ENSEMBLES FOR PREDICTING STRUCTURED OUTPUTS

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia, March 2011

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

Evaluation Board:
Prof. Dr. Nada Lavrač, Chair, Jožef Stefan Institute, Ljubljana, Slovenia
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

March 2011

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

To my parents

На моите родители

Contents

Abstract

In this thesis, we address the task of learning models for predicting structured outputs, which take as input a tuple of attribute values and produce as output a structured object. In contrast to classification and regression, where the output is a single scalar value, in our case the output is a data structure, such as a tuple or a directed acyclic graph. We consider both global and local prediction of structured outputs, the first based on a single model that predicts the entire output structure and the latter based on a collection of models, each predicting a component of the output structure.

In particular, we take the notion of an ensemble, i.e., a collection of predictive models whose predictions are combined, and apply it in the context of predicting structured outputs. Ensembles have proved to be highly effective methods for improving the predictive performance of their constituent models, especially for classification tree models. We propose in this thesis to build ensemble models consisting of predictive clustering trees, which generalize classification trees: these have been used for predicting different types of structured outputs, both locally and globally.

More specifically, we develop methods for learning several different types of ensembles of predictive clustering trees for global and local prediction of different types of structured outputs. The types of outputs considered correspond to different predictive modeling tasks, i.e., multi-target regression, multi-target classification, and hierarchical multi-label classification. The different types of ensembles include bagging, random forests, random subspaces and bagging of subspaces. Each of the combinations can be applied both in the context of global prediction (producing a single ensemble) or local prediction (producing a collection of ensembles).

We conduct an extensive experimental evaluation of bagging and random forests for structured prediction across a range of benchmark datasets for each of the three types of structured outputs. We compare ensembles for global and local prediction, as well as single trees for global prediction and tree collections for local prediction, both in terms of predictive performance and in terms of

efficiency (running times and model complexity). Both global and local ensembles perform better than the single model counterparts in terms of predictive power. Global and local ensembles perform equally well, with global ensembles being more efficient and producing smaller models, as well as needing fewer trees in the ensemble to achieve the maximal performance.

We also analyze the computational complexity of the methods theoretically. The theoretical analyses are consistent with the empirical evidence, showing that the global ensembles are most efficient, especially random forests. The analyses also indicate that the proposed approaches are scalable to datasets, which can be large along any of the following dimensions: number of attributes, number of examples, and size of the target.

We apply the developed ensemble approaches to three practically relevant problems, resulting in three detailed case studies: we compare our results to results of state-of-the-art methods used in the respective domains. First, we constructed models to better understand the resilience of some vegetation types and generated maps of the vegetation condition in the sate of Victoria, Australia. Next, on the task of hierarchical annotation of medical X-ray images, the ensembles for predicting structured outputs provided the best annotation results reported so far in the literature. Finally, extensive experimental evaluation over several tasks of gene function prediction in three organisms showed that bagging for predicting structured outputs is superior to or competitive with state-of-the-art methods in this area.

Finally, we present some preliminary results that further explore the proposed paradigm of ensembles for structured prediction. We first discuss structured prediction for different types of structured outputs and correspondingly different distance measures on these. We further propose a method for feature ranking in the context of structured outputs, based on random forests. Moreover, we suggest a novel ensemble learning method that is based on the beam search strategy and can control directly the diversity in the ensemble. With this, we open a number of avenues for further developments.

Povzetek

Disertacija obravnava nalogo učenja modelov za napovedovanje strukturiranih vrednosti, ki kot vhod vzamejo vektor vrednosti značilk in na izhodu podajo napoved strukturirane vrednosti. Za razliko od klasifikacije in regresije, kjer je na izhodu ena sama skalarna vrednost, je v našem primeru izhod, oziroma napoved, podatkovna struktura kot na primer vektor ali usmerjeni graf brez ciklov. Obravnavamo tako globalno kot lokalno napovedovanje strukturiranih vrednosti: pri prvem uporabimo en sam model, ki poda napoved celotne strukture, v drugem pa uporabimo zbirko modelov od katerih vsak napove del izhodne strukture.

Pojem ansambla, to je zbirke napovednih modelov čigar napovedi združimo, smo razširili na kontekst napovedovanja strukturiranih vrednosti. Ansambli so izjemno učinkovit način za izboljšanje napovedne moči sestavnih modelov, še zlasti v primeru modelov v obliki klasifikacijskih dreves. V tej disertaciji predlagamo gradnjo ansamblov, ki vsebujejo drevesa za napovedno razvrščanje, to je posplošena odločitvena drevesa: le ta so bila uporabljena za napovedovanje različnih vrst strukturiranih vrednosti tako lokalno kot globalno.

Razvijemo metode za učenje več vrst ansamblov dreves za napovedno razvrščanje tako za globalno kot lokalno napovedovanje različnih vrst strukturiranih vrednosti. Obravnavamo različne naloge napovedovanja strukturiranih vrednosti, kot so večciljna regresija, večciljna klasifikacija ter hierarhična večznačkovna klasifikacija. Različne vrste ansamblov zajemajo metodo bagging, metodo naključne gozdove, metodo naključnih podprostorov ter metodo bagging podprostorov. Kombinacije le teh lahko uporabimo tako v kontekstu globalnega napovedovanja z enim samim ansamblom kot tudi lokalnega napovedovanja z zbirko ansamblov.

Temeljito ovrednotimo metode za gradnjo ansamblov za napovedovanje strukturiranih vrednosti, še posebej metodi bagging in naključni gozdovi, na vrsti množic podatkov za vsakega od treh tipov strukturiranih vrednosti. Primerjamo ansamble za globalno in lokalno napovedovanje, kot tudi posamezna drevesa za globalno napovedovanje ter zbirke dreves za lokalno napovedovanje: primer-

jamo napovedno moč modelov, oziroma metod za gradnjo ansamblov kot tudi njihovo učinkovitost, to je čas izvajanja ter kompleksnost nastalih modelov. Tako globalni kot lokalni ansambli imajo večjo napovedno moč kot individualni modeli. Globalni in lokalni ansambli imajo enako napovedno moč, pri čemer so globalni ansambli bolj učinkoviti in dajo manjše modele. Slednji potrebujejo tudi manj dreves v ansamblu za doseganje maksimalne napovedne točnosti.

Opravimo tudi teoretično analizo računske kompleksnosti predlaganih metod. Teoretične analize so v skladu z eksperimentalnimi rezultati, ki kažejo da so najbolj učinkoviti globalni ansambli, še posebej naključni gozdovi. Analize tudi kažejo da predlagane pristope lahko uporabimo za velike podatkovne množice, ki so lahko velike po vsaki od naslednjih dimenzij: število atributov, število primerov in velikost strukturiranega izhoda.

Razvite metode za gradnjo ansamblov uporabimo na treh praktično relevantnih problemih in s tem dobimo tri podrobne študije primerov: primerjamo naše rezultate z rezultati trenutno najboljših metod na ustreznih področjih uporabe. Najprej, za boljše razumevanje odpornosti nekaterih vrst vegetacije, zgradimo napovedne modele in jih uporabimo za generiranje kart stanja vegetacije v državi Viktorija v Avstraliji. Nato na nalogi hierarhičnega označevanja medicinskih rentgenskih posnetkov z ansambli za napovedovanje strukturiranih vrednosti, dosežemo najboljše do sedaj objavljene rezultate označevanja. Končno s temeljito eksperimentalno primerjavo na nekaj nalogah napovedovanja funkcij genov v treh organizmih pokažemo, da z metodo bagging za gradnjo ansamblov za napovedovanje strukturiranih vrednosti dosežemo boljše ali primerljive rezultate z rezultati najboljših obstoječih metod na tem področju.

Na koncu predstavimo nekaj preliminarnih rezultatov, ki nadalje izkoristijo predlagane pristope k gradnji ansamblov za napovedovanje strukturiranih vrednosti. Najprej obravnavamo napovedovanje različnih vrst strukturiranih vrednosti, tudi takih, ki jih še nismo obravnavali ter ustreznih mer razdalje nad njimi. Nato predlagamo metodo za rangiranje značilk v kontekstu napovedovanja strukturiranih vrednosti, ki temelji na metodi naključnih gozdov. Nadalje predlagamo novo metodo za gradnjo ansamblov, ki temelji na strategiji preiskovanja s snopom in lahko neposredno nadzoruje stopnjo raznolikosti napovedi v ansamblu. S tem odpremo vrsto smernic za nadaljne raziskave.

Abbreviations

$\mu F - score$	=	micro weighted average F-score
$AUPRC$	=	area under the precision-recall curve
$AU\overline{PRC}$	=	area under the average precision-recall curve
Bag	=	bagging
CLUS-BS	=	implementation of CLUS for beam search induction of trees
CLUS-BS-S	=	implementation of CLUS for beam search induction of trees with similarity constraint
CLUS-ENS-HMC	=	implementation of CLUS for learning ensembles for hierarchical multi-label classification
CLUS-HMC	=	implementation of CLUS for learning trees for hierarchical multi-label classification
CLUS-MTDT	=	implementation of CLUS for learning multi-target decision trees
HMC	=	hierarchical multi-label classification
HSC	=	hierarchical single-label classification
MT	=	multi target
PCT	=	predictive clustering tree
ST	=	random forests
$RRMSE$	=	relative root mean squared error
TDIDT	=	top-down induction of decision tree
ST	=	single target
SVM	=	support vector machine

1 Introduction

We first present an overview of the thesis and motivate it within its research area. We start by outlying the context of the work performed. Next, we state the motivation and the original contributions of the thesis. Finally, we present the structure of the remainder 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 computer programs that have the ability to learn, i.e., improve with experience (Mitchell, 1997). A very significant part of the research in machine learning is concerned with extracting new knowledge from available data where the experience is given in the form of training examples (instances). This type of machine learning is called inductive learning (Bratko, 2000).

In the classical inductive learning setting, the available training examples are given in the form of a table. Each row of the table is an example and each column is a property of the example (called an attribute). If the goal is to predict the value of one property of the examples (called a target attribute or output) using the values of the remaining properties (called descriptive or input attributes), then the task is called *predictive modelling* (or supervised learning). On the other hand, if such a 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 (SVMs) and examples of machine learning methods for descriptive modelling include clustering, association rule learning and principal-component analysis (Bishop, 2007).

Predictive and descriptive modelling are considered as different machine learning tasks. However, the predictive modelling can be seen as a special case of clustering (Blockeel, 1998). In this case, the goal of predictive modelling is to identify clusters with reasonable

size that are compact in the target space (i.e., group the instances with similar values of the target variable). The goal of the descriptive modelling, on the other hand, is to identify reasonably sized clusters compact in the descriptive space (i.e., group the instances with similar values of the descriptive variables). Blockeel (1998) presented a machine learning task, called *predictive clustering*, that combines the notions of predictive and descriptive modelling. Predictive clustering identifies clusters that are compact in both the target and the descriptive space. The methods presented in this thesis are based on the *predictive clustering* framework.

Predictive and descriptive modelling are connected through machine learning methods that partition the space of 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 decision trees for predictive clustering, called predictive clustering trees (PCTs), and Ženko (2007) developed decision rules for predictive clustering, called predictive clustering rules. These methods, in addition to providing clusters of instances, provide symbolic descriptions of the clusters. The clusters from the decision trees are described by the conjunction of 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 rules' conditions.

Predictive modelling methods are used for predicting an output (i.e., target attribute) for an example. Typically, the output 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, namely, predictive modelling 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, 2011): (1) methods that predict component(s) of the output and then combine the components 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 exploit and use the dependencies that exist between the components of the structured output in the model learning phase and thus result in better predictive performance. Next, they are more efficient: it can easily happen that the number of components in the output is 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. Methods from the predictive clustering framework belong to the group of global approaches.

In the predictive clustering framework, the following tasks were considered so far: predicting multiple discrete variables, predicting multiple continuous variables, hierarchical multi-label classification (HMC) and prediction of short time series. The variance and prototype functions (needed for the induction of the trees and the rules) were designed appropriately for each task. PCT methods for each of the tasks have been evaluated empirically and have been shown to have the advantages of the global methods stated above.

In this thesis, we extend the predictive clustering framework in the context of ensemble methods to further increase the predictive performance of predictive clustering trees. Ensemble methods construct a set of predictive models (an ensemble) and combine their outputs to obtain a single prediction (Seni and Elder, 2010). There are many practical studies and theoretical explanations that show that ensembles achieve high predictive performance and lift the predictive performance of a single predictive model, both for classification and regression (Kuncheva, 2004; Seni and Elder, 2010).

Ensemble methods can differ in how they construct the set of constituent (or base) predictive models and in how they combine their predictions. Having in mind that combining identical or very similar predictive models will not produce an increase of predictive performance, it only makes sense to construct ensembles of predictive models that are diverse. The diversity in the ensemble can be obtained by learning heterogeneous predictive models, by modifying the training set or by changing the learning algorithm. To obtain a prediction of the ensemble, model selection or model fusion can be used (Džeroski *et al.*, 2009). The former selects the best predictive model and uses its predictions as predictions of the ensemble. The latter combines the predictions of all base predictive models 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)).

1.2 Motivation

In many real life problems of predictive modelling, the output (i.e., the target property) is structured, meaning that there can be dependencies between classes (e.g., classes are organized into a tree-shaped hierarchy or a directed acyclic graph) or some internal relations between the classes (e.g., sequences). These types of problems occur in domains such as life sciences (predicting the gene function, selecting the most important genes for a given disease, detecting toxic molecules, etc), ecology (analysis of remotely sensed data, habitat modelling), multimedia (annotation and retrieval of images and videos) and the

semantic web (categorization and analysis of text and web). Having in mind the needs of the application domains and the increasing quantities of structured data, Yang and Wu (2006) listed the task of "mining complex knowledge from complex data" as one of the ten most challenging problems in machine learning.

A variety of methods, specialized for predicting a given type of structured output (e.g., a hierarchy of classes (Silla and Freitas, 2011)), have been proposed (Bakır *et al.*, 2007). However, many of them are computationally demanding and not suited for dealing with large datasets (especially large outputs). The predictive clustering framework offers a unifying approach for dealing with different types of structured outputs and the algorithms developed in this framework construct the predictive models very efficiently. Moreover, predictive clustering trees and predictive clustering rules can be easily interpreted by a domain expert, thus supporting the process of knowledge extraction.

To further increase the predictive performance of a single predictive model, one can construct an ensemble of predictive models. For the simple classification and regression tasks, it is widely accepted that an ensemble of predictive models lifts the predictive performance of its base predictive models (Seni and Elder, 2010). However, for the task of predicting structured outputs, whether we use the predictive clustering framework or other global predictive models, this is not obvious.

In the case when the base predictive models 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 trees for classification of a single target variable. Having in mind these two conflicting influences, it is not obvious whether an ensemble of predictive clustering trees can significantly increase the predictive performance over that of a single predictive clustering tree.

Global predictive models exploit the dependencies between the components of the structured outputs and, as a result, have better predictive performance than local predictive models. However, in an ensemble learning setting, it is not clear if the predictive performance of an ensemble of global predictive models will be better or worse than the predictive performance of ensembles of local predictive models when an ensemble is constructed for each component of the structured output. It is also not clear which method will be more efficient, both in terms of running (execution) time and size of the predictive models.

Another open issue in ensemble learning is the question of how many predictive models are enough to get an optimal performance. Bauer and Kohavi (1999) observe ensembles of up to 30 predictive models and show that the biggest improvement in terms of

predictive performance is achieved after adding the first 10-15 predictive models. After that, the error rate gradually reaches a plateau. They suggest 25 predictive models as a reasonable compromise between the predictive performance and the efficiency (running time and model complexity) of an ensemble. On the other hand, Banfield *et al.* (2007) investigate ensembles of 1000 predictive models and propose an algorithm that uses the stabilization of the error rates to decide when the ensemble learning should stop. Hence, this approach further adds to the computational complexity of 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 predictive models are enough for optimal performance of an ensemble of global predictive models.

1.3 Contributions

We propose ensembles of PCTs for predicting structured outputs to address the issues raised above. The methodology presented in this thesis was partially published in several conference and journal publications, such as (Dimitrovski *et al.*, 2011; Kocev *et al.*, 2007a,b, 2008, 2009, 2010; Schietgat *et al.*, 2010). The complete list of papers related to this thesis is given in *Appendix C*. The main contributions of the work presented in this thesis are summarized as follows:

- We propose ensemble learning methods for predicting structured outputs that use PCTs as base predictive models. To the best of our knowledge, this is the first approach that uses ensembles of global predictive models. Furthermore, the proposed methods are general in terms of the type of the structured output: they support the tasks of predicting multiple continuous targets, predicting multiple discrete targets, and hierarchical multi-label classification.

- We perform extensive empirical evaluation of the proposed methods over a variety of benchmark datasets. We construct ensembles of up to 1000 predictive models and select ensembles of 50 global predictive models as optimal in terms of predictive performance and efficiency, in terms of training time and size of the trees in the ensembles.

- We compare the performance of ensembles of global predictive models and single global predictive models, as well as ensembles of local predictive models. Both, global and local ensembles lift the predictive performance of a single global or local predictive model. Next, the predictive performance of global ensembles is not statistically significantly different from the performance of local ensembles. However,

global ensembles are more efficient than local ensembles, yielding smaller training times and smaller models. Moreover, global ensembles require fewer trees than local ensemblesto reach the optimal performance.

- We apply the ensembles for predicting structured outputs in three practically relevant domains: modelling vegetation condition, image annotation and prediction of gene function. Each application provides a contribution to the respective domain.
  - We construct PCTs and ensembles of PCTs using remotely acquired data in conjunction with adequate field data (Kocev *et al.*, 2009). The PCTs offer further insights into the resilience of some remnant indigenous vegetation types and the relative importance of biophysical and landscape attributes that influence their condition. The ensembles of PCTs were used to generate maps of the condition of the indigenous vegetation, which can be used to support biodiversity planning, management and investment decisions.
  - We apply ensembles of PCTs for HMC to two benchmark tasks for hierarchical annotation of medical (X-ray) images and an additional task for photo annotation (Dimitrovski *et al.*, 2011). Ensembles of PCTs for HMC have better predictive performance and are more efficient than a collection of SVMs, on all tasks. Furthermore, ensembles of PCTs applied to the two medical image datasets provide the best annotation results reported in the literature so far.
  - We use bagging of PCTs for HMC to predict gene functions for three model organisms (Schietgat *et al.*, 2010). Bagging of PCTs for HMC has competitive predictive performance with other state-of-the-art approaches for gene function prediction. Only the approaches that make use of additional information about the gene interaction network show better predictive performance.
- We propose a method for feature ranking for structured outputs based on random forests (Kocev *et al.*, 2008). It produces a single feature ranking for the structured output, while taking into account the dependencies and the relations that exist between the components of the structured output. Applied to a case study for biomarker discovery, we show that feature ranking for multiple targets offers some advantages over the rankings for single targets.
- We suggest a novel ensemble learning method based on a beam search technique (Kocev *et al.*, 2007a). This ensemble learning method controls directly the diversity in the ensemble, thus allowing to further investigate the trade-off between diversity and predictive performance of an ensemble. In addition, we get an interpretable

ensemble by selecting the top-ranked tree from the ensemble as a representative (since the beam keeps the trees ranked by a heuristic score).

1.4 Organization

This introductory chapter presents the general perspective and context of the thesis. It also provides the motivation for performing the research and lists the main original scientific contributions. The rest of the thesis is organized as follows.

In Chapter 2, we first give the broader context of the thesis and present the background by defining the machine learning tasks we consider, such as predicting multiple targets and hierarchical multi-label classification. Next, we give a brief overview of the research related to the methods proposed in the thesis, i.e., ensemble methods, predictive clustering and the task of predicting structured outputs.

Chapter 3 describes PCTs and their extensions for the considered tasks. We first describe the PCTs as global predictive models, i.e., PCTs for predicting the complete structured output. Next, we present the use of PCTs in the context of local predictive models, i.e., an architecture of PCTs for local prediction of the structured output.

Chapter 4 describes the ensemble methods for predicting structured outputs, which are the main contribution of this thesis. We describe four ensemble methods for global prediction of structured outputs that use predictive clustering trees as base predictive models. In addition, we present the respective extensions of the combination schemes for the base predictive models. Next, we present an architecture of ensembles for local prediction of structured outputs. Finally, we analyze the computational complexity of the proposed methods.

Chapter 5 presents the experimental evaluation of the proposed methods. We first present the experimental questions that we address, the benchmark domains, the evaluation measures and the parameter instantiations for the learning methods. Next, we present and discuss the results for each considered task separately.

Chapter 6 presents three case studies which illustrate the application of the developed methods on practically relevant domains. The domains include predicting vegetation condition, annotation of medical images and gene function prediction. Each of the case studies was published as a journal publication, hence we present them in this thesis as they appeared in the respective journal.

Chapter 7 presents further extensions of the proposed methods. First, it presents how the developed methods can be extended for additional types of structured outputs and the use of additional distance measures. Next, it presents a feature ranking method for

structured outputs and its evaluation on a case study for biomarker discovery. Finally, it describes a new ensemble learning method that uses a beam search strategy to construct the base predictive models.

