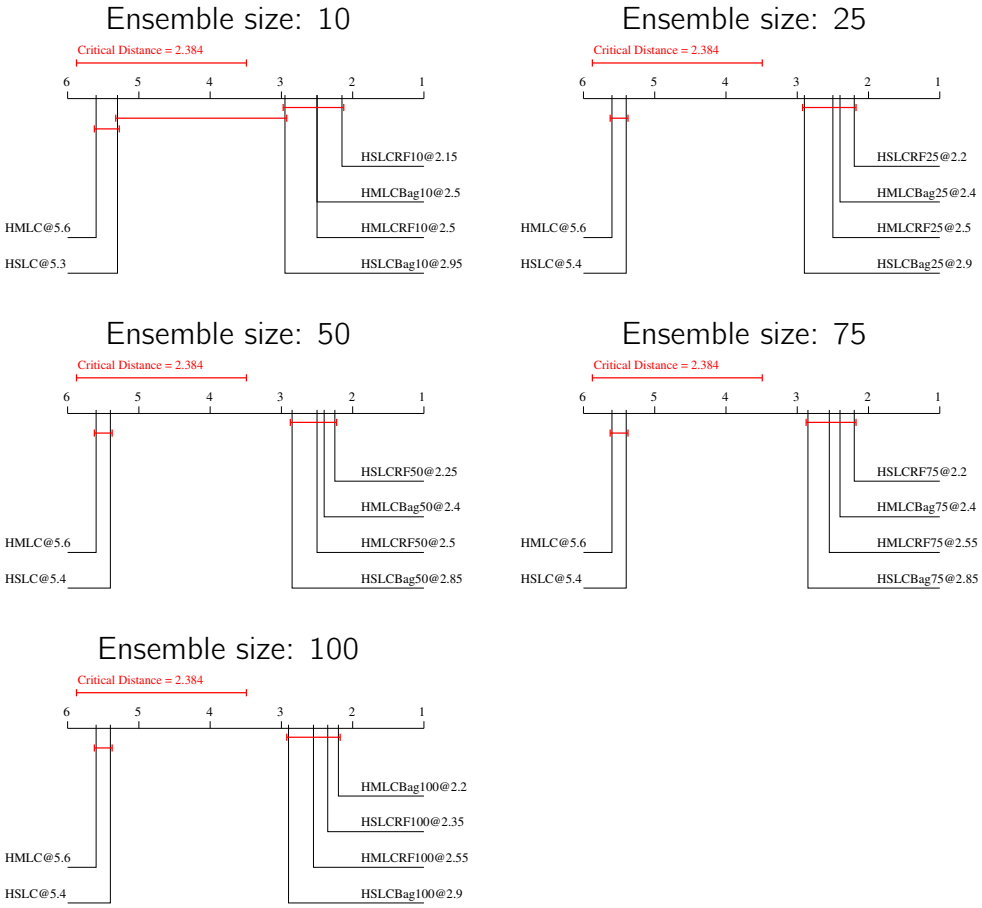

SCOP-GO
$\overline{AUPRC}$
Size of ensemble
HMLCBag
HSLCBag
HMLCRF
HSLCRF


Sequence-FunCat
$\overline{AUPRC}$
Size of ensemble
HMLCBag
HSLCBag
HMLCRF
HSLCRF


Yeast-GO
$\overline{AUPRC}$
Size of ensemble
HMLCBag
HSLCBag
HMLCRF
HSLCRF


Overall
$\overline{AUPRC}$
Size of ensemble
HMLCBag
HSLCBag
HMLCRF
HSLCRF

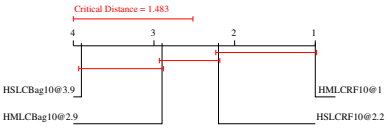
8.3.2 Statistical tests for predictive performance



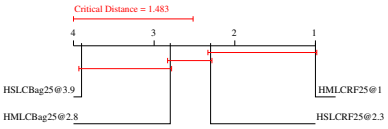
8.3.3 Statistical tests for efficiency

Time consumption

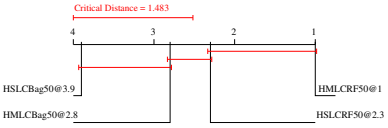
Ensemble size: 10



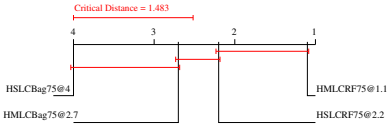
Ensemble size: 25



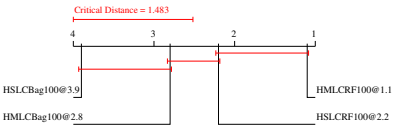
Ensemble size: 50



Ensemble size: 75

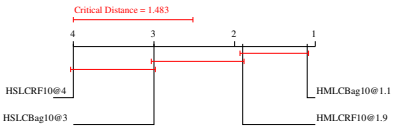


Ensemble size: 100

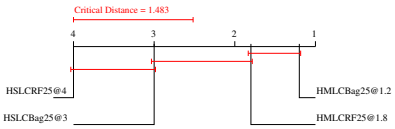


Model size

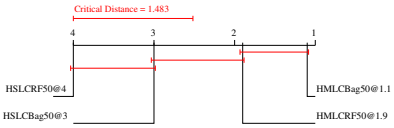
Ensemble size: 10



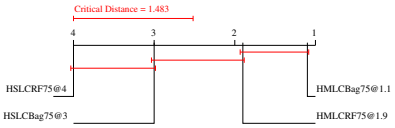
Ensemble size: 25



Ensemble size: 50



Ensemble size: 75



Ensemble size: 100