Finally, Chapter 8 presents the conclusions. It presents a summary of the thesis, its original contributions and directions for further work.

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 three paradigms that are the basis for the methods presented in this thesis: ensemble learning, predictive clustering and predicting structured outputs.

2.1 Machine learning tasks considered

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

2.1.1 The task of predicting multiple targets

The task of predicting multiple targets 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 therefore 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 previously classified instances and applying the obtained model to a set of previously unseen examples (Breiman *et al.*, 1984; Langley, 1996). The unseen examples are classified into a single class from a set of possible classes.

	Descriptive variables						Target variables												
Sample ID	Temperature	$K_2Cr_2O_7$	NO_2	Cl	CO_2	...	*Cladophora sp.*	*Gongrosira incrustans*	*Oedogonium sp.*	*Stigeoclonium tenue*	*Melosira varians*	*Nitzschia palea*	*Audouinella chalybea*	*Erpobdella octoculata*	*Gammarus fossarum*	*Baetis rhodani*	*Hydropsyche sp.*	*Rhyacophila sp.*	*Simulim sp.*	*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 water samples, while the target variables are the abundances of 14 bioindicator organisms.

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, 2011; 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
…	…	…	…	…	…	…	…

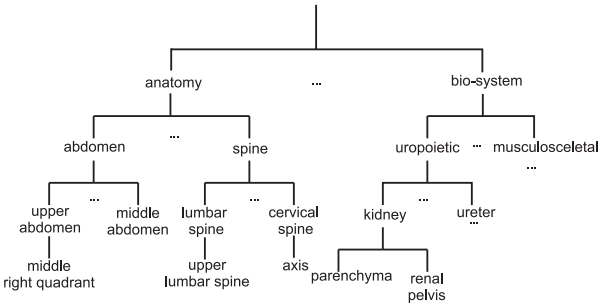


Figure 2.3: An example of a dataset for hierarchical multi-label classification of medical X-ray images (Dimitrovski *et al.*, 2008).

2.2 Related work

In this section, we 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 methods for predicting structured outputs.

2.2.1 Ensemble learning

Ensemble methods are machine learning techniques that generate a set of predictive models and combine their predictions into a single prediction (Dietterich, 2000a; Džeroski *et al.*, 2009; Kuncheva, 2004; Valentini, 2003). Each of the constituent predictive models is called a *base predictive model* and the set of predictive models is called an *ensemble*. The notion of ensemble is general and also applies to 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).

There is a distinction between an ensemble and an architecture of predictive models. An ensemble of predictive models directly combines the outputs of each base predictive model to obtain the overall prediction. An architecture of predictive models is a set of predictive models, whose outputs are not directly combined to obtain the overall prediction, but rather the output of one predictive model can be used in the training of some of the next predictive models (Ianakiev and Govindaraju, 2000). More to the point, this output can be copied as a part of the overall prediction. In other words, each predictive model in an ensemble operates in the same (input/)output space, while the predictive models in an architecture operate on different (input/)output spaces.

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 focus on how ensembles are constructed, how the base predictive models are combined to obtain a single prediction, and why do ensembles have good predictive performance.

Ensemble creation techniques

An ensemble is a set of predictive models. We present the three most widely used approaches to ensemble learning (i.e., constructing the different base predictive models): (1) use of heterogeneous predictive models; (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 of the most representative methods.

In the first approach, the ensemble is constructed by learning heterogeneous predictive models (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 predictive models into a single prediction. However, the most prominent ensemble learning

Table 2.1: Summarized approaches to ensemble learning.

Method	Use of heterogeneous models	Manipulate the data instances	Manipulate the data features	Manipulate the learn. algorithm
Stacking (Wolpert, 1992)	√			
Bagging (Breiman, 1996a)		√		
Random forests (Breiman, 2001a)		√	√	√
Bootstrap ensemble with noise (Raviv and Intrator, 1996)		√		
Boosting (Freund and Schapire, 1996)		√		
Importance sampled ensembles (Friedman and Popescu, 2003)		√		
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)				√

method that employs this technique is *stacking* (Wolpert, 1992). Stacking combines the predictive models not by a fixed voting scheme, but by learning an additional *meta* predictive model that uses as input the predictions of the base predictive models. The performance of stacking highly depends on the attributes that are used in the dataset for constructing the *meta* predictive model and the selection of the learning algorithm for the *meta* predictive model.

In the second approach, the base predictive models are constructed by manipulating the training set. This approach is typically used in combination with unstable predictive models. An unstable predictive model is the one which suffers great changes in its structure with small changes in the training set. A typical example of such a predictive model is the decision tree (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*, *boosting* or *importance sampling*. Bootstrapping creates several bootstrap replicates of the training dataset by random selection with replacement (Berthold and Hand, 2003). A predictive model is then learned using each of the bootstrap replicates. The most prominent ensemble learning method that uses bootstrapping is *Bagging* (Breiman, 1996a). *Bagging* can use any type of predictive model as a base predictive model, but, most often it uses decision trees.

Raviv and Intrator (1996) construct ensembles of neural networks using bootstrap replicates of the training set. Additionally, 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 predictive model, thus creating a chain of predictive models. If an instance was correctly classified/predicted, then its weight is decreased when training the next predictive model or if an instance was miss-classified, then its weight is increased when training the next predictive model. The training set with the re-weighted instances is used to train the next predictive model. This procedure ensures that the different predictive models are focused on different areas of the instance space. The procedure iterates until the predictive performance of the ensemble or the number of trained predictive models reaches some user defined threshold.

Importance sampling (Friedman and Popescu, 2003, 2008) uses stage-wise additive modelling to construct the base predictive models. The training samples for the base predictive models are obtained by random selection without replacement. It further uses a importance sampling strategy to select a set of points for parameter evaluation and weights for combination of the predictions from the base models. The *importance sampled learned ensembles* is a unifying framework for ensemble learning. Several ensemble learning methods, such as *Bagging*, *Random forests* and *Boosting*, can be represented using the formalism of this framework.

The manipulation of the feature space can be done by random selection of feature subspaces from the original feature space. Each of the base predictive models 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 affect 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 predictive models using both bootstrap replicates of the training set and feature sub-spaces. This method can use any type of predictive model as a base predictive model. *Random forests* are the most famous ensemble learning method that uses only decision trees as base predictive models. 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 an 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 predictive models by changing the learning algorithm (e.g., some of its parameters) for each base predictive model. 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 predictive model is a neural network obtained with different initial parameters. Another group of ensemble methods that use trees and rules as base predictive models 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 predictive models. 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 predictive models into a single prediction (Kittler *et al.*, 1998; Kuncheva, 2004). There are generally two approaches used to obtain a single prediction from an ensemble: model selection and model fusion/combination (Džeroski *et al.*, 2009).

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

The model fusion/combination approach combines the predictions of all base predictive models into an overall prediction of the ensemble. *Stacking* can be viewed as a model fusion approach: it uses the predictions of the base predictive models to train a *meta* predictive model, and is then used to combine the predictions of the base predictive models to produce the overall prediction from of the ensemble. However, by far the most common method for model 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) or using some complex procedure (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 predictive models and calculates their mean value. This mean value is then used as the prediction of the ensemble. Weights can also be used for the predictions of the base models. The weights (as for classification) can be related to the performance of the models (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 and decision templates.

Why do ensembles perform well?

A necessary condition for an ensemble to perform better than any of its base predictive models is that the base predictive models are accurate and diverse (Hansen and Salamon, 1990; Hastie and Tibshirani, 1990). An accurate predictive model makes smaller error on unseen instances than random guessing. Diverse predictive models make different errors on unseen instances (i.e., the errors of the predictive models are independent). These conditions were regarded as a sufficient requirement for an effective ensemble. However, Kuncheva and Whitaker (2003) have shown that this is not always the case. The predictive models producing independent errors do not always outperform the ones that produce dependent errors. Actually, there exists a trade-off between the accuracy and the independence of the base predictive models. Dietterich (2000a); Džeroski *et al.* (2009); Valentini (2003) offer several fundamental reasons and theoretical analyses as to why ensembles of predictive models 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 of the true model.

Finally, 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 process 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. A 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 inspected 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 (see the references above), show that ensembles often outperform their base predictive models and offer high predictive performance. Since the ensembles are a (large) set of predictive models they are more difficult to inspect and interpret than individual predictive models. This is further complicated by the fact that the predictions of the individual predictive models are combined. Thus, they do not offer additional insight into 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) (Van Assche, 2008) or by performing feature ranking by using the ensembles' mechanism (Breiman, 2001a). We briefly discuss these two approaches below.

A meta model representing an ensemble can be constructed as a complex model while building the ensemble or by learning a new model based on the ensemble. The first approach constructs a complex understandable model (such as an alternating decision tree,

a consolidated tree or an orthogonal decision tree) that has some ensemble characteristics: 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 using these data to learn a representative model (Craven, 1996; Domingos, 1998; Van Assche, 2008).

Breiman (2001a) proposed to further exploit the random forests (or bagging) mechanism and rank the descriptive variables (i.e., provide a feature ranking). To calculate the importance of each descriptive variable for the class, this approach performs random permutations of the variable's values and calculates out-of-bag error estimates on the randomized data. 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 a 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 are usually of higher dimensionality.

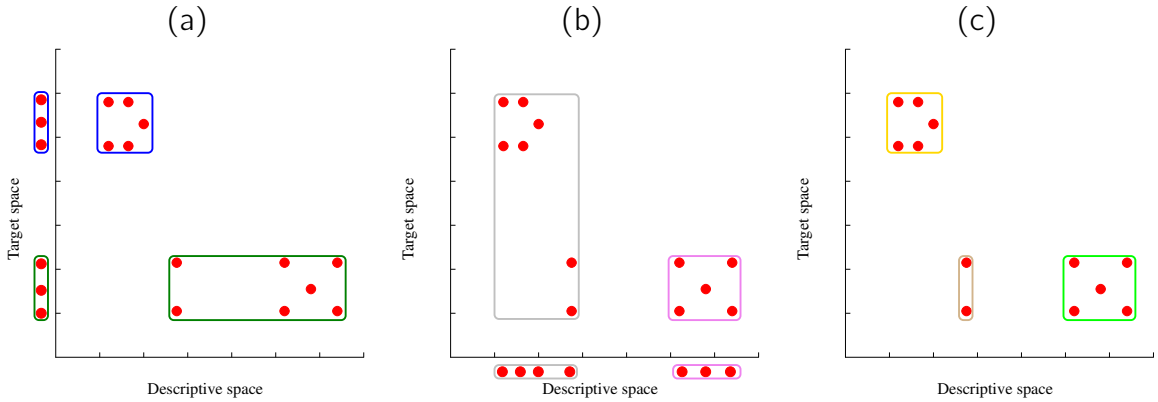


Figure 2.4: An illustration of predictive clustering: (a) clustering in the target space, (b) clustering in the descriptive space, and (c) clustering in both the target and the descriptive space. Figure adapted from Blockeel (1998) and Ženko (2007).

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 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, prediction of the model is the projection of the cluster prototype on the target space. The prototype is an instance representative of the cluster and it is the 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).

For the different prediction tasks, 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 and semantic similarity measure can also be 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 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 frequency of each class and then applying a user defined threshold (see Section 3.2.2 for details). When predicting time series, the prototype is calculated as the mean and/or the medoid, where the medoid is taken with respect to the used distance measure. Both prototypes are reported when all

[1]The CLUS system is available for download at `http://www.cs.kuleuven.be/~dtai/clus` and developer version is available from `http://sourceforge.net/projects/clus/`.

time series in the dataset 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 improves the performance of individual predictive clustering trees. We also investigate whether an ensemble for predicting structured outputs outperforms the ensembles learned separately for each component of the target.

2.2.3 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, 2011). 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 the model's outputs to obtain a structured prediction. Traditional 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 predictive models for each component of the output.

The global methods, on the other hand, construct only a single predictive model that predicts the complete structured output at once, known as 'big-bang' method (Silla and Freitas, 2011). The main advantage of the global methods 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 (e.g., 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. In this thesis, we consider the last task. The methods we propose can handle two types of outputs: multiple discrete variables (multi-target classification) and multiple continuous variables (multi-target regression), while most of the methods from the literature can handle only one type of output.

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 predictive models over a class of tasks.

We present and categorize the related work along four dimensions: statistics, statistical learning theory, Bayesian theory, kernel learning and decision tree learning. 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 methods use hierarchical Bayesian models.

Next, there are many methods 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 SVMs 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.

Finally, several methods for predicting multiple target variables based on decision trees were proposed. These methods focused on predicting multiple discrete target variables (Kim and Lee, 1998; Zhang, 1998) or predicting multiple continuous target variables (Segal, 1992; Xiao and Segal, 2009) or both (Lee *et al.*, 2006). Zhang (1998) proposed a method that constructs decision trees for multiple binary target variables and evaluates the method on a database for building-related occupant complaint syndrome. Next, Kim and Lee (1998) evaluated different splitting criteria for the nodes in the multi-target tree, such as extensions of Entropy and Gini index, weighted sum of node homogeneities and minimization of the expected loss.

Segal (1992) introduced the decision trees for multiple continuous target variables in a longitudinal setting (i.e., the same target variable measured in subsequent time points) using Mahalanobis distance to measure the within-node homogeneity. Moreover, Xiao and Segal (2009) used this method as a base predictive model for constructing random forests. They were primarily used to assess the variable importance on a dataset for identification of yeast regulatory network. Larsen and Speckman (2004) suggested a variation of the method of Segal (1992) and applied it for multi-target regression. Finally, Lee *et al.* (2006) proposed classification and regression tree based methods for multiple correlated target variables by fitting a generalized estimating equation in each node and selecting the split using the equation's residuals. This approach was evaluated on epileptic seizure data and graphic visualization user surveys.

Methods for hierarchical multi-label classification

A number of methods have been proposed for the task of hierarchical multi-label classification (Bakır *et al.*, 2007). Silla and Freitas (2011) survey and categorize the HMC methods based on some characteristics and the application domains. The characteristics of the methods 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 method 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 methods are functional genomics (biology), image classification, text categorization, and genre classification.

Here, we present and group some existing methods 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 methods 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 multiple networks derived from different genomic and proteomic data sources. Since the network based methods are based on label propagation, a number of methods 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 SVMs generated for each class separately. They compute diffusion kernels and use them in kernel logistic regression. Obozinski *et al.* (2008) present a two-step method 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. The method 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 method that does not require a second step to make sure that the hierarchy constraint is satisfied. Their method 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). Furthermore, Gärtner and Vembu (2009) propose to use counting of

super-structures from the output to efficiently calculate (in polynomial time) the argmax of the discriminant function.

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 method 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, a method 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 HSC decision tree method in the context of tree shaped class hierarchies. Vens *et al.* (2008) extend the algorithm towards hierarchies structured as directed acyclic graph 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 Predictive clustering trees for structured outputs

In this chapter, we present two tree based methods for predicting structured outputs. We first present the predictive clustering trees (PCTs) that predict the complete structured output - global predictive models. In particular, we present the instantiations of the PCTs for predicting multiple continuous variables, predicting multiple discrete variables, and hierarchical multi-label classification. We then present a method for local prediction of structured outputs with an architecture of PCTs - local predictive models.

3.1 Basic algorithm for learning PCTs

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 of classes. 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). By maximizing the variance reduction the cluster homogeneity is maximized and it improves the predictive performance. If no acceptable test can be found (see line 6), that is, if the test does not significantly reduces the variance,

then the algorithm creates a leaf and computes the prototype of the instances belonging to that leaf.

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** $E_i \in \mathcal{P}^*$ **do**
4: $tree_i = \text{PCT}(E_i)$
5: **return** node(t^*, $\bigcup_i \{tree_i\}$)
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: $h = Var(E) - \sum_{E_i \in \mathcal{P}} \frac{|E_i|}{|E|} Var(E_i)$
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 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 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.

3.2 Global prediction of structured outputs with PCTs

In this section, we present three instantiations of the algorithm for PCTs. We first present the instantiation of PCTs for the task of predicting multiple target variables. Next, we present the instantiation of PCTs for HMC.

3.2.1 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 leaves store the predictions (in this case, a tuple 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 tuple with the mean values of the target variables, calculated by using the training instances that belong to the given leaf.

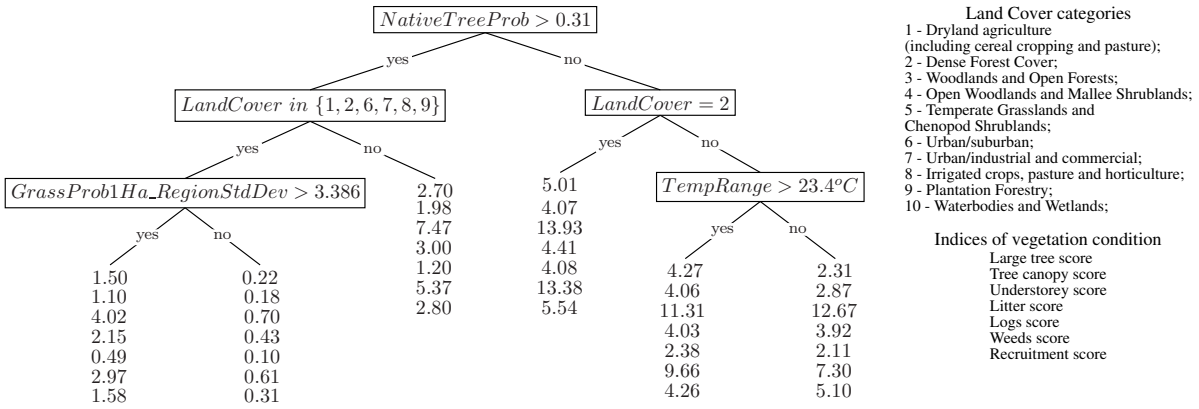


Figure 3.1: Example of a predictive clustering tree for predicting multiple continuous targets (Kocev *et al.*, 2009). Each leaf stores a prediction for the values of a set of indices of the state of indigenous vegetation in Victoria, Australia. The tree was learned using the data shown in Figure 2.1.

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 leaves 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)). Note that in the single target case, $Var(E) = Entropy(E)$ corresponds to information gain.

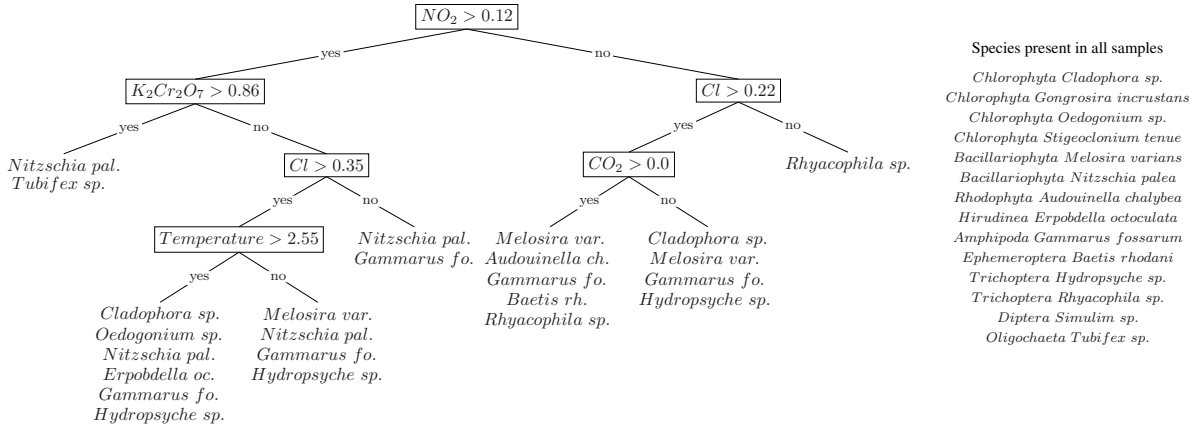


Figure 3.2: Example of a predictive clustering tree for predicting multiple discrete targets. Each leaf stores a prediction for the presence or absence of each bioindicator species. The tree was learned using the data shown in Figure 2.2.

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, gain ratio and the *m*-estimate.

3.2.2 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 classes are organized in a form of 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 (2011) give a detailed overview of the possible application areas and the available approaches to HMC.

Silla and Freitas (2011) 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 hierarchies of classes organized both as a tree or a directed acyclic graph.
- 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 leaves store the predictions/annotations (in this case, classes organized into a tree-shaped hierarchy called IRMA coding scheme (Lehmann *et al.*, 2003)).

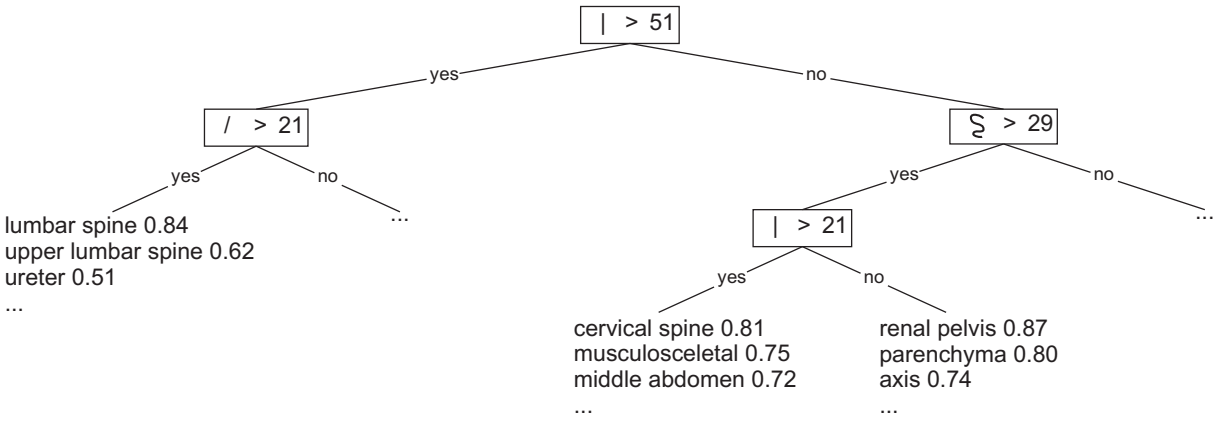


Figure 3.3: Example of a predictive clustering tree for hierarchical multi-label classification. Each leaf stores a prediction for the annotations of a given medical X-ray image from the IRMA coding scheme. The tree was learned using the data shown in Figure 2.3.

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_i) and the set's mean class vector ($\overline{L}$), i.e.,

$$Var(E) = \frac{1}{|E|} \cdot \sum_{E_i \in E} d(L_i, \overline{L})^2.$$

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:

$$d(L_1, L_2) = \sqrt{\sum_{l=1}^{|L|} w(c_l) \cdot (L_{1,l} - L_{2,l})^2},$$

where $L_{i,l}$ is the l'th component of the class vector L_i of an instance E_i, $|L|$ is the size of the class vector, 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:

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

Note that our definition of $w(c)$ allows the classes to be structured in a form of a directed acyclic graph. Figure 3.4(c) depicts an example of a directed acyclic graph structured hierarchy. In a general case, a directed acyclic graph-like 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 shaped as a directed acyclic graph, then the depth of a class is not unique: classes do not have a single path from the top-node (for example see

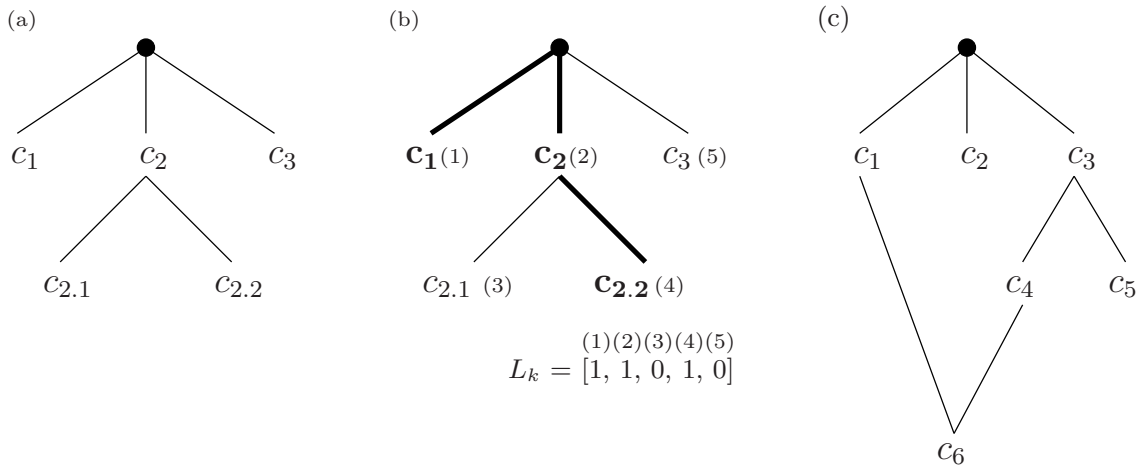


Figure 3.4: Toy examples of hierarchies structured as a tree and a directed acyclic graph.(a) Class label names contain information about the position in the hierarchy, e.g., $c_{2.1}$ is a subclass of c_2. (b) The set of classes $\{c_1, c_2, c_{2.2}\}$, shown in bold in the hierarchy, represented as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

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 τ 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.

3.3 Local prediction of structured outputs with PCTs

The presented structured output learning algorithms (CLUS-MTDT and CLUS-HMC) belong to the group of methods known as 'big-bang' or global predictive models (Bakır *et al.*, 2007; Silla and Freitas, 2011). 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, on the other hand, use 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 different approaches that can be used (Silla and Freitas, 2011): 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 the 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. A binary classifier is then learned using the labeled training set. This procedure is repeated for each leaf node from the hierarchy. In this approach, the classifiers are not aware of the hierarchical dependencies that exist between the classes and they are incapable of making predictions for the non-leaf nodes of the hierarchy.

The second approach, local classifiers per level, constructs a classifier for each level of the hierarchy. This approach also requires post-processing to solve the class-membership inconsistencies that may appear. It has been used only as a baseline in comparisons by 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 selecting 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 the one vs. 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 collection of datasets from functional genomics. The conclusion of the study was that the last approach (called hierarchical single-label classification - HSC) 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.

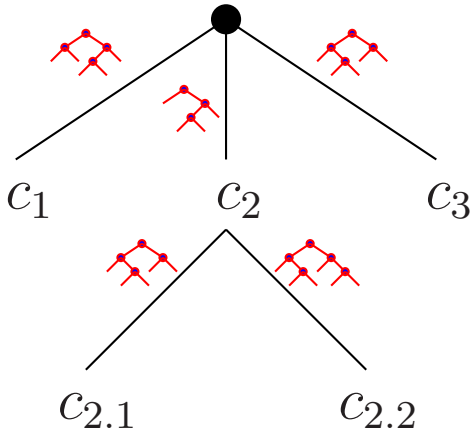


Figure 3.5: An illustration of the hierarchical single-label classification approach used by Vens *et al.* (2008). The local classifiers at each branch from the hierarchy are decision trees.

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 the case of a directed acyclic graph), starting from the tree for the 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.

4 Ensembles for predicting structured outputs

In this chapter, we present several ensemble methods for predicting structured outputs, which are the main contribution of this thesis. We first present how ensembles of predictive clustering trees (PCTs) for global prediction of the structured outputs are constructed. In particular, we present four ensemble learning methods: bagging, random forests, random subspaces and bagging of subspaces. We then present ensembles used as local predictive models for prediction of components of the structured output. Finally, we present the computational complexity of the proposed methods.

4.1 Global prediction of structured outputs with ensembles of PCTs

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 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 bagging and random forests methods that are often used in the context of decision trees. We have adapted these methods to use 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 4.1. For the random forests method (top-right in Table 4.1), 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 methods, 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.

4.1.1 Bagging

Bagging (Breiman, 1996a) is an ensemble method that constructs the different predictive models by making bootstrap replicates of the training set and using each of these replicates to construct a predictive model (Table 4.1, 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) showed 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.

4.1.2 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 set of descriptive attributes during learning (Table 4.1, top-right). More precisely, at each node in the decision trees, a random subset of the descriptive attributes is taken, and the best attribute 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., $f(D) = 1$, $f(D) = \lfloor\sqrt{D} + 1\rfloor$, $f(D) = \lfloor\log_2(D) + 1\rfloor \ldots$). By setting $f(D) = D$, we obtain the bagging procedure. The algorithm for learning a random forest using PCTs as base predictive models is presented in Table 4.1, top-right.

4.1.3 Random subspaces

The random subspaces method (Ho, 1998) creates an ensemble by learning each of the base models on different feature subspaces (Table 4.1, bottom-left). This method first

Table 4.1: 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: $\quad E_i = bootstrap(E)$
4: $\quad T_i = PCT(E_i)$
5: $\quad F = F \bigcup \{T_i\}$
6: **return** F

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

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

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

selects a subset of the descriptive attributes and then learns a base model using the projection of the dataset on these attributes. The number of descriptive attributes that are used to learn the base models is given with a function f of the total number of descriptive attributes D (as for random forests). 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.

4.1.4 Bagging of subspaces

The bagging of subspaces method (Panov and Džeroski, 2007) combines bagging and random subspaces. This method is outlined in Table 4.1, 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 (in a manner similar to bagging) and then by random selection of a subset of the descriptive attributes (in a manner similar to 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 (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, etc) as a base model.

4.1.5 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 PCTs 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 the average value per target of the predictions of the base predictive models. For the datasets for hierarchical classification we also use the average of the predictions and apply the thresholding described in Section 3.2.2. 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.

4.1.6 Implementation issues

In this section, we discuss two issues related to the implementation of the proposed ensemble methods for predicting structured outputs: out-of-bag error estimation and memory optimization.

In this thesis, we estimate predictive performance by using 10-fold cross-validation (Section 5.1.4). However, out-of-bag error is also a good estimate of the generalization error of an ensemble learning algorithm (Breiman, 1996a). Moreover, one can use out-of-bag error estimation when the datasets are small, because the calculation of out-of-bag error does not require a splitting of the training set, as for cross validation, or the creation

of a separate holdout set. In addition, the out-of-bag estimation of the error can be used to obtain a feature ranking (see Section 7.2 for details).

The next implementation issue is the memory usage of the ensemble construction algorithms. Typically, one first constructs all base predictive models. A prediction for a test example is then produced by running the example through each of the base predictive models and combining the obtained predictions. This means that all base predictive models need to be stored. In problems with a large number of examples and descriptive attributes, the base predictive models can grow large and thus require a lot of memory. Moreover, the ensembles of such base predictive models will require even more memory.

We optimize the ensemble construction algorithm so it does not require too much memory. We do not store the base predictive models themselves, but rather store the predictions they make on the training, testing and out-of-bag set. The stored predictions are calculated incrementally: the predictions of the second base predictive model are combined with the predictions of the first base predictive model, and the predictions of the third base predictive model are combined with the joint prediction of the two base predictive models from the previous step, etc. With this optimization, the construction algorithm will require the same amount of memory regardless of the number of trees in the ensemble.

4.2 Local prediction of structured outputs with ensembles of PCTs

The presented ensembles methods for predicting structured outputs belong to the group of global predictive models (Bakır *et al.*, 2007; Silla and Freitas, 2011). They make a single prediction for the complete structured output, i.e., simultaneously predict all of its components. In Section 3.3 we presented a method that predicts components of the structured output (local predictive models), i.e., an architecture of PCTs for local prediction. Here, we extend this approach and, instead of PCTs, we use an ensemble of PCTs as local predictive model, i.e., an architecture of ensembles of PCTs for local prediction.

The local predictive models for the task of predicting multiple targets are constructed by learning an ensembles of predictive models for each of the targets separately. In the task of hierarchical multi-label classification (CLUS-HMC), we construct an ensemble of PCTs for each hierarchy edge (Figure 4.1).

In this thesis, we extend the approach of Vens *et al.* (2008) by applying ensembles as local classifiers, instead of single decision trees. The CLUS-HSC algorithm can be applied

to ensemble learning in two ways by constructing: an ensemble of architectures or an architecture of ensembles. The first approach creates the ensemble by creating multiple architectures (as shown in Figure 3.5). These multiple architectures can be created on different bootstrap replicates, on different feature spaces, by 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 a classifier at each branch (depicted in Figure 4.1). We prefer here the second approach since it is closer to the learning of local classifiers for predicting multiple target variables.

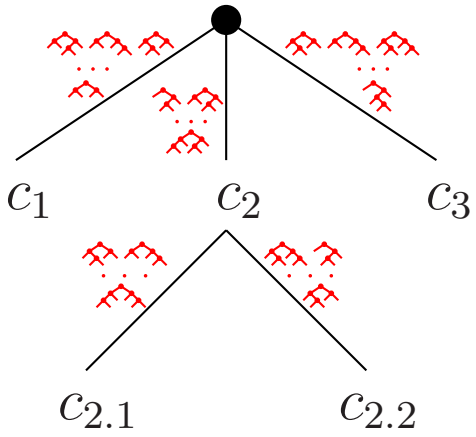


Figure 4.1: Local classification with an architecture of ensembles. The local classifiers at each branch from the hierarchy are ensembles of PCTs.

The method presented in Figure 4.1 constructs an ensembles of PCTs for each edge (connecting a class c with a parent class $par(c)$) in the hierarchy, thus creating an architecture of ensembles (hierarchical single-label classification - HSC). The corresponding ensemble predicts membership to class c, using the instances that belong to $par(c)$. The construction of this type of ensemble 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.

Similarly as for the PCTs for local prediction, the HSC ensemble predicts the conditional probability $P(c|par(c))$. A new instance is predicted by recursive application of the product rule $P(c) = \min_j P(c \mid par_j(c)) \cdot P(par_j(c))$ (with $par_j(c)$ denoting the j-th parent of c in the case of a DAG), starting from the ensemble for the 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 Chapter 5, 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), following the architecture-of-ensembles approach from Figure 4.1.

4.3 Computational complexity

In this section, we analyze and discuss the computational complexity aspects of the proposed methods. We first analyze the computational complexity of PCTs for local prediction of structured outputs. Next, we derive the computational complexity of ensembles for local prediction of structured outputs. We then analyze the computational complexity of PCTs for global prediction of structured outputs and ensembles of PCTs for global prediction of structured outputs. Finally, we summarize, discuss and compare the analyses of the computational complexity of all proposed methods.

4.3.1 PCTs for local prediction

In the analyses of the computational complexity of the proposed methods, we follow the notation introduced in Chapters 2 and 3. We assume that the training set contains N instances and D descriptive attributes (from which M are continuous). For the task of predicting multiple targets, we assume that the output contains T targets, while for the task of CLUS-HMC, we assume that the output hierarchy contains L classes (i.e., nodes) and G edges. We first give the computational cost for constructing a single tree used for prediction of a single target variable. We then extend the analysis for local prediction of structured outputs.

The construction of a tree includes three phases that contribute to the computational complexity of the tree learning algorithm described in Chapter 3. The phases are executed at each node of the tree and they include: sorting the values of the numeric descriptive attributes, calculating the best split, and applying the split to the training instances.

Sorting a single continuous descriptive attribute costs $\mathcal{O}(N \log N)$, thus sorting all numeric descriptive attributes costs $\mathcal{O}(MN \log N)$. The cost of passing all instances and calculating the needed statistics for all descriptive attributes over a single target variable is $\mathcal{O}(DN)$. Once the best test is selected, splitting the instances into the respective nodes costs $\mathcal{O}(N)$.

To sum up, the computational complexity of creating a node in the tree is $\mathcal{O}(MN \log N) + \mathcal{O}(DN) + \mathcal{O}(N)$. Furthermore, if we take into consideration that $M = \mathcal{O}(D)$, then the

computational complexity of the process of creating a node in the tree is $\mathcal{O}(DN \log N) + \mathcal{O}(DN) + \mathcal{O}(N)$.

We assume, as in (Witten and Frank, 2005), that the tree is balanced and bushy. This means that the number of nodes is in the order of $logN$, i.e., $\mathcal{O}(\log N)$. Having this in mind, the total computational cost of tree construction is:

$$\mathcal{O}(DN \log^2 N) + \mathcal{O}(DN \log N) + \mathcal{O}(N \log N).$$

The upper bound of this cost is determined by the first term of the overall computational complexity, i.e., $\mathcal{O}(DN \log^2 N)$. Note that, for the construction of PCTs using CLUS, as for the construction of C4.5 decision trees (Quinlan, 1993), the continuous descriptive attributes are sorted at each node. However, there are tree induction algorithms that do not sort the continuous descriptive attributes at each node, such as SLIQ (Mehta *et al.*, 1996) and SPRINT (Shafer *et al.*, 1996), but rather keep the instances sorted at all time using more complex data structures.

We further extend this analysis in the context of predicting structured outputs. In the task of predicting multiple targets, a tree is constructed for each target separately, thus the computational complexity is T times higher than the computational complexity of a tree for a single target attribute. In the HMC task, a tree is constructed for each edge in the hierarchy (see Figure 3.5). This means, that the computational complexity of an architecture of decision trees is at most G times higher than the computational complexity of a single tree. However, the computational complexity of an architecture of decision trees also depends on the average number of classes per instance and the average number of leaf classes per instance. Smaller average numbers of classes and leaf classes per instance lead to smaller sets of training instances for the local trees and, consequently, smaller computational complexity. In typical applications, the computational complexity of this approach is smaller than G times the complexity of constructing a single tree.

4.3.2 Ensembles for local prediction

The computational complexity of constructing an ensemble depends of the computational complexity of constructing its base predictive models. In general, the computation complexity of an ensemble with k base predictive models is k times higher than the computational complexity of a single base predictive model. However, ensemble methods perform sampling of the instances and/or the features thus reducing the computational complexity of constructing a single base predictive model by a constant factor as compared to the construction of a base predictive model without selection of instances or features. In the following, we first discuss the ensemble methods of bagging, random forests, random

subspaces and bagging of subspaces, one by one, for predicting a single target. Next, we extend this analysis in the context of predicting structured outputs.

The overall computational complexity of an ensemble method can be calculated as $k(\mathcal{O}(DN\log^2 N) + \mathcal{O}(DN\log N) + \mathcal{O}(N\log N))$. Bagging uses bootstrap replicates of the training instances to construct the base predictive models. The creation of the bootstrap replicates has a computational complexity of $\mathcal{O}(N)$, which is added to the complexity of constructing a single base predictive model. However, the bootstrap replicates keep 62.3%, i.e., $1 - e^{-1}$, of the training instances (Breiman, 1996a), the number of instances used to train the base predictive models is not N, but $N' = 0.632 \cdot N$. This means that the computational complexity of constructing a single tree in bagging is at least 1.58 times smaller than the computational complexity of building a stand-alone tree.

The random forests and bagging of subspaces methods, in addition to the bootstrap sampling of the instances, also perform random sampling of the features. Having in mind that the selection of the features is uniform (i.e., the probability that a given feature will be selected is equal for all features), the computational complexity of constructing one base predictive model also depends of the selection function $f(D)$. In these ensemble methods, the number of descriptive attributes is $D' = f(D)$, where $f(D)$ is typically selected to be a logarithmic function of the number of attributes $f(D) = \lfloor log_2(D) + 1 \rfloor$. The overall computational complexity in this case is $k(\mathcal{O}(D'N'\log^2 N') + \mathcal{O}(D'N'\log N') + \mathcal{O}(N'\log N') + \mathcal{O}(N))$.

The random subspaces method uses different feature subsets to construct the base predictive models, without performing selection of the training instances. As for random forests and bagging of subspaces, the number of continuous descriptive attributes is $M' = f(M)$. Considering this, the overall computational complexity of this ensemble method is $k(\mathcal{O}(D'N\log^2 N) + \mathcal{O}(D'N\log N) + \mathcal{O}(N\log N))$.

We next extend the analysis of the computational complexity of ensemble methods for predicting a single target variable in the context of predicting structured outputs. The computational complexity of the ensemble methods for the task of predicting multiple targets is T times bigger than the complexity of an ensemble for predicting a single target. For the task of HMC, the computational complexity of the ensembles for local prediction of the structured output is G times higher than for the construction of an ensemble for predicting a single target. In this setting, as for the case of constructing a single tree, it holds that for real applications the computational complexity rarely reaches G times the complexity of an ensemble for predicting a single target.

4.3.3 PCTs for global prediction

The derivation of the computational complexity of constructing PCTs for global prediction of structured outputs follows the same pattern as for the PCTs for local prediction. The difference here is in the procedure for calculating the best split at a given node. This procedure, instead of a computational complexity $\mathcal{O}(DN)$, for the task of predicting multiple targets has a computational complexity of $\mathcal{O}(DNT)$. The computational complexity for the construction of the complete tree is as follows:

$$\mathcal{O}(DN \log^2 N) + \mathcal{O}(TDN \log N) + \mathcal{O}(N \log N).$$

The upper bound of the computational complexity is $\mathcal{O}(DN \log^2 N) + \mathcal{O}(TDN \log N)$. If $T > \log N$, then the upper bound is $\mathcal{O}(TDN \log N)$. If $T < \log N$, then the upper bound is $\mathcal{O}(DN \log^2 N)$.

The computational complexity of PCTs for HMC is similar to the one for predicting multiple targets. Considering the internal representation of the hierarchy of classes as a vector with length equal to the number of classes in the hierarchy (see Section 3.2.2), the computational cost to calculate the best split is $\mathcal{O}(DNL)$. Adding this cost to the costs of sorting and splitting the instances, the computational complexity of PCTs for HMC is:

$$\mathcal{O}(DN \log^2 N) + \mathcal{O}(LDN \log N) + \mathcal{O}(N \log N).$$

The upper bound of the computational complexity is $\mathcal{O}(DN \log^2 N) + \mathcal{O}(LDN \log N)$. Furthermore, if $L > \log N$, then the upper bound is $\mathcal{O}(LDN \log N)$. If $L < \log N$, then the upper bound is $\mathcal{O}(DN \log^2 N)$.

4.3.4 Ensembles for global prediction

The construction of an ensemble that consist of k PCTs for global prediction costs k times more than construction of a single PCT. As for the ensembles for local prediction, the ensemble methods reduce the computational cost with the selection of instances and/or features. The computational costs for each ensemble learning method is calculated in the same manner as for the ensembles for local prediction. Instead of a detailed description of the computational complexity, we present the overall computational complexities in Table 4.2. In the next section, we summarize the computational costs of all methods and compare them.

4.3.5 Summary and discussion

In Table 4.2, we summarize the computational complexities of the methods for predicting structured outputs that are based on predictive clustering trees. In the remainder of this section, we compare and discuss the computational complexities of the different methods. We focus the discussion on the comparison of local and global methods for predicting structured outputs.

Table 4.2: Computational complexity of the methods for predicting structured outputs based on predictive clustering trees. The notation is as follows: N - number of training instances, M - number of continuous descriptive attributes, T - number of target attributes, L - number of classes in a hierarchy, G - number of edges in a hierarchy, N' - number of unique instances in a bootstrap replicate of the training set, M' is the number of features considered when constructing the base predictive models, k - number of base predictive models.

Local	Multiple Targets	PCT	$T(\mathcal{O}(DN\log^2 N)+\mathcal{O}(DN\log N)+\mathcal{O}(N\log N))$
		Bagging	$Tk(\mathcal{O}(DN'\log^2 N')+\mathcal{O}(DN'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
		Rand. forests	$Tk(\mathcal{O}(D'N'\log^2 N')+\mathcal{O}(D'N'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
		Rand. subspaces	$Tk(\mathcal{O}(D'N\log^2 N)+\mathcal{O}(D'N\log N)+\mathcal{O}(N\log N))$
		Bag. subspaces	$Tk(\mathcal{O}(D'N'\log^2 N')+\mathcal{O}(D'N'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
	HSC	PCT	$G(\mathcal{O}(DN\log^2 N)+\mathcal{O}(DN\log N)+\mathcal{O}(N\log N))$
		Bagging	$Gk(\mathcal{O}(DN'\log^2 N')+\mathcal{O}(DN'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
		Rand. forests	$Gk(\mathcal{O}(D'N'\log^2 N')+\mathcal{O}(D'N'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
		Rand. subspaces	$Gk(\mathcal{O}(D'N\log^2 N)+\mathcal{O}(D'N\log N)+\mathcal{O}(N\log N))$
		Bag. subspaces	$Gk(\mathcal{O}(D'N'\log^2 N')+\mathcal{O}(D'N'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
Global	Multiple Targets	PCT	$\mathcal{O}(DN\log^2 N)+\mathcal{O}(TDN\log N)+\mathcal{O}(N\log N)$
		Bagging	$k(\mathcal{O}(DN'\log^2 N')+\mathcal{O}(TDN'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
		Rand. forests	$k(\mathcal{O}(D'N'\log^2 N')+\mathcal{O}(TD'N'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
		Rand. subspaces	$k(\mathcal{O}(D'N\log^2 N)+\mathcal{O}(TD'N\log N)+\mathcal{O}(N\log N))$
		Bag. subspaces	$k(\mathcal{O}(D'N'\log^2 N')+\mathcal{O}(TD'N'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
	HMC	PCT	$\mathcal{O}(DN\log^2 N)+\mathcal{O}(LDN\log N)+\mathcal{O}(N\log N)$
		Bagging	$k(\mathcal{O}(DN'\log^2 N')+\mathcal{O}(LDN'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
		Rand. forests	$k(\mathcal{O}(D'N'\log^2 N')+\mathcal{O}(LD'N'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$
		Rand. subspaces	$k(\mathcal{O}(D'N\log^2 N)+\mathcal{O}(LD'N\log N)+\mathcal{O}(N\log N))$
		Bag. subspaces	$k(\mathcal{O}(D'N'\log^2 N')+\mathcal{O}(LD'N'\log N')+\mathcal{O}(N'\log N')+\mathcal{O}(N))$

We begin the discussion by comparing PCTs for local prediction of multiple targets and PCTs for global prediction of multiple targets. The dominant term in the computational complexity of local PCTs for multiple targets is $\mathcal{O}(TDN\log^2 N)$. Let us assume that $T < \log N$. This means that the dominant term in the computational complexity of global PCTs for multiple targets is $\mathcal{O}(DN\log^2 N)$. Considering this, global PCTs have $\mathcal{O}(T)$ times lower computational complexity than local PCTs. On the other hand, if we assume that $T > \log N$, then the dominant term in the computational complexity of global PCTs is $\mathcal{O}(TDN\log N)$. In this case, global PCTs have $\mathcal{O}(\log N)$ times lower computational complexity than local PCTs.

Global PCTs have lower computational complexity than local PCTs mainly due to the multiple repetitions of the sorting of the numeric attributes. The difference in the computational complexity is further amplified by the fact that global PCTs are smaller, on average, than the trees constructed for each target variable separately. More to the point, global PCTs are faster to construct also because of some Java compiler issues. Namely, it is faster and easier to multiply/add/subtract two arrays with size T, than to multiply/add/subtract two arrays with size one T times (i.e., variables represented as doubles).

Next, we compare the computational complexity of (global) PCTs for HMC and (local) PCTs for HSC. The dominant term in the computational complexity of the PCTs for HSC is $\mathcal{O}(GDN\log^2 N)$. If we assume that $L < \log N$, then the dominant term in the computational complexity of the PCTs for HMC is $\mathcal{O}(DN\log^2 N)$. This means that the global PCTs have $\mathcal{O}(G)$ smaller computational complexity than local PCTs. Next, if we assume that $L > \log N$, then the dominant term in the computational complexity of global PCTs is $\mathcal{O}(LMN\log N)$. Furthermore, if the target hierarchy is tree-shaped then $G = L-1$, i.e., $G \approx L$ and the global PCTs have $\mathcal{O}(\log N)$ times smaller computational complexity than local PCTs. If the target classes are organized into a directed acyclic graph, then $G > L$ and the global PCTs have $\mathcal{O}(\frac{G}{L}\log N)$ times smaller computational complexity than local PCTs. In this case, the computational advantage of the global PCTs depends of the average number of parents that classes have. If the classes have more parents on average, the $\frac{G}{L}$ ratio will be larger and the computational complexity of the global PCTs will be lower.

The comparison of global and local ensemble methods by their computational complexities follows the same conclusions as the comparison of the global and local PCTs. Global ensemble methods have lower computational complexities than the local ensemble methods: $\mathcal{O}(T)$ or $\mathcal{O}(\log N)$ times for predicting multiple targets and $\mathcal{O}(G)$ or $\mathcal{O}(\frac{G}{L}\log N)$ for HMC. Furthermore, random forests and bagging of subspaces have an additional ad-

vantage over bagging because they use a subset of the descriptive attributes for tree construction.

Finally, we discuss in detail the computational complexity of random forests for global prediction of structured outputs. Random forests (together with bagging of subspaces) are very efficient ensemble methods and have very good predictive performance. The upper bounds of the computational complexity of global random forests for predicting multiple targets is $k(\mathcal{O}(D'N'\log^2 N') + \mathcal{O}(TD'N'\log N'))$ and for HMC is $k(\mathcal{O}(D'N'\log^2 N') + \mathcal{O}(LM'N'\log N'))$. The complexity of global random forests depends linearly on the number of base predictive models ($\mathcal{O}(k)$) and logarithmically (or via some other user defined function that determines the number of features sampled) on the number of numeric descriptive attributes ($\mathcal{O}(D')$). Furthermore, if $T > \log N$ and $L > \log N$ then the complexity will depend linearly from the size of the structured output (i.e., number of target variables and number of classes in the hierarchy) and $\mathcal{O}(N'\log^2 N')$ on the number of training instances. If $T < \log N$ and $L < \log N$, then the complexity of the random forests depends $\mathcal{O}(N'\log N'^2)$ on the number of training instances, and not on the size of the structured output.

The computational efficiency of the method for predicting structured outputs proposed in Gärtner and Vembu (2009), one of the most recent and most efficient methods for predicting structured outputs, depends polynomially on the size of the structured outputs and the number of training instances. On the other hand, the complexity of global random forests, depends linearly from the number of base predictive models (typically the ensembles have at most hundreds of base predictive models), logarithmically on the number of continuous descriptive attributes and $N\log N$ on the number of training instances. Furthermore, if the size of the output is larger than $\log N$ then the computational complexity will depend linearly on the output size. All in all, global random forests (and global ensembles, in general) are very efficient methods for predicting structured outputs.

5 Experimental evaluation

To evaluate the proposed ensemble methods, we conduct an extensive experimental evaluation on a large number of datasets. We first describe the experimental design, including the datasets. We then present and discuss the results of the experiments.

5.1 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 used 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.

5.1.1 Experimental questions

Given the methodology from Chapters 3 and 4, 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 (ST) and a separate tree for each hierarchy edge (HSC). Second, we learn PCTs that predict the entire structured output simultaneously: a tree for the complete target tuple (MT) and a tree for the complete hierarchy (HMC). Finally, we construct the ensemble classifiers in the same manner 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 for 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 with the algorithms that predict the components of the structured outputs separately. First, we inspect the predictive performance of all the algorithms. Next, 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 predicting the components of the structured output. At the end, we compare the running times and the sizes of the obtained models.

5.1.2 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 based on the type of their output: 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 5.1, 5.2, and 5.3, respectively.

The datasets with multiple continuous targets (13 in total, see Table 5.1) are mainly from the domain of ecological modelling. The datasets with multiple discrete targets (9 in total, see Table 5.2) are from various domains: ecological modelling (*Sigmea Real* and *Water Quality*), biological (*Yeast*), multimedia (*Scene* and *Emotions*) and media space analysis (*Mediana*). 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 5.3). Note that two datasets from the biological domain have a hierarchy organized as a directed acyclic graph (they have GO – Gene ontology – in the dataset name), and the remaining datasets have tree-shaped hierarchies. The datasets and their statistics and descriptions can be downloaded from `http://kt.ijs.si/DragiKocev/PhD/resources/`. For further details on the datasets, we refer the reader to the referenced literature.

Table 5.1: 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 (Blockeel *et al.*, 1999; Džeroski *et al.*, 2000)	1060	0/16	14

Table 5.2: 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 (Blockeel *et al.*, 1999; Džeroski *et al.*, 2000)	1060	0/16	14
Yeast (Elisseeff and Weston, 2001)	2417	0/103	14

Table 5.3: 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

5.1.3 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 assessed using some evaluation measure. The different machine learning tasks, described in Section 2.1, use 'task-specific' evaluation measures. We first describe the evaluation 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, root mean squared error and relative root mean squared error ($RRMSE$). For each of these measures, we performed statistical analysis (i.e., performed tests for statistical significance) 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 (shown in Table 5.4): 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. Micro averaging calculates the numbers of true positives/negatives and false positives/negatives by adding the values per class and then uses those values to calculate *Precision*, for instances. Macro averaging calculates the given evaluation measure for each class separately and then averages those values. 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. However, the conclusions drawn from the evaluation of the performance of the algorithms using the other measures concur with the ones presented here.

Table 5.4: Evaluation measures for classification; TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives and FN is the number of false negatives.

Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$
Precision (P)	$\frac{TP}{TP+FP}$
Recall (R)	$\frac{TP}{TP+FN}$
F-score (F)	$2 \cdot \frac{P \cdot R}{P+R}$
Matthews coefficient	$\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}}$
Balanced accuracy	$\frac{1}{2} \cdot \left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$
Discriminant power	$\frac{\sqrt{3}}{\pi} \cdot \ln\left(\frac{TP}{FP} \cdot \frac{TN}{FN}\right)$

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 Precision-Recall 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 Precision-Recall evaluation of these algorithms is most suitable in this context because we are more interested in correctly predicting 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).

5.1.4 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 outputs 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 results in 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, as an input parameter, the size of the ensemble: 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 outputs 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 $\lfloor \log_2 |D| \rfloor + 1$, which is recommended by Breiman (2001a). For the hierarchical classification datasets, we used $\lfloor 0.1 \cdot |D| \rfloor + 1$, 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 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 5.3, 5.4, 5.7, 5.8, 5.11 and 5.12). The diagrams plot the average ranks of the algorithms and connect the ones whose average ranks are smaller than a given value, called critical distance. The critical distance depends on the level of the statistical significance, in our case 0.05. The difference in the performance of the algorithms connected with a line is not statistically significant at the given significance level.

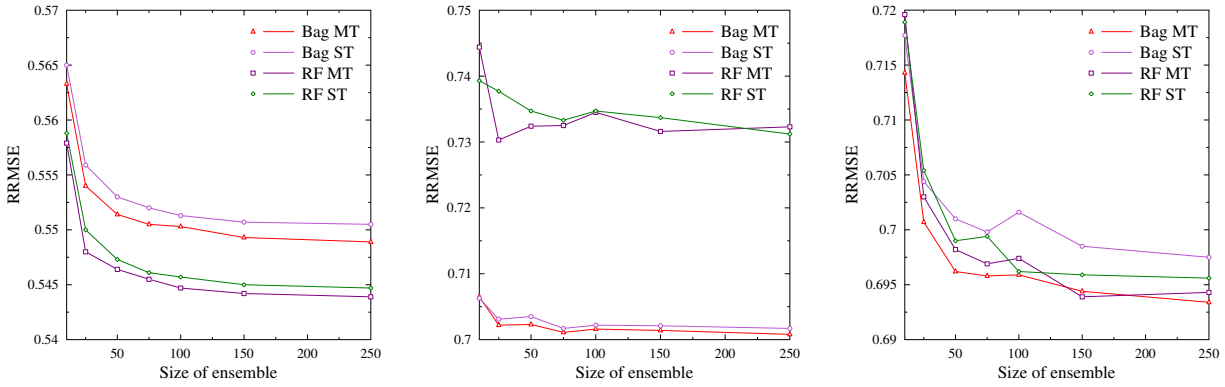
5.2 Results and discussion

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 with 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.

5.2.1 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 5.1, 5.2, 5.3 and 5.4. First, we discuss the results with respect to the saturation curves (Figure 5.1). Next, we discuss the statistical comparison of predictive performance (Figure 5.3). Finally, we compare the efficiency of the algorithms (Figure 5.4).

In Figure 5.1, 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 5.1(c)



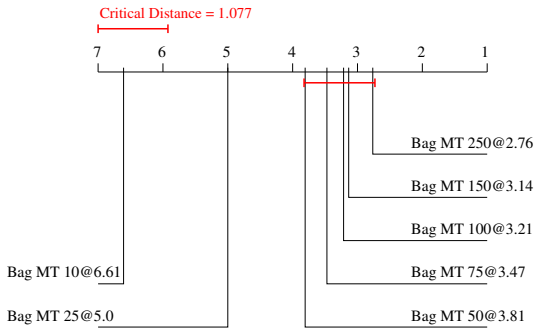
(a) *Forestry-Kras* (b) *Soil quality* (c) *Overall*

Figure 5.1: 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. Note that in order to increase visibility the range of the y-axis is adapted for each curve. The algorithm names are abbreviated as follows: bagging - Bag, random forests - RF, multi-target prediction - MT and single-target prediction - ST.

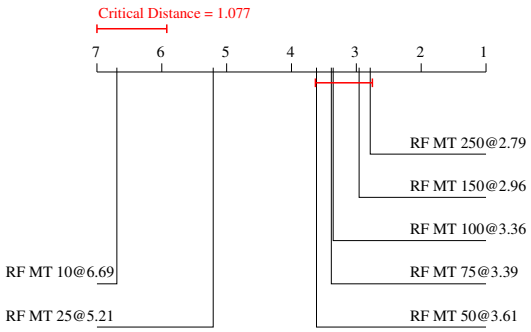
averaged across all datasets), they still provide useful insight into the performance of the algorithms. Random forests perform better than bagging, both when predicting the multiple targets simultaneously and separately, on the 'larger' datasets (the ones with more than 10000 examples), such as *Forestry-Kras* from Figure 5.1(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 5.1(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 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 5.1(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).

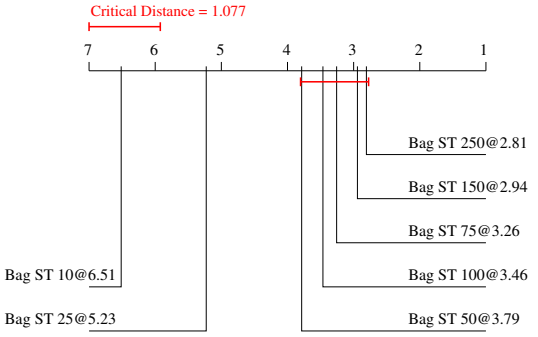
The saturation point in such a curve is the ensemble size after which the difference in performance achieved by increasing the ensemble size is no longer statistically significant. To detect the saturation point, we perform Friedman and Nemenyi tests for assessment of statistical significance for each method/algorithm separately. Figure 5.2 shows the results



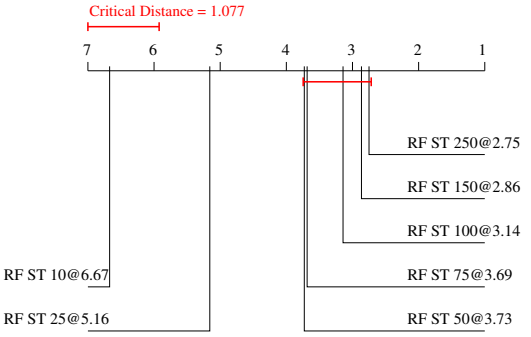
(a) Bagging for multiple targets



(b) Random forests for multiple targets



(c) Bagging for single target



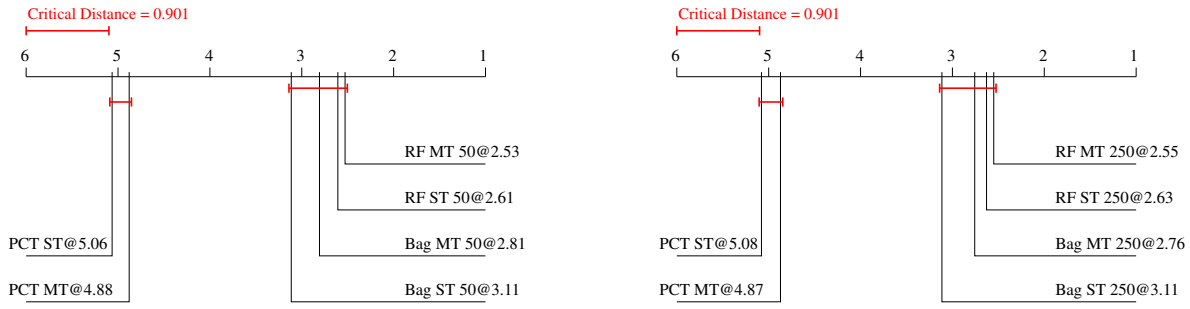
(d) Random forests for single target

Figure 5.2: 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 5.1.

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 5.3) 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, 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.



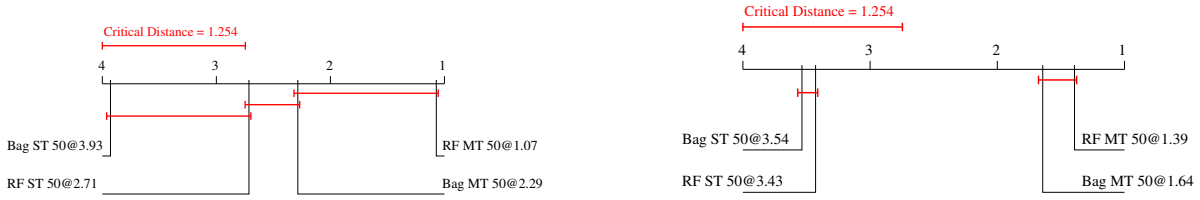
(a) ensemble with 50 trees (b) ensemble with 250 trees

Figure 5.3: 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 5.1 with the addition of single predicting clustering trees - *PCT*.

Finally, we compare the algorithms by their running time and the size of the models, considering ensembles that consist of 50 trees (see Figure 5.4). Note that the same conclusions are valid for the other ensemble sizes. 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.

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



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

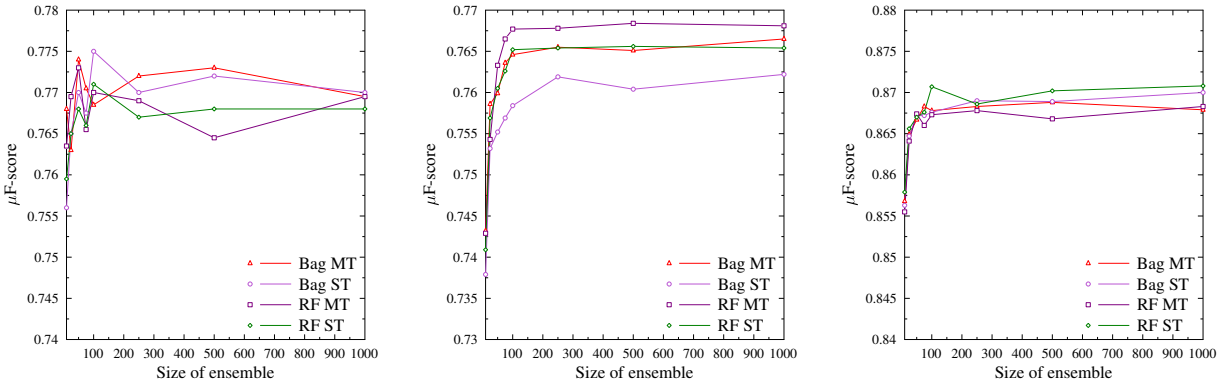
Figure 5.4: Efficiency (running time and model size) of the ensembles for prediction of multiple continuous targets. The size of the ensembles is 50 trees.

5.2.2 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 5.1.3. 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 5.5, 5.6, 5.7 and 5.8. In Figure 5.5, we present the saturation curves. Next, we discuss the statistical analysis of the results (Figure 5.7). At the end, we compare the algorithms in terms of efficiency (Figure 5.8).

In Figure 5.5, 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 5.5(c) averaged across all datasets). 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 5.5(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 (5 datasets from Table 5.2), 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 5.5(b)); (2) on the datasets with more than 30 descriptive variables (in total 4 datasets from Table 5.2), 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 5.5(c)). Finally, as for the multiple



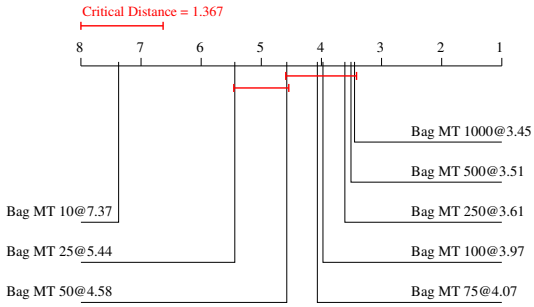
(a) *Sigmea real* (b) *Water quality* (c) *Overall*

Figure 5.5: 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. Note that, in order to increase visibility the range of the y-axis is adapted for each curve. The algorithm names are abbreviated as follows: bagging - Bag, random forests - RF, multi-target prediction - MT and single-target prediction - ST.

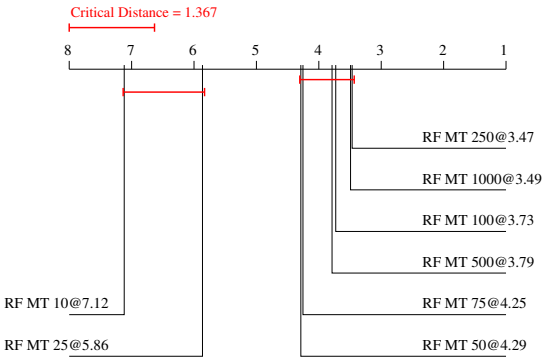
continuous targets, there is no connection between the predictive performance of the algorithms and the size of the target tuple.

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 5.6. 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.

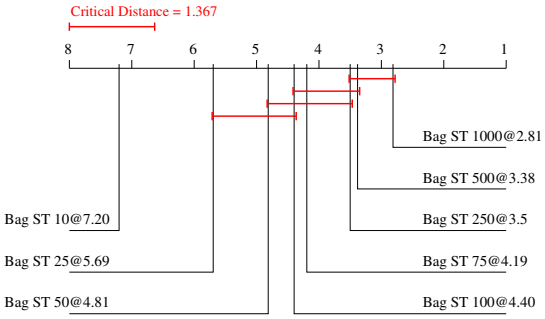
The results from the statistical analysis of the predictive performance ($\mu F - score$) are shown in Figure 5.7. The statistical tests reveal that there is no statistically significant difference in the performance of the ensemble methods and that all ensemble methods perform statistically significantly better than the corresponding single trees. When the ensembles have 50 trees (Figure 5.7(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 5.7(b)), with the difference that now random forests for simultaneous prediction are the worst performing method (average rank 3.26) and the



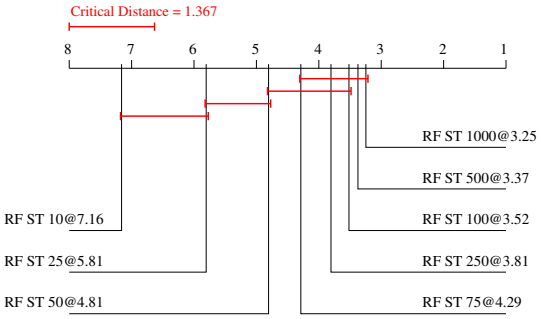
(a) Bagging for multiple targets



(b) Random forests for multiple targets



(c) Bagging for single target

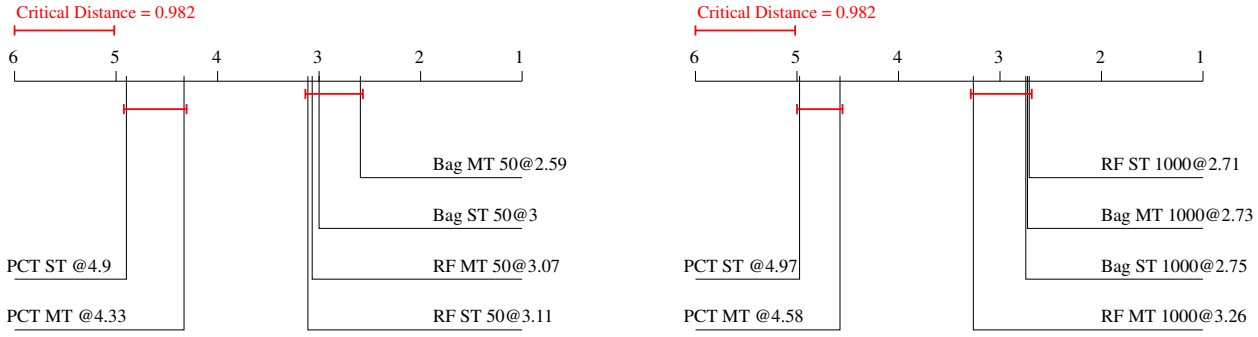


(d) Random forests for single target

Figure 5.6: 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 5.5.

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 5.8(a)) and model sizes (Figure 5.8(b)). Concerning the running time, we can only state that the random forests for predicting multiple targets simultaneously significantly 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.

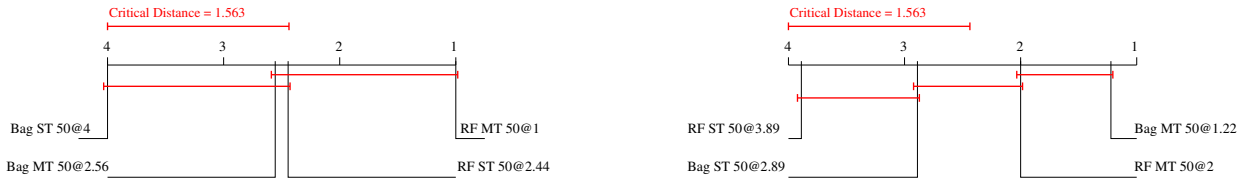


(a) ensembles with 50 trees (b) ensembles with 1000 trees

Figure 5.7: 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 5.5, with the addition of single predicting clustering trees - *PCT*.

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 significantly 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



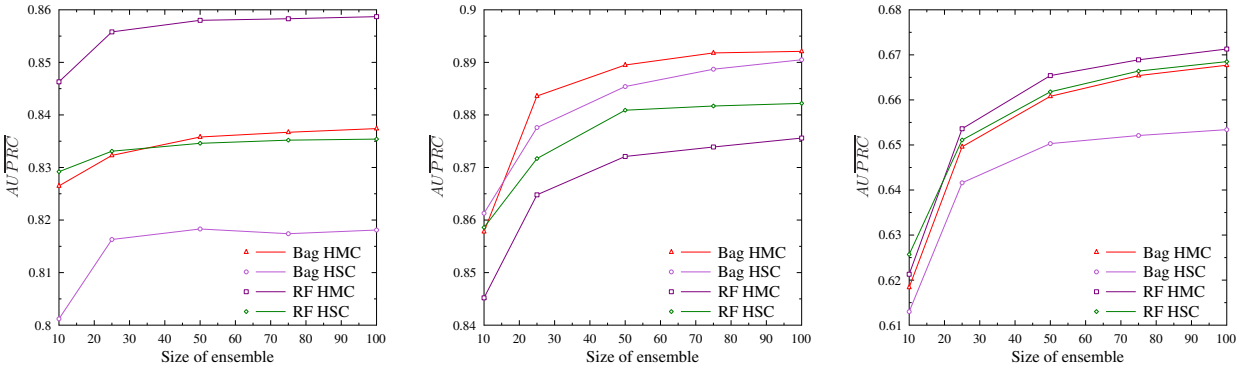
(a) *Time efficiency* (b) *Size of the models*

Figure 5.8: Efficiency of the ensembles for prediction of multiple discrete targets. The size of the ensembles is 50 trees.

simultaneous prediction are faster (especially random forests) and produce smaller models (especially bagging) than the ensemble methods for separate prediction.

5.2.3 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 5.9), statistical tests (Figures 5.10 and 5.11) and efficiency figures (Figure 5.12).



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

Figure 5.9: 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. Note that, in order to increase visibility the range of the y-axis is adapted for each curve. The algorithm names are abbreviated as follows: bagging - *Bag*, random forests - *RF*, hierarchical multi-label classification - HMC and hierarchical single-label classification - HSC.

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 directed acyclic graph (for instance, see the saturation curve for the *SCOP-GO* dataset in Figure 5.9(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 5.9(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

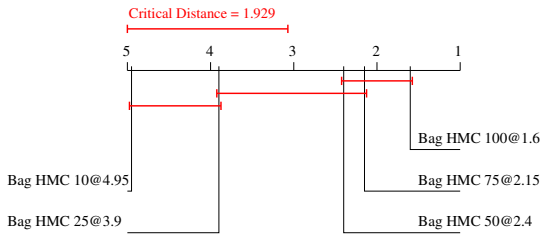
of HMC. We hypothesize that this is because of the large number of descriptive variables. The performance of ensembles of HMC on text classification datasets should be further investigated.

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.

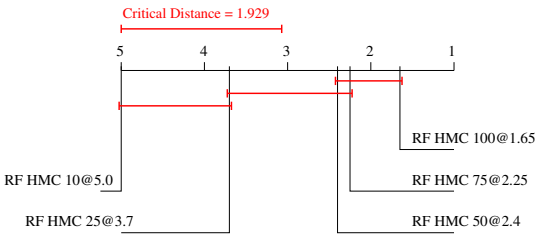
The overall saturation curve (Figure 5.9(c)) shows the performance of the algorithms averaged over all datasets from the three domains. The best performing method is random forests for HMC and the worst performing method is bagging for HSC. To further investigate the differences in performance, we perform 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 5.10. 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 5.11(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.

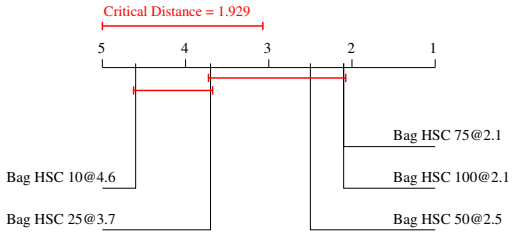
Finally, we compare the algorithms by their efficiency when they contain 50 trees (running times in Figure 5.12(a) and model sizes in Figure 5.12(b)). Random forests for HMC are statistically significantly faster than both bagging for HMC and HSC, while random forests for HSC are significantly faster than bagging for HSC. The models of



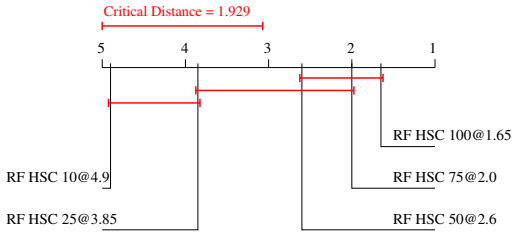
(a) Bagging for HMC



(b) Random forests for HMC

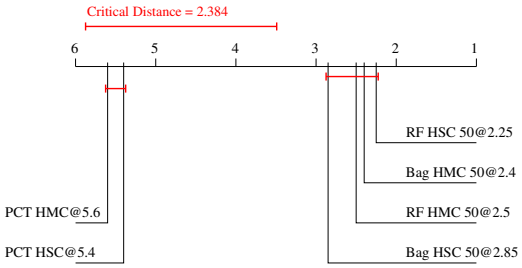


(c) Bagging for HSC

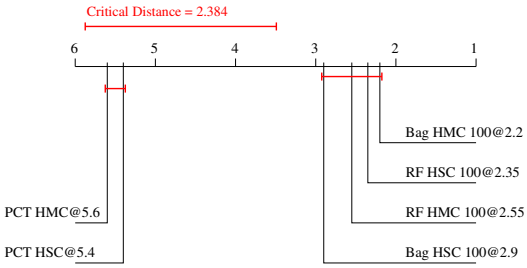


(d) Random forests for HSC

Figure 5.10: 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 5.9.



(a) ensemble with 50 trees

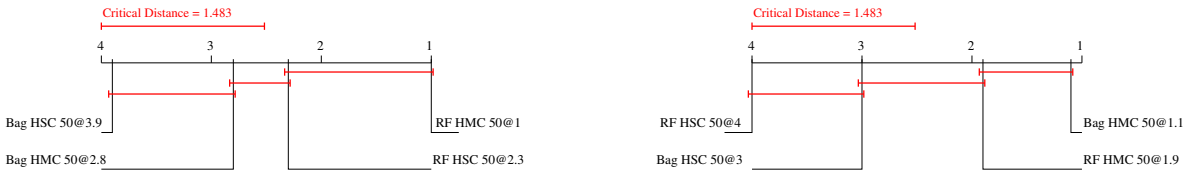


(b) ensemble with 100 trees

Figure 5.11: 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 5.9 with the addition of single predicting clustering trees - *PCT*.

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 5.12: 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.

5.3 Summary of the results

In this section, we summarize the findings from the empirical evaluation of the proposed methods. The results show that ensembles lift the predictive performance of a single classifier also if the output is structured. Next, we construct saturation curves for the ensemble methods (for the ensembles predicting both the structured output and the components). The saturation curves help us to determine the number of base classifiers in an ensemble that offers optimal predictive performance and efficiency. 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 (with the selected number of base predictive models).

We performed the empirical evaluation over a wide range of datasets. In particular, we used 13 datasets for the task of multi-target regression, 9 datasets for the task of multi-target classification and 10 datasets for the task of hierarchical multi-label classification. 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 the significance level of 0.05. Previously this was shown only on applications where the target is a single continuous or discrete variable. This finding is valid for all three machine learning tasks that we consider in this thesis. This suggests that the non-trivial relations that might exist between the components of the structure are still captured when combining predictions of several classifiers or when injecting randomness in the learning algorithm.

- The saturation 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 predictive performance and efficiency. Furthermore, the saturation curves show that, in the majority of the domains, the ensembles of PCTs have better predictive performance than the ensembles that predict the components. This is especially the case when the ensembles contain fewer PCTs.

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

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 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.

Ecological Modelling 220 (2009) 1159–1168

Contents lists available at ScienceDirect

Ecological Modelling

journal homepage: www.elsevier.com/locate/ecolmodel

Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition

Dragi Kocev[a,*], Sašo Džeroski[a], Matt D. White[b], Graeme R. Newell[b], Peter Griffioen[c]

[a] Dept. of Knowledge Technologies, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
[b] Dept. of Sustainability and Environment, Arthur Rylah Institute for Environmental Research, 123 Brown Street, Heidelberg, Victoria 3084, Australia
[c] Acromap, Pty. Ltd., 37 Gloucester Drive, Heidelberg, Victoria 3084, Australia

ARTICLE INFO

Article history:
Received 12 May 2008
Received in revised form 20 January 2009
Accepted 25 January 2009
Available online 13 March 2009

Keywords:
Multi-target prediction
Ensemble methods
Regression trees
Indigenous vegetation
Vegetation quality/condition

ABSTRACT

An important consideration in conservation and biodiversity planning is an appreciation of the condition or integrity of ecosystems. In this study, we have applied various machine learning methods to the problem of predicting the condition or quality of the remnant indigenous vegetation across an extensive area of south-eastern Australia—the state of Victoria. The field data were obtained using the 'habitat hectares' approach. This rapid assessment technique produces multiple scores that describe the condition of various attributes of the vegetation at a given site. Multiple sites were assessed and subsequently circumscribed with GIS and remote-sensed data.

We explore and compare two approaches for modelling this type of data: to learn a model for each score separately (single-target approach, a regression tree), or to learn one model for all scores simultaneously (multi-target approach, a multi-target regression tree). In order to lift the predictive performance, we also employ ensembles (bagging and random forests) of regression trees and multi-target regression trees. Our results demonstrate the advantages of a multi-target over a single-target modelling approach. While there is no statistically significant difference between the multi-target and single-target models in terms of model performance, the multi-target models are smaller and faster to learn than the single-target ones. Ensembles of multi-target models, also, improve the spatial prediction of condition.

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, in support of biodiversity planning, management and investment decisions.

1. Introduction

Governments and other agencies worldwide are increasingly required to demonstrate their compliance with the policies and legislation relevant to the protection and management of remnant indigenous vegetation (Parkes and Lyon, 2006). To this end, government agencies are seeking to extend the requisite knowledge base and representation of vegetation beyond just 'extent' and 'type', to incorporate the notion of 'condition' or 'quality'. The concept of vegetation condition is typically idiosyncratic and/or context-specific. For example, the performance or quality of native vegetation could be evaluated in terms of its capacity to deliver services such as energy storage (including carbon sequestration), nutrient cycling, landscape stability, fodder production for domestic stock or habitat for species. A key challenge has been to develop metrics that facilitate comparisons of condition both within and between disparate ecosystem types. Recent attempts have been made to clarify these concepts (Andreasen et al., 2001; Gibbons et al., 2006), and develop general and widely applicable metrics and indices for assessing vegetation or ecosystem condition from a biodiversity perspective (Parkes et al., 2003; Scholes and Biggs, 2005; Oliver, 2004; Eyre et al., 2006; Gibbons et al., 2009).

With an increasing emphasis on landscape scale planning for biodiversity investment (Margules and Pressey, 2000; Rouget et al., 2006; Knight et al., 2006; Moilanen, 2007) and widespread access to Geographic Information Systems (GIS) and associated data and software, the production of maps or spatially explicit models of landscape indices, species distributions and other ecological phenomena has become commonplace (see Li and Wu, 2004; Guisan and Thuiller, 2005). The apparent utility of compound indices, such as vegetation condition or ecosystem integrity presents a generic problem for the land management agencies which employ them:

* Corresponding author.
E-mail addresses: Dragi.Kocev@ijs.si (D. Kocev), Saso.Dzeroski@ijs.si (S. Džeroski), Matt.White@dse.vic.gov.au (M.D. White), Graeme.Newell@dse.vic.gov.au (G.R. Newell), pgriffioen@acromap.com (P. Griffioen).

0304-3800/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ecolmodel.2009.01.037

can we usefully predict such attributes from site data across extensive geographic regions, from a vector of covariate remote sensed and ancillary environmental data?

The focus of this study is to take data from site assessments employing a multi-component index of vegetation condition and attempt to fit a generalized view of this index over an extensive area—in this case the State of Victoria, Australia, an area of some 227,000 km^2. So, the problem that we are addressing is how to predict multiple target variables (responses) from a vector of ecological/remote-sensed data. We employed two modelling scenarios: (1) learn a model for each component of the overall index separately and (2) learn a model for all component scores simultaneously. For the first scenario, we applied regression trees (RTs) (Breiman et al., 1984) and ensembles of RTs (Breiman, 1996, 2001) to the problem, while for the second, we applied multi-target regression trees (MTRTs) (Struyf and Džeroski, 2006) and ensembles of MTRTs (Kocev et al., 2007).

Regression trees are decision trees that predict the value of a single numeric target variable. The multi-target regression trees are a generalization of RTs. They are able to predict the value of multiple numeric target variables. Their main advantages (over building a separate model for each target attribute) are: (1) a multi-target model is smaller than the total size of the individual models for all target attributes and (2) a multi-target model explains dependencies between different target attributes (Blockeel et al., 1998; Struyf and Džeroski, 2006). We selected regression trees and multi-target regression trees because they are easy to understand and interpret and yet offer satisfactory predictive power.

To obtain models that have improved predictive performance we used ensembles. Ensemble learning combines the predictions of multiple models and lifts the predictive performance of their base classifiers, both in the single-target (Breiman, 1996) and the multi-target setting (Kocev et al., 2007). We focus on the two most widely used ensemble learning methods that use tree models as base classifiers: bagging (Breiman, 1996) and random forests (Breiman, 2001).

We perform the analysis using two scenarios: (1) we learn pruned tree models (smaller tree models) to obtain some knowledge and understanding about the condition of the indigenous vegetation and (2) we learn ensembles of trees opting for better predictive performance that will yield more precise and reliable maps of the vegetation condition.

The development of predictive models of condition for remnant indigenous vegetation may assist in identifying the relative importance of associated biophysical and landscape attributes in explaining observed condition states, across vegetation types and landscape scales. In addition, spatially explicit models of condition, could, when used in conjunction with other data, inform natural resource investment decisions, statutory protection and reserve design, while providing a basis for new forms of environmental accounting and potentially monitoring landscape change.

The remainder of this paper is organized as follows: In Section 2, we describe our modelling methodology, and in Section 3 the data. The experimental setup for data analysis is presented in Section 4. In Section 5, we present, discuss and compare the models that we obtained. Finally, we outline our conclusions in Section 6.

2. Machine learning methodology

2.1. Regression trees

Regression trees are decision trees that predict the value of a numeric target variable (Breiman et al., 1984). Regression trees are hierarchical structures, where the internal nodes contain tests on the input attributes. Each branch of an internal test corresponds to an outcome of the test, and the prediction for the value of the target attribute is stored in a leaf. Each leaf of a regression tree contains a constant value as a prediction for the target variable (regression trees represent piece-wise constant functions).

To obtain the prediction for a new data record, the record is sorted down the tree, starting from the root (the top-most node of the tree). For each internal node that is encountered on the path, the test that is stored in the node is applied. Depending on the outcome of the test, the path continues along the corresponding branch (to the corresponding subtree). The resulting prediction of the tree is taken from the leaf at the end of the path. The tests in the internal nodes can have more than two outcomes (this is usually the case when the test is on discrete-valued attributes where a separate branch/subtree is created for each value). Typically each test has two outcomes: the test has succeeded or the test has failed. The trees in this case are also called binary trees.

2.2. Multi-target regression trees

Multi-target regression trees (Blockeel et al., 1998; Struyf and Džeroski, 2006) generalize regression trees to the prediction of several numeric target attributes simultaneously. The leaves of a multi-target regression tree store a vector, instead of storing a single numeric value. Each component of this vector is a prediction for one of the target attributes. An example of a multi-target regression tree is shown in Fig. 3.

A multi-target regression tree (of which a regression tree is the special case with a single response variable) is usually constructed with a recursive partitioning algorithm from a training set of records. The algorithm is known as Top-Down Induction of Decision Trees (TDIDT). The records include measured values of the descriptive and the target attributes. The tests in the internal nodes of the tree refer to the descriptive, while the predicted values in the leaves refer to the target attributes.

The TDIDT algorithm starts by selecting a test for the root node. Based on this test the training set is partitioned into subsets according to the test outcome. In the case of binary trees, the training set is split into two subsets: one containing the records for which the test succeeds (typically the left subtree) and the other contains the records for which the test fails (typically the right subtree). This procedure is recursively repeated to construct the subtrees.

The partitioning process stops when a stopping criterion is satisfied (e.g., the number of records in the induced subsets is smaller than some predefined value; the length of the path from the root to the current subset exceeds some predefined value, etc.). In that case, the prediction vector is calculated and stored in a leaf. The components of the prediction vector are the mean values of the target attributes calculated over the records that are sorted into the leaf.

One of the most important steps in the tree induction algorithm is the test selection procedure. For each node a test is selected by using a heuristic function computed on the training data. The goal of the heuristic is to guide the algorithm towards smaller trees with good predictive performance.

In this paper, we use the CLUS (Blockeel and Struyf, 2002) system for constructing (multi-target) regression trees (the system is available at http://www.cs.kuleuven.be/~dtai/clus/). The heuristic used for selecting the attribute tests (that define the internal nodes) in this algorithm is the intra-cluster variance summed over the subsets induced by the test. Intra-cluster variance is defined as $N \cdot \sum_{t=1}^{T} Var[y_t]$ with N the number of examples in the cluster, T the number of target variables, and $Var[y_t]$ the variance of target variable y_t in the cluster. The variance function is standardized so that the relative contribution of the different targets to the heuris-

tic score is equal. Lower intra-subset variance results in predictions that are more accurate.

The multi-target regression trees are an instantiation of the predictive clustering trees (PCTs) framework proposed in (Blockeel et al., 1998). In the PCTs framework, a tree is viewed 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 PCTs can be instantiated for different tasks using adequate variance and prototype functions. So far, PCTs have been used to handle multiple targets (Struyf and Džeroski, 2006), time series (Džeroski et al., 2007) and hierarchical multi-label classification (Vens et al., 2008).

2.3. Ensembles

An ensemble method constructs a set of predictive models (called an ensemble) (Dietterich, 2000). An ensemble gives a prediction for a new data instance by combining the predictions of its models for that instance. For regression tasks, the predictions can be combined by averaging the outputs of the models.

In order for an ensemble to be more accurate than any of its individual members, the individual models need to be accurate and diverse (Hansen and Salamon, 1990). An accurate model is one that performs better than random guessing on new examples. A set of models is diverse if the models make different errors on new examples. The diversity in an ensemble can be introduced in various ways: by manipulating the training set (changing the weight of examples or changing the weight of attributes) or by manipulating the learning algorithm used to obtain the models.

Ensembles of MTRTs are sets of MTRTs, obtained by applying the same TDIDT algorithm. A prediction of an ensemble of MTRTs is obtained by averaging the predictions of its models. They are able to lift the predictive performance of a single MTRT (also in the case of a single target) (Breiman, 1996; Kocev et al., 2007). In this work, we use bagging and random forests, the two most widely used ensemble methods to produce ensembles of RTs and MTRTs. An illustration of these two methods is presented in Fig. 2.

2.3.1. Bagging

Bagging (Breiman, 1996) is an ensemble method that constructs the different models in the ensemble by making bootstrap replicates of the training set; these are used to construct individual models (Fig. 2). Each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set. The bootstrap sample and the training set have the same number of instances. Bagging can yield substantial gains in predictive performance, when applied to unstable learners (i.e., a learner for which small changes in the training set can result in large changes in the predictions), such as classification and regression tree learners (Breiman, 1996). The diversity in bagging comes from the variation in the training sets used to construct the individual models in the ensemble.

2.3.2. Random forests

A random forest (Breiman, 2001) is an ensemble of trees, where the diversity of the trees is obtained from two sources: (1) by using bootstrap sampling and (2) by changing the feature set during learning (this is done by a randomized decision tree algorithm, see Fig. 2). At each node in the decision tree, a random subset of the input features is taken and the best split is selected from this subset. The size of the random subset is given by a function F of the number of descriptive attributes M $\left(\text{e.g., } F = 1, F = \left\lfloor \sqrt{M} \right\rfloor, F = \left\lfloor \log_2 M \right\rfloor + 1, F = \left\lfloor M/2 \right\rfloor, \ldots\right)$. If $F = M$, then the random forests algorithm is equal to the bagging algorithm.

3. Data description

In this study, we use field data acquired using the habitat hectares approach (Parkes et al., 2003), a technique for the rapid assessment of vegetation condition, developed primarily for biodiversity conservation planning. 'Vegetation quality' in the 'habitat hectares' approach is defined as the degree to which the current vegetation differs from a 'benchmark' that represents the average characteristics of a mature and long-undisturbed stand of the same plant community. Against the benchmark, the decline in quality can be estimated for each vegetation type and dissimilar community assemblages, such as rainforests and savannahs can be compared by employing the same general index. This general approach has become a standard method used to quantify the condition of habitat within the state of Victoria (www.dse.vic.gov.au) and has been emulated to some degree by other jurisdictions within Australia (see Eyre et al., 2006; Gibbons et al., 2009).

The 'habitat hectares' score is the weighted sum of 7 site and 3 landscape scale metrics. The landscape components of the 'habitat hectares' score can be readily rendered spatially within a GIS using tools such as FRAGSTATS (McGarigal et al., 2002) and have not been further considered in this study. The objective was to make spatially explicit predictions of the 7 site scale components of the 'habitat hectares' score (hereafter referred to as the 'habitat hectares' site score or HHSS).

Employing the 'habitat hectares' approach, 16,967 'homogenous' sites were sampled within the State of Victoria, Australia (see Fig. 1) between the years 2001 and 2005. Each sampling point is described by 40 independent (or feature) variables (GIS and remote-sensed data with a pixel resolution of 30 m × 30 m) and 7 dependent (or target) variables (the HHSS). The HHSS is a numeric variable composed as a weighted average of the following components: *Large Trees*; *Tree (canopy) Cover*; *Understorey (non-tree) Strata*; *Lack of Weeds*; *Recruitment*; *Organic Litter*; and, *Logs*. Apart from *Lack of Weeds*, each component score was calculated comparing the current status of the vegetation with a benchmark. For a basic statistic of the target attributes see Table A2 in Appendix.

The *Large trees score* represents the number of large trees (both living and dead) that are present at the measuring site (compared to the 'benchmark' archetype). The *Tree Canopy* score assesses the

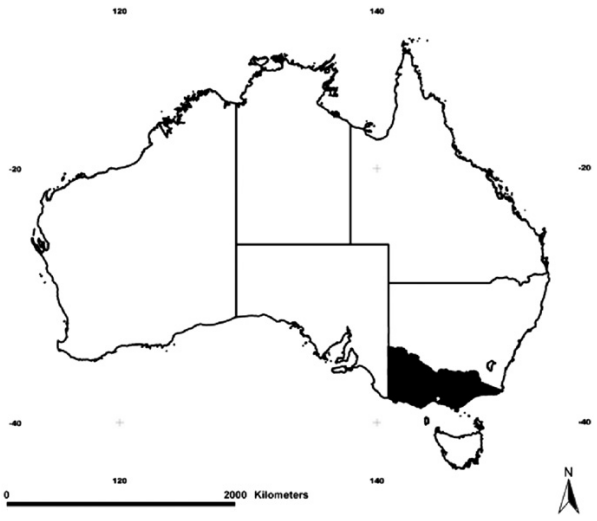


Fig. 1. Map of Australia with latitude and longitude shown. The State of Victoria in the south east of mainland Australia (our study area) is shaded.

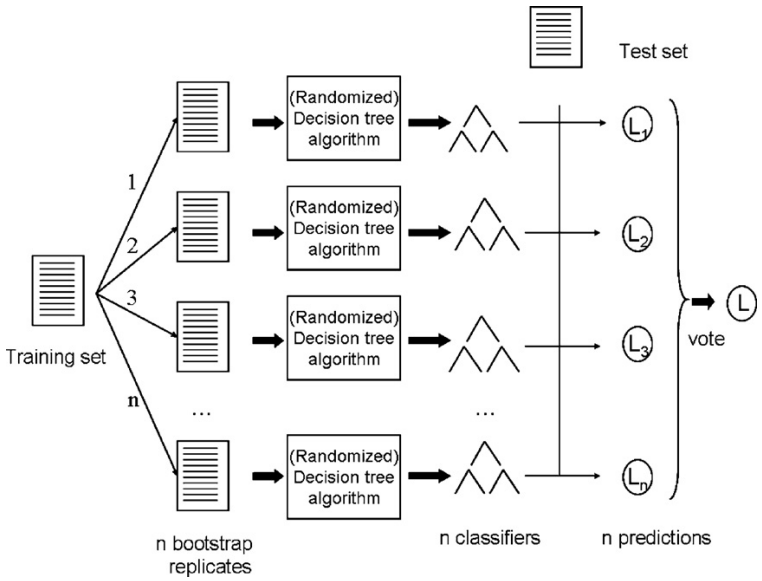


Fig. 2. A generic algorithm for learning ensembles of decision trees. Bagging uses a standard decision tree algorithm, while random forests use a randomized decision tree algorithm.

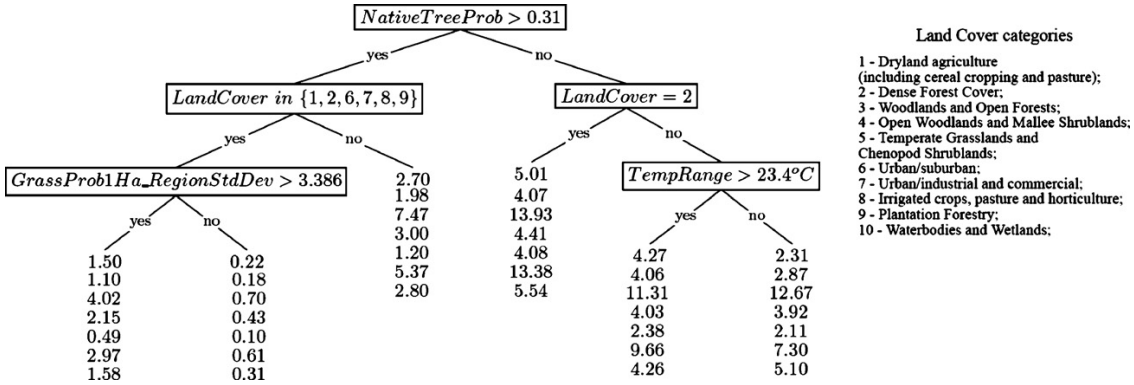


Fig. 3. Pruned multi-target regression tree (the target attributes in the leaves are ordered as per Table 1).

projective foliage cover of canopy trees in the stand, while the *Understorey score* assesses the abundance and diversity of various shrubs and forb/herb strata of a community. The understorey assessment includes only indigenous plant species. The *Lack of weeds score* is calculated from the cover of non-indigenous weed species.

The *Recruitment score* provides an indication of the level of regeneration of woody plant species and could be seen as a surrogate measure of the long-term viability of the site's structural characteristics. *Litter* represents both fine and coarse plant debris less than 10 cm diameter, while *Logs* represent the fallen timber or branches of trees that are substantially detached from the parent tree. An unabridged description of the 'habitat hectares' scores and methods can be found in (Parkes et al., 2003, 2004) and at www.dse.vic.gov.au.

The 40 independent variables include 39 continuous variables and one categorical variable (see Appendix Table A1). The categorical variable *LandCover* surface was derived from Landsat 7 TM spectral data. Classes were obtained by applying a k-means clustering procedure to a stack of median values for all Landsat 7 TM spectral bands and the Normalised Difference Vegetation Index (see Tucker, 1979) across the years spanning 1989–2005. The 50 classes that emerged from the unsupervised classification were 'lumped' into 10 bins that were partially informed by a landuse model similarly derived using an ANN process. This procedure allowed for temporal states consequent of clearing, wildfire and forest harvesting to remain evident within broad landuse classes. The 10 categories approximate to the descriptions in Fig. 3.

4. Experimental setup for data analysis

From the description of the data, we can define a multi-target regression problem, to be solved either by the single-target or the multi-target regression approach. The goal is to predict multiple continuous targets (responses, outputs) from a vector of descriptive (independent) variables. When applying the single-target approach, we learn a regression tree (or an ensemble of regression trees) for each target attribute separately (in our case, this means that we will have seven models or ensembles). With the multi-target approach we learn a multi-target regression tree (or ensemble of multi-target regression trees) for all target attributes (meaning that the output is a single model or ensemble).

We define two experimental scenarios. In the first scenario, the purpose of the modelling is to learn about the condition of the indigenous vegetation, and the relative importance of different biophysical and landscape attributes for that condition. We focus on interpretability to obtain such knowledge: the models need to have reasonable size and predictive power. We prune our models by setting the minimal number of instances in a leaf to 2048 (for both the single-target and multi-target approach). We varied this pruning parameter starting from 4 up till 4096 (taking numbers that are power of 2). We selected 2048, because it offered the best trade-off between the size and the performance of the model.

In the second scenario, we are not interested in the size of the models, but in their predictive power. To improve predictive performance, we use ensembles of unpruned single- and multi-target regression trees. We constructed ensembles consisting of 100 unpruned trees as recommended in (Bauer and Kohavi, 1999; Breiman, 1996, 2001). To combine the predictions of the trees we averaged the predictions from each tree. The size of the feature subsets for the random forests (F) was set to $F = \lfloor \log_2 M \rfloor + 1$ as suggested in (Breiman, 2001).

The learned models, from both scenarios, were then used to derive maps of remnant indigenous vegetation condition. Combined with other data, these maps will contribute to investment decisions in natural resource management, statutory protection and reserve design.

We compare the single-target and multi-target regression trees and ensembles. For baseline comparison, we use linear regression (as implemented in the WEKA system, Witten and Frank, 2005). We compared the methods in terms of their predictive performance (correlation coefficient between predictions and observed values, and root mean squared error—RMSE), time efficiency and model size. To estimate the predictive performance of the models on unseen data, we employed 10 times 10-fold cross-validation, thus we present the performance results with respective confidence intervals.

To assess whether the differences in performance are statistically significant, we employed the corrected Friedman test (Friedman, 1940) and the post hoc Nemenyi test (Nemenyi, 1963) as recommended by Demšar (2006). The Friedman test is a non-parametric test for multiple hypotheses testing. It ranks the algorithms according to their performance for each dataset separately, thus the best performing algorithm gets the rank of 1, second best the rank of 2. . ., and in case of ties it assigns average ranks (see Tables A2 and A3 in Appendix). Then, the Friedman test compares the average ranks of the algorithms and calculates the Friedman statistic χ_F^2, distributed according to the χ_F^2 distribution with $k-1$ degrees of freedom (k being the number of algorithms). Iman and Davenport (1980) show that the Friedman statistic is undesirably conservative and derive a corrected F-statistic that is distributed according to the F-distribution with $k-1$ and $(k-1)\times(N-1)$ degrees of freedom (k being the number of algorithms and N being the number of datasets).

If there is a statistically significant difference in the performance, than we can proceed with a post hoc test. The Nemenyi test is used to compare all the classifiers to each other. In this procedure, the performance of two classifiers is significantly different if their average ranks differ more than some critical distance. The critical distance depends on the number of algorithms, number of datasets and critical value (for a given significance level) that is based on the Studentized range statistic and can be found in statistical textbooks.

We present the result from the Nemenyi post hoc test with an average ranks diagram as suggested by Demšar (2006). An average ranks diagram can be seen in Fig. 6 (and Figure A1 in Appendix). The ranks are depicted on the axis, in such a manner that the best ranking algorithms are at the right-most side of the diagram. The algorithms that do not differ significantly are connected with a line.

5. Interpretation and evaluation of the vegetation condition models

We followed the analysis scenarios, described in the previous section and obtained two sets of models. The first set consists of single models (single-target regression trees and multi-target regression trees) and is concerned with the process of knowledge extraction (the first scenario). The second set consists of ensembles (of single-target and multi-target regression trees) and is concerned with better predictive power (the second scenario). All models are presented and discussed in the next subsections.

5.1. Models for knowledge extraction

In this sub-section, we present and discuss the models that were obtained with the first scenario described in Section 4. This set of models contains single-target regression trees for each target and one multi-target regression tree for all targets. We compared the performance of the models (Table 1), with both approaches yielding models of comparable predictive performance. The difference is in the interpretability and the time and size efficiency. The time needed for learning the MTRT was 2.33 s, while learning all regression trees takes 13.77 s (a speed-up of factor 5.9). The speed can be very important in real-time applications. Also, the MTRT is of size 11 (total number of nodes), while all single-target regression trees taken together have size 81 (a ratio of 7.4). These models are depicted in Figs. 3 (MTRT) and 4 (single-target trees).

One of the most important differences between the two approaches is in their interpretability. It is much easier to interpret one tree that describes all target variables, than interpreting each regression tree separately and trying to find some connection between the different models. The multi-target regression tree gives us a more general overview of the knowledge that is hidden in the data.

The pruned multi-target regression tree shown in Fig. 3 is readily interpreted, grouping the data into six clusters. The clusters that are in the right-hand side have (on average) a higher HHSS, indicating that such sites are likely to support indigenous vegetation close to its benchmark state. An intuitively robust, if somewhat simplified overview of vegetation condition across the State of Victoria is provided by a map generated from the multi-target solution and applied to the spatial covariates (Fig. 5).

The key variable at the initial node of the tree is *NativeTreeProb* which is the prediction of a Neural Network model (ANN) of the probability of a lack of native tree cover for Victoria, informed by a chronosequence of Landsat imagery from 1989 to 2005. A *NativeTreeProb* > 0.31 is equivalent to a predicted probability of greater than 0.31 of the subject pixel supporting tree cover. Given that three of the sub-components of the HHSS depend directly on the presence of tree cover (*Large tree score*, *Canopy cover score* and *Logs score*), its central role in partitioning the data is logical.

Table 1
Comparison of the performance of the pruned multi-target regression tree for all scores with the regression trees for each score (MTRT—multi-target regression tree, RT—regression tree).

Target	Correlation		RMSE	
	MTRT	RT	MTRT	RT
Large tree score	0.52 ± 0.02	0.53 ± 0.02	2.88 ± 0.06	2.86 ± 0.06
Tree canopy score	0.68 ± 0.02	0.68 ± 0.01	1.63 ± 0.04	1.64 ± 0.03
Understorey score	0.70 ± 0.02	0.71 ± 0.02	5.11 ± 0.13	5.05 ± 0.13
Litter score	0.72 ± 0.02	0.69 ± 0.02	1.43 ± 0.03	1.47 ± 0.04
Logs score	0.70 ± 0.02	0.71 ± 0.02	1.48 ± 0.03	1.47 ± 0.03
Weeds score	0.75 ± 0.01	0.78 ± 0.01	4.04 ± 0.09	3.83 ± 0.10
Recruitment score	0.61 ± 0.02	0.62 ± 0.02	2.59 ± 0.07	2.57 ± 0.06

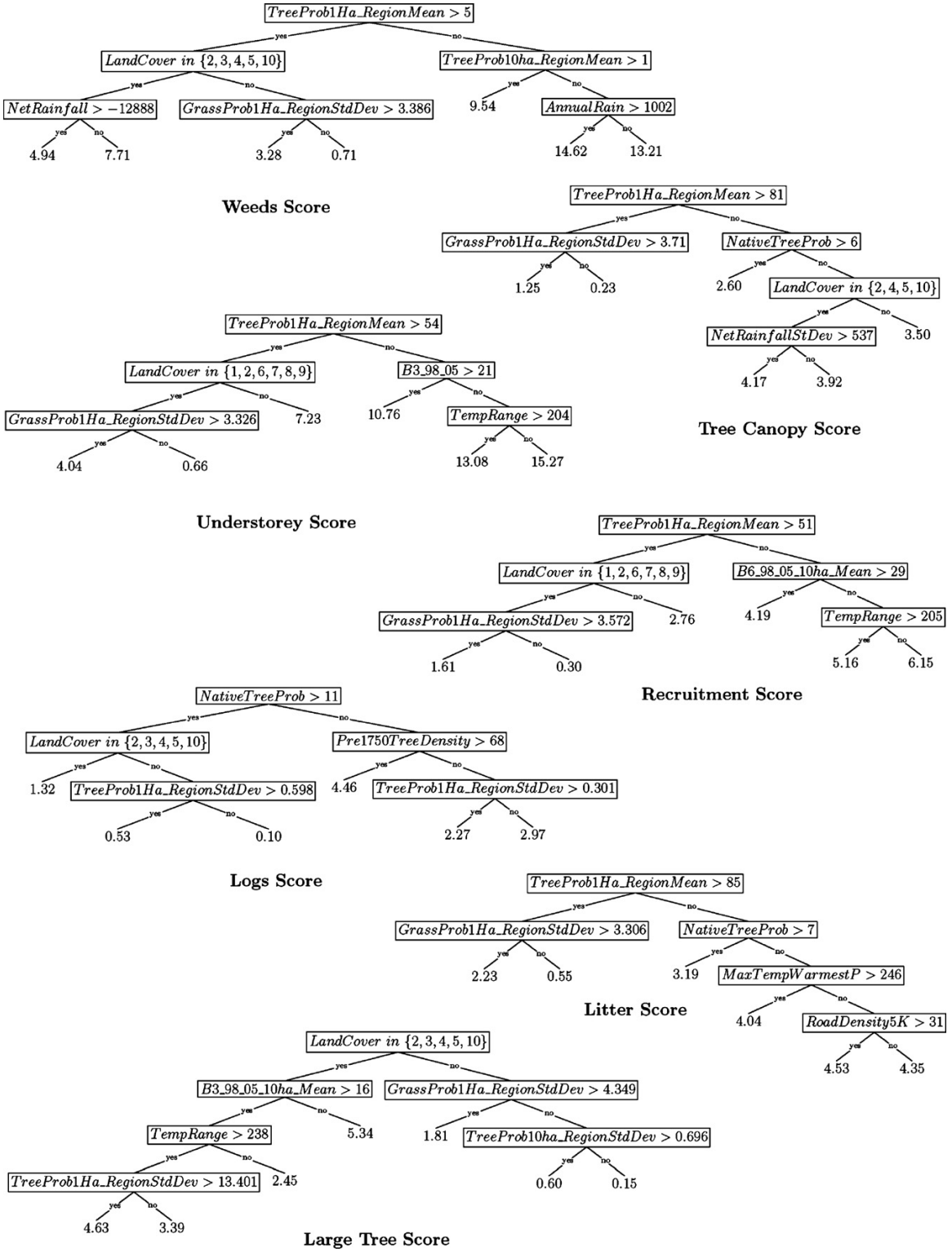


Fig. 4. Regression trees for each *Habitat Hectares* site score. The sum of these attributes comprises the overall *Habitat Hectares* site score.

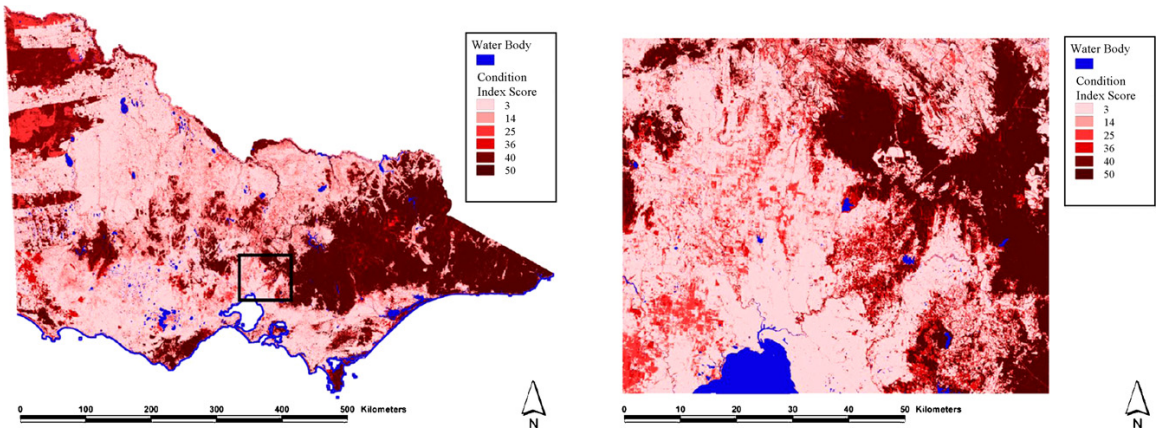


Fig. 5. Map of the condition of indigenous remnant vegetation in Victoria derived from the application of the pruned multi-target regression tree model to the explanatory features (left-hand side figure). The dark bordered rectangular inset refers to the area represented at higher resolution in the right-hand side figure.

Focusing on the 'no' branch of the tree (with the higher HHSS) the next decision node pivots on the membership of data to the *LandCover* category 2. *LandCover* category 2 corresponds with dense comparatively undisturbed forest cover and has the highest overall site score (of 50). All other *LandCover* categories proceed to the next node that partitions further, employing the variable *TempRange*. *TempRange* is one of many climate variables or features created using the ANUCLIM software package (see Houlder et al., 2000). This variable describes the annual range in temperature at a site by subtracting the climate model for the minimum temperature of coldest period of the year from the maximum temperature of the warmest period of the year. A *TempRange* of greater than 23.4 °C can be found in the semi-arid North West of Victoria where plant growth rates and consequently recovery from perturbation is generally slow.

The left-hand side of the tree, where the probability of tree cover is smaller than 0.31, is further partitioned by membership or otherwise of the *LandCover* categories 1, 2, 6, 7, 8, and 9. Apart from *LandCover* category 2 (i.e., Dense Forest Cover) these land cover types are all highly modified land use settings with correspondingly low habitat hectare scores. The small areas with *LandCover* category 2 that have a high probability of not finding tree cover (i.e. greater than 0.31) are likely to be feature data errors carried over from the land use mapping employed.

All these categories when *NativeTreeProb* is greater than 0.31 are assigned moderate condition scores (mean 25) by the pruned regression tree model. These are predominantly areas where tree cover is either absent, partially cleared or tree cover has been removed by recent wildfires. Fire scars are apparent in the North West region of the map. The incidence of fire has not been explicitly addressed in this study, however, future modelling will investigate the impact of fire on the HHSS and other condition indices through the inclusion of mapped fire boundaries derived from satellite imagery and historic cartographic sources.

The final node in the multi-target regression tree to be discussed here is regulated by the variable *Grass1ha RegionStdDev*. This variable is derived from an ANN model of the probability of native grass cover for every pixel in Victoria, informed by aforementioned chronosequence of Landsat imagery. A neighborhood of 1 ha around each pixel was interrogated and the standard deviation of the probability of indigenous grass cover across that area was obtained. Although speculative, this variable identifies spectrally uniform areas—regions that if they support treeless native vegetation could be relatively free of the degrading edge effects such as weed invasion that may emanate from surrounding land uses. The variable may be interpreted as a surrogate for the core area concept in landscape ecology (*sensu* Botequila Leitão et al., 2006) seen here as a useful predictor of grassland vegetation condition in Victoria.

The regression trees for each target attribute are shown in Fig. 4. If we compare Figs. 3 and 4, each of the components of the habitat hectare site score use different features and sequences of features to that of the tree that predicts the site score alone. This adds complexity and removes ecological naivety from the model. As with the single-target solution, we can closely examine the internal logic of each regression tree for the component scores. Prima-facie, each of the single-target regression trees is ecologically interpretable.

For example, if we just follow the positive or far left-hand side of the tree predicting Weed Score, it initially partitions the data on the basis of *TreeProb1HaRegionMean*: mean probability of detecting no tree cover within a 1 ha area around the subject pixel. This variable effectively divides the landscape into forests and treeless areas or areas with only scattered trees. Following the positive or left-hand side of the tree the data is further partitioned by the land cover classes. Classes 2, 3, 4, 5, and 10—all of these classes are natural or semi-natural areas and we should expect these areas to have a higher weed score (a high positive score reflects the absence of weeds rather than weed infestation) relative to other thinly treed areas. This is borne out by the regression tree. The final node is controlled by *NetRainfall*. *NetRainfall* is a variable that is derived from both mean monthly rainfall and mean evaporation rates. In essence it reflects the amount of effective rainfall (rainfall less evaporation) over an entire year. Once we have reached this node the model predicts that the drier and hotter a place is, the higher the weed score (provided we have satisfied earlier criteria). This reflects the current on-ground ecological reality in south-eastern Australia where there have been few deliberate introductions of exotic plant species into specialist habitat types, such as semi-arid regions, in comparison with temperate and sub-humid climatic regions that have been favoured by human settlement and intensive agriculture.

A further advantage of the multi-target approach is that is can reveal relationships between response variables. It is apparent that *Recruitment score* and *Understorey score* are positively related (see Fig. 4). The single-target regression trees of these scores are structurally identical and both employ very similar explanatory variables at similar junctures. Again, this is consistent with both field observation and ecological theory: a diverse and structurally intact understorey implies an adequate level of shrub and tree regeneration. The reverse is also likely. Within defined ecosystem types and states, a positive relationship between ecosystem function and structure is generally accepted by ecologists (Cortina et al., 2006; Bradshaw, 1984). Overall, the most important variables influ-

Table 2
Correlation coefficients of the obtained models (LR—linear regression, MTRT—multi-target regression tree, RT—regression tree, Bag—bagging, RF—random forests).

Target	LR	MTRT	RT	BagMTRT	Bag RT	RF MTRT	RF RT
Large tree score	0.61 ± 0.02	0.63 ± 0.02	0.60 ± 0.02	0.69 ± 0.01	0.69 ± 0.02	0.69 ± 0.01	0.69 ± 0.01
Tree canopy score	0.76 ± 0.01	0.76 ± 0.01	0.74 ± 0.02	0.80 ± 0.01	0.81 ± 0.01	0.81 ± 0.01	0.81 ± 0.01
Understorey score	0.77 ± 0.02	0.78 ± 0.01	0.77 ± 0.01	0.83 ± 0.01	0.83 ± 0.01	0.83 ± 0.01	0.83 ± 0.01
Litter score	0.76 ± 0.01	0.77 ± 0.01	0.76 ± 0.01	0.81 ± 0.01	0.82 ± 0.01	0.82 ± 0.01	0.82 ± 0.01
Logs score	0.75 ± 0.01	0.76 ± 0.01	0.75 ± 0.02	0.80 ± 0.01	0.80 ± 0.01	0.80 ± 0.01	0.80 ± 0.01
Weeds score	0.82 ± 0.01	0.83 ± 0.01	0.83 ± 0.01	0.87 ± 0.01	0.87 ± 0.01	0.87 ± 0.01	0.87 ± 0.01
Recruitment score	0.67 ± 0.02	0.69 ± 0.02	0.67 ± 0.02	0.74 ± 0.02	0.74 ± 0.01	0.74 ± 0.01	0.75 ± 0.01

Table 3
Root mean squared error of the obtained models (LR—linear regression, MTRT—multi-target regression tree, RT—regression tree, Bag—bagging, RF—random forests).

Target	LR	MTRT	RT	BagMTRT	Bag RT	RF MTRT	RF RT
Large tree score	2.66 ± 0.05	2.62 ± 0.05	2.72 ± 0.06	2.43 ± 0.05	2.44 ± 0.06	2.44 ± 0.05	2.43 ± 0.05
Tree canopy score	1.46 ± 0.03	1.45 ± 0.03	1.52 ± 0.04	1.33 ± 0.03	1.32 ± 0.03	1.32 ± 0.03	1.32 ± 0.03
Understorey score	4.59 ± 0.16	4.47 ± 0.13	4.58 ± 0.15	4.04 ± 0.12	4.04 ± 0.12	4.05 ± 0.11	4.03 ± 0.11
Litter score	1.34 ± 0.03	1.30 ± 0.03	1.34 ± 0.03	1.19 ± 0.03	1.18 ± 0.03	1.18 ± 0.03	1.18 ± 0.03
Logs score	1.37 ± 0.03	1.35 ± 0.03	1.39 ± 0.04	1.25 ± 0.03	1.26 ± 0.03	1.25 ± 0.03	1.25 ± 0.03
Weeds score	3.48 ± 0.09	3.41 ± 0.09	3.49 ± 0.10	3.01 ± 0.08	3.01 ± 0.08	3.02 ± 0.08	3.01 ± 0.08
Recruitment score	2.41 ± 0.08	2.35 ± 0.07	2.43 ± 0.08	2.18 ± 0.07	2.18 ± 0.06	2.18 ± 0.06	2.18 ± 0.06

encing all components of the HHSS are those immediately related to (the probability) of (indigenous and non-native) tree cover (such as *NativeTreeProb* that appears in the root of the multi-target tree, and *TreeProb1HaRegionMean*, which appears in the roots of 5/7 single-target trees). It is interesting to note that this is the case also for the sub-components that do not depend directly on the presence of tree cover, e.g. *Weeds Score*. Following closely is *LandCover* (as modelled from satellite images), with dense forest cover (category 2) yielding high HHSS scores. Finally, climate plays an important role, with variables describing temperatures, rainfall and their variability appearing in most of the models.

5.2. Models with superior predictive performance

This sub-section presents and discusses the models obtained with the second scenario (see Section 4). Here, we compare linear regression, multi-target regression trees, regression trees, ensembles of multi-target regression trees and ensembles of regression trees to investigate the possible improvements in prediction performance (Tables 2 and 3) and computational efficiency (Table 4) that can be achieved by ensemble methods.

We present the predictive performance of the obtained models in terms of their correlation coefficient s and RMSEs. The results are presented with the corresponding confidence intervals, to show the stability of the used algorithms. Recall that 10 times 10-fold cross-validation was used to estimate the performance on unseen data. We can note that the confidence intervals are small. This is due to the size of the dataset (16,967 samples).

To check whether the differences in performance are of statistical significance, we used the corrected Friedman test for multiple hypothesis testing. To detect which algorithms perform significantly better or worse than the others we used the Nemenyi post hoc test. The result of the corrected Friedman test is that the difference in performance of these algorithms is statistically significant with a *p-value* smaller than 0.01. The results of the Nemenyi post hoc test for the RMSE comparison are presented in Fig. 6 with an average ranks diagram. On the axis the algorithms are plotted according to their average rank. The best performing algorithm is random forests with single-target regression trees, while the worst performing algorithm is the single-target regression tree. The critical distance is calculated for the significance level of 0.05.

The Nemenyi test shows that the performance of the ensemble methods (in terms of RMSE) is significantly better than the one of individual trees. The ensembles from both MTRTs and RTs are not significantly better than the single MTRT (at $p = 0.05$). However, the ensembles of MTRTs (both bagging and random forests) and the random forests of RTs are significantly better than linear regression and single-target regression trees. The difference in performance between MTRTs, RTs and linear regression is not statistically sig-

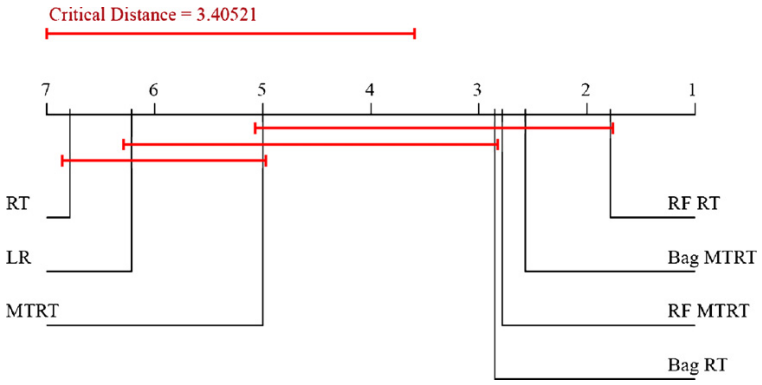


Fig. 6. Average ranks diagram for the applied algorithms (comparing by RMSE). Algorithms that do not differ significantly ($p = 0.05$) are connected with a line.

Table 4
Comparison of the time and size efficiency of the algorithms (LR—linear regression, MTRT—multi-target regression tree, RT—regression tree, Bag—bagging, RF—random forests).

	LR	MTRT	RT	Bag MTRT	Bag RT	RF MTRT	RF RT
Time (s)	8.06	7.18	36.18	430.94	2053.50	87.69	385.38
Size	332	345	4729	10,639.94	35,145.02	10,907.66	43,030.76

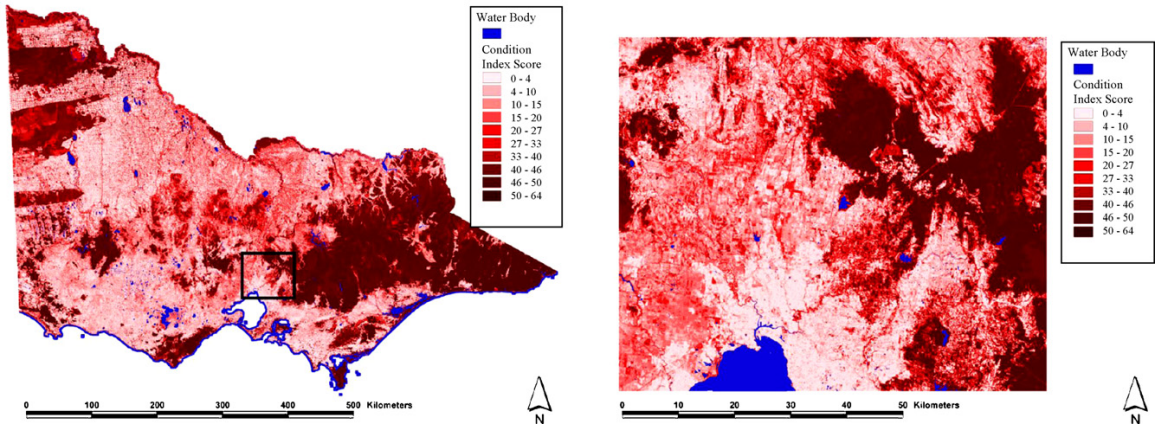


Fig. 7. Map of the condition of indigenous remnant vegetation in Victoria derived from the application of the random forests of MTRTs (left-hand side figure). The dark bordered rectangular inset refers to the area represented at higher resolution at the right-hand side figure.

nificant. Similar conclusions can be drawn if instead of the results for RMSE we consider the results for the correlation coefficient (Figure A1 from Appendix).

In addition, we compared the approaches by their time and size efficiency (Table 4). For the single-target scenarios (linear regression, regression trees, bagging of regression trees and random forests of regression trees) the time efficiency is calculated as the sum of the times used to learn a model for each target separately. The size of a linear regression model is the number of terms in the equation. The size of a MTRT is calculated as the number of nodes in the tree, while the size of a regression tree is the sum of the number of nodes in the trees over all targets. For bagging and random forests of multi-target regression trees, the size efficiency is the sum of size of the trees in the ensemble, while for the bagging and random forests of regression trees the size is the sum of the sizes of the ensembles for each target.

When comparing ensemble methods, the speed-up ratio of multi-target over single-target tree models remains high (4.5 on average), while the size of the multi-target tree models is about 0.25 of the size of single-target tree models. Multi-target regression consistently delivers models that have equally good predictive power, but are smaller and faster to learn (and apply). Linear regression has comparable time and size efficiency with multi-target regression models.

Overall, random forests of multi-target regression trees should be preferred, given that they improve the predictive performance and stability of multi-target trees in general, and are not as computationally expensive as bagging.

The spatially explicit map produced by the MTRT random forest ensemble, provides a subtle and accurate reflection of the condition of indigenous vegetation across the State of Victoria (Fig. 7). As we can see in the detailed inset, the modelled condition is finely resolved and nuanced, responding appropriately to local conditions, land use and land tenure. Application of the models allows for their further evaluation by experts familiar with local study areas. Such an evaluation is an ongoing process—but preliminary assessment indicates that the random forest MTRT is a robust model across a wide range of landscape, landuse and historical contexts.

6. Conclusions

In this work, we model the condition of remnant indigenous vegetation with machine learning techniques. 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. To model the multiple scores, we used two approaches: single-target and multi-target regression. With single-target regression we learn a model for each score separately, while with the multi-target regression we learn one model for all scores. The results show the advantages of multi-target over single-target regression: multi-target models have a smaller size and are faster to learn and apply. Also, there is no statistically significant difference in their predictive power.

We performed two sets of experiments. With the first set we were interested in knowledge extraction, and with the other we opted for models that have better predictive power. For knowledge extraction, we used pruned regression trees and pruned multi-target regression trees. 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.

Predictive power and efficiency was an imperative for the selection of the preferred model from the second set of experiments. In order to obtain models that have high predictive power we used unpruned regression trees, ensembles of regression trees, unpruned multi-target regression trees and ensembles of multi-target regression trees. Given the results of the statistical tests for the predictive

power, and the time and size efficiency, the random forests of multi-target regression trees should be preferred.

An important consideration of model utility is the spatial aspect at which the models are to be used and the specific purpose for which the model has been developed. The development of both single trees and ensembles of trees has highlighted the trade-off in model selection between complexity and predictive power on one hand and interpretability on the other. The pruned single tree based solutions to the prediction problem are transparent and facilitate almost immediate interpretation and qualitative evaluation by a range of users with varying degrees of understanding of the underlying learning algorithm. However, due to their simplicity, the predictions of single (pruned) trees as rendered by mapping produce generalized surfaces apparently devoid of the heterogeneity and subtlety of the real world. This may be a useful outcome if the objective is to produce a simple model. Conversely, due to the high predictive power, the ensemble models provide for the complexity and fine scale accuracy absent from the single trees, but are not readily interpretable to users.

It is apparent from this study that complex weighted metrics such as the habitat hectare index of vegetation condition can be modelled across extensive areas with some predictive confidence, using easily obtained remotely acquired data and provided adequate field data is collected. Such products can provide a 'snapshot' of the prevailing conditions and provide investment and decision support for natural resource managers.

We intend to extend out work in several directions. We hope to use new features that summarise relevant past and prevailing environmental disturbances and land uses, with a view to improving spatial models of vegetation condition, while realising some view of condition trajectory. In addition, we intend to develop spatially explicit models of both the untransformed and unweighted field measures that inform each of the components of the HHSS and the benchmark or reference values for these measures. Finally, we are interested in investigating the potential for implementing cost-sensitive learning to reflect heightened regulatory, planning or investment interest in particular geographic regions or particular index value ranges.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.ecolmodel.2009.01.037.

References

Andreasen, J.K., O'Neill, R.V., Noss, R., Slosser, N.C., 2001. Considerations for the development of a terrestrial index of ecological integrity. Ecological Indicators 1 (1), 21–35.

Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Machine Learning 36 (1–2), 105–139.

Botequila Leitão, A., Miller, J., Ahern, J., McGarigal, K., 2006. Measuring Landscapes: A Planner's Handbook. Island Press, Washington, DC.

Blockeel, H., De Raedt, L., Ramon, J., 1998. Top-down induction of clustering trees. In: 15th Int'l Conf. on Machine Learning, pp. 55–63.

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

Bradshaw, A., 1984. Ecological principles and land reclamation practice. Landscape Planning 11, 35–48.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Wadsworth, Belmont.

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

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

Cortina, J., Maestre, F.T., Vallejo, R., Baeza, M.J., Valdecantos, A., Pérez-Devesa, M., 2006. Ecosystem structure, function, and restoration success: are they related? Journal for Nature Conservation 14 (3–4), 152–160.

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

Dietterich, T., 2000. Ensemble methods in machine learning. In: Proc. of the 1st Int'l Workshop on Multiple Classifier Systems, vol. 1857 of LNCS. Springer, Berlin, pp. 1–15.

Džeroski, S., Gjorgjioski, V., Slavkov, I., Struyf, J., 2007. Analysis of time series data with predictive clustering trees. LNCS 4747, 63–80.

Eyre, T.J., Kelly, A.L., Neldner, V.J., 2006. BioCondition: A Terrestrial Vegetation Condition Assessment Tool for Biodiversity in Queensland. Field Assessment Manual. Version 1. 5. B. S. u. Environmental Protection Agency, Brisbane.

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

Gibbons, P., Briggs, S.V., Ayers, D., Seddon, J., Doyle, S., Cosier, P., McElhinny, C., Pelly, V., Roberts, K., 2009. An operational method to assess impacts of land clearing on terrestrial biodiversity. Ecological Indicators 9, 26–40.

Gibbons, P., Zerger, A., Jones, S., Ryan, P., 2006. Mapping vegetation condition in the context of biodiversity conservation. Ecological Management and Restoration 7, S1–S2.

Guisan, A., Thuiller, W., 2005. Predicting species distribution: offering more than simple habitat models. Ecology Letters 8 (9), 993–1009.

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

Houlder, D.J., Hutchinson, M.F., Nix, H.A., McMahon, J.P., 2000. NUCLIM User Guide, Version 5.1. Centre for Resource and Environmental Studies, Australian National University, Canberra, Australia.

Iman, R.L., Davenport, J.M., 1980. Approximations of the critical region of the Friedman statistic. Communications in Statistics, 571–595.

Knight, A.T., Cowling, R.M., Campbell, B.M., 2006. An operational model for implementing conservation action. Conservation Biology 20 (2), 408–419.

Kocev, D., Vens, C., Struyf, J., Džeroski, S., 2007. Ensembles of multi-objective decision trees. In: Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Proceedings, volume 4701 of LNCS, pp. 624–631.

Li, H., Wu, J., 2004. Use and misuse of landscape indices. Landscape Ecology 19 (4), 389–399.

McGarigal, K., Cushman, S.A., Neel, M.C., 2002. FRAGSTATS: Spatial Pattern Analysis Program for Categorical Maps. University of Massachusetts, Amherst (Computer software program produced by the authors at the University of Massachusetts, Amherst).

Margules, C., Pressey, B., 2000. Systematic conservation planning. Nature 405, 243–253.

Moilanen, A., 2007. Landscape zonation, benefit functions and target-based planning: unifying reserve selection strategies. Biological Conservation 134 (4), 571–579.

Nemenyi, P.B., 1963. Distribution-free multiple comparisons. PhD Thesis. Princeton University, Princeton, NY, USA.

Oliver, I., 2004. A framework and toolkit for scoring the biodiversity value of habitat, and the biodiversity benefits of land use change. Ecological Management & Restoration 5 (1), 75–78.

Parkes, D., Lyon, P., 2006. Towards a national approach to vegetation condition assessment that meets government investors needs: a policy perspective. Ecological Management and Restoration 7, S3–S5.

Parkes, D., Newell, G., Cheal, D., 2003. Assessing the quality of native vegetation: the habitat hectares approach. Ecological Management and Restoration 4, S29–S38.

Parkes, D., Newell, G., Cheal, D., 2004. The development and raison *d'etre* of 'habitat hectares': a response to McCarthy et al. (2004). Ecological Management and Restoration 5 (1), 28–29.

Rouget, M., Cowling, R.M., Lombard, A.T., Knight, A.T., Kerley, G.I.H., 2006. Designing large-scale conservation corridors for pattern and process. Conservation Biology 20 (2), 549–561.

Scholes, R.J., Biggs, R., 2005. A biodiversity intactness index. Nature 434, 45–49.

Struyf, J., Džeroski, S., 2006. Constraint based induction of multi-objective regression trees. In: Knowledge Discovery in Inductive Databases, 4th Int'l Workshop, KDID'05, Revised, Selected and Invited Papers, vol. 3933 of LNCS, pp. 222–233.

Tucker, C.J., 1979. Red and photographic infrared linear combinations for monitoring vegetation. Remote Sensing of Environment 8 (2), 127–150.

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

Witten, I.H., Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition. Morgan Kaufmann, San Francisco.

APPENDIX

Table A1. Table describing the descriptive (explanatory, independent) variables

Environmental Variables	Brief Description	Pixel Resolution
ThorPot07	Radiometric Data – Ratio of the radioelement count of Thorium and the radioelement count of Potassium. Sourced from Various Australian State Agencies	50 m resampled to 30 m
ThorInvPot07	Radiometric Data – Ratio of the inverse radioelement count of Thorium and the radioelement count of Potassium. Sourced from Various Australian State Agencies	50 m resampled to 30 m
B1_89_05	Band 1 (Blue-green reflectance 0.45-0.52 micrometers) Landsat 7 TM Median value years 1989 – 2005	30 m
B2_89_05	Band 2 (Green reflectance 0.52-0.60 micrometers) Landsat 7 TM Median value years 1989 – 2005	30 m
B3_89_05	Band 3 (Red reflectance 0.63-0.69 micrometers) Landsat 7 TM Median value years 1989 – 2005	30 m
B4_89_05	Band 4 (Near-infrared reflectance 0.76-0.90 micrometers) Landsat 7 TM Median value years 1989 – 2005	30 m
B5_89_05	Band 5 (Mid-infrared reflectance 1.55-1.75 micrometers) Landsat 7 TM Median value years 1989 – 2005	30 m
B7_89_05	Band 7 (Mid-infrared reflectance 2.08-2.35 micrometers) Landsat 7 TM Median value years 1989 – 2005	30 m
Ndvi_89_05	Mean Normalised Difference Vegetation Index derived from LANDSAT 7 TM of years 1989 – 2005	30 m
Ndwi_89_05_Mean	Mean Normalised Difference Wetness Index derived from LANDSAT 7 TM of years 1989 – 2005	30 m
Ndwi_89_05_StdError	Standard Error of Normalised Difference Wetness Index derived from LANDSAT 7 TM of years 1989 – 2005	30 m
B3_98_05_10ha_Mean	Mean value across a 10 hectare neighbourhood of cells calculated from the surface - Mean value for Band 3 (Blue-green reflectance 0.45-0.52 micrometers) Landsat 7 TM years 1998 – 2005	30 m
B4_98_05_10ha_Mean	Mean value across a 10 hectare neighbourhood of cells calculated from the surface - Mean value for Band 4 (Near-infrared reflectance 0.76-0.90 micrometers) Landsat 7 TM years 1998 – 2005	30 m
B5_98_05_10ha_Mean	Mean value across a 10 hectare neighbourhood of cells calculated from the surface - Mean value for Band 5 (Mid-infrared reflectance 1.55-1.75 micrometers) Landsat 7 TM years 1998 – 2005	30 m
B6_98_05_10ha_Mean	Mean value across a 10 hectare neighbourhood of cells calculated from the surface - Mean value for Band 7 (Mid-infrared reflectance 2.08-2.35 micrometers) Landsat 7 TM years 1998 – 2005	30 m
Nvdi_98_05_10haMean	Mean Normalised Difference Vegetation Index derived from LANDSAT 7 TM of years 1998 – 2005	30 m
B3_98_05_10ha_StdDev	Standard Deviation across a 10 hectare neighbourhood of cells calculated from the surface - Mean value for Band 3 (Blue-green reflectance 0.45-0.52 micrometers) Landsat 7 TM years 1998 – 2005	30 m
B4_98_05_10ha_StdDev	Standard Deviation across a 10 hectare neighbourhood of cells calculated from the surface - Mean value for Band 4 (Near-infrared reflectance 0.76-0.90 micrometers) Landsat 7 TM years 1998 – 2005	30 m
B5_98_05_10ha_StdDev	Standard Deviation across a 10 hectare neighbourhood of cells calculated from the surface - Mean value for Band 5 (Mid-infrared reflectance 1.55-1.75 micrometers) Landsat 7 TM years 1998 – 2005	30 m
B6_98_05_10ha_StdDev	Standard Deviation across a 10 hectare neighbourhood of cells calculated from the surface - Mean value for Band 7 (Mid-infrared reflectance 2.08-2.35 micrometers) Landsat 7 TM years 1998 – 2005	30 m
RoadDensity5K	Density of Roads in a 5 kilometre radius - line count	30 m
TempRange	Annual range in temperature (ºC) between minimum temperature of coldest period of the year and the maximum temperature of the warmest period of the year. Developed using ANUCLIM (Houlder et al. 2000)	100m resampled to 30m
MaxTempWarmestP	The highest temperature (ºC) of any weekly maximum temperature.	100m resampled to 30m

Table A1 (ctd.). Table describing the descriptive (explanatory, independent) variables

Environmental Variables	Brief Description	Pixel Resolution
AnnualRain	Mean Annual Rainfall Surface (mm) developed using ANUCLIM (Houlder et al. 2000)	100m resampled to 30m
NetRainfall	Mean Annual Rainfall (mm) (from ANUCLIM model) less Mean Annual Evaporation (mm) (from ANUCLIM model)	100m resampled to 30m
NetRainfallStdev	The Standard Deviation of Monthly Net Mean Rainfall (Monthly Net mean Rainfall is the mean Monthly Rainfall (mm) less the Mean Monthly Evaporation). Monthly means were developed using ANUCLIM (Houlder et al. 2000)	100m resampled to 30m
TWIx1000	Topographic Wetness Index a compound terrain attribute (*sensu* Bevan and Kirby 1979) implemented using the Shuttle Radar Topography Mission (SRTM) Digital Elevation Model and TOPOCROP Version 2.1 (Schmidt 2002)	100m resampled to 30m
Rad_Direct	Direct Solar Radiation (Watts m2 per year). Derived from Shuttle Radar Topography Mission (SRTM) Digital Elevation Model using The Solar Analyst 1.0 (Fu and Rich 2000)	100m resampled to 30m
LandCover	Categorical variable 10 Landcover classes derived from K-means clustering of median satellite imagery captured between 1989 and 2005	30 m
NativeTreeProb	An Artificial Neural Network Model of the probability of a lack of tree cover for Victoria trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
TreeProb1Ha_RegionMean	The mean result for a 1 hectare neighbourhood for the probability of a lack of tree cover for Victoria (see NativeTreeProb). Trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
TreeProb10ha_RegionMean	The mean result for a 10 hectare neighbourhood for the probability of a lack of tree cover for Victoria (see NativeTreeProb). Trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
TreeProb1Ha_RegionStdDev	The standard deviation across a 1 hectare neighbourhood for the probability of a lack of tree cover for Victoria (see NativeTreeProb). Trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
TreeProb10ha_RegionStdDev	The standard deviation across a 10 hectare neighbourhood for the probability of a lack of tree cover for Victoria (see NativeTreeProb). Trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
Pre1750TreeDensity	An Artificial Neural Network model of the density of tree cover across south eastern-Australia prior to European invasion in the easly 19th century. The model was trained and validated using tree cover sampling along roads and other parts of the landscape in which the tree cover has been relatively undisturbed by subsequent land use.	100m resampled to 30m
NativeGrassProb	An Artificial Neural Network Model of the probability of native grassland cover for Victoria trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
GrassProb1Ha_RegionMean	The mean result for a 1 hectare neighbourhood for the probability of native grassland cover for Victoria (see NativeGrassProb). Trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
GrassProb1Ha_RegionStdDev	The standard deviation across a 1 hectare neighbourhood for the probability of native grassland cover for Victoria (see NativeGrassProb). Trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
GrassProb10ha_RegionMean	The mean result for a 1 hectare neighbourhood for the probability of native grassland cover for Victoria (see NativeGrassProb). Trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m
GrassProb10ha_RegionStdDev	The standard deviation across a 10 hectare neighbourhood for the probability of native grassland cover for Victoria (see NativeGrassProb). Trained using Landsat 7 TM chronosequence and Spot 4 panchromatic imagery.	30 m

Table A2. Basic statistics for the habitat hectares site score field data.

	Minimum	Maximum	Mean value	Standard Deviation
Large Tree Score	0	10	2.82	3.36
Tree Canopy Score	0	5	2.46	2.23
Understorey Score	0	25	8.50	7.16
Litter Score	0	5	3.00	2.04
Logs Score	0	5	1.88	2.08
Weeds Score	0	15	6.97	6.14
Recruitment Score	0	10	3.33	3.26

Table 3A. Ranking of the algorithms by the RMSE for the Friedman test. Outcome of Freidman test is that with p-value less than 0.01 the difference in the performance is statistically significant.

Target	**LR**	**MTRT**	**RT**	**BagMTRT**	**Bag RT**	**RF MTRT**	**RF RT**
Large Tree Score	6	5	7	1.5	3.5	3.5	1.5
Tree Canopy Score	6	5	7	4	2	2	2
Understorey Score	7	5	6	2.5	2.5	4	1
Litter Score	6.5	5	6.5	4	2	2	2
Logs Score	6	5	7	2	4	2	2
Weeds Score	6	5	7	1.5	3.5	3.5	1.5
Recruitment Score	6	5	7	2.5	2.5	2.5	2.5
Average Ranks	6.21	5.00	6.79	2.57	2.86	2.79	1.79

Table 4A. Ranking of the algorithms by the Correlation Coefficient for the Friedman test. Outcome of Freidman test is that with p-value less than 0.01 the difference in the performance is statistically significant.

Target	**LR**	**MTRT**	**RT**	**BagMTRT**	**Bag RT**	**RF MTRT**	**RF RT**
Large Tree Score	6	5	7	2.5	2.5	2.5	2.5
Tree Canopy Score	5.5	5.5	7	4	2	2	2
Understorey Score	6.5	5	6.5	2.5	2.5	2.5	2.5
Litter Score	6.5	5	6.5	4	2	2	2
Logs Score	6.5	5	6.5	2.5	2.5	2.5	2.5
Weeds Score	7	5.5	5.5	2.5	2.5	2.5	2.5
Recruitment Score	6.5	5	6.5	3	3	3	1
Average Ranks	6.36	5.14	6.50	3.00	2.43	2.43	2.14

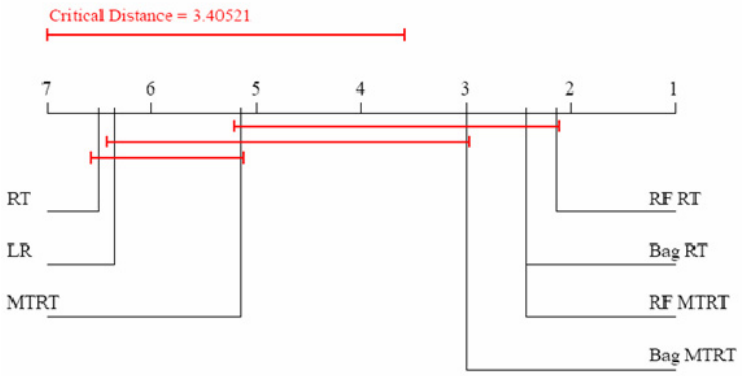


Figure A1. Average ranks diagram for the applied algorithms (comparing by correlation coefficient). Algorithms that do not differ significantly ($p-value = 0.05$) are connected with a line.

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).

Hierarchical Annotation of Medical Images

Ivica Dimitrovski[a,b,*], Dragi Kocev[a], Suzana Loskovska[b], Sašo Džeroski[a]

[a] *Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[b] *Department of Computer Science and Computer Engineering, Faculty of Electrical Engineering and Information Technologies, Rugjer Boshkovikj bb, 1000 Skopje, Republic of Macedonia*

Abstract

We present a hierarchical multi-label classification (HMC) system for medical image annotation. HMC is a variant of classification where an instance may belong to multiple classes at the same time and these classes/labels are organized in a hierarchy. Our approach to HMC exploits the annotation hierarchy by building a single predictive clustering tree (PCT) that can simultaneously predict all annotations of an image. Hence, PCTs are very efficient: a single classifier is valid for the hierarchical semantics as a whole, as compared to other approaches that produce many classifiers, each valid just for one given class. To improve performance, we construct ensembles of PCTs. We evaluate our system on the IRMA database that consists of X-ray images. We investigate its performance under a variety of conditions. To begin with, we consider two ensemble approaches, bagging and random forests. Next, we use several state-of-the-art feature extraction approaches and combinations thereof. Finally, we employ two types of feature fusion, i.e., low- and high-level fusion. The experiments show that our system outperforms the best-performing approach from the literature (a collection of SVMs, each predicting one label at the lowest level of the hierarchy), both in terms of error and efficiency. This holds across a range of descriptors and descriptor combinations, regardless of the type of feature fusion used. To stress the generality of the proposed approach, we have also applied it for automatic annotation of a large number of consumer photos with multiple annotations organized in semantic hierarchy. The obtained results show that this approach is general and easily applicable in different domains, offering state-of-the-art performance.

Keywords: Automatic Image Annotation, Hierarchical Multi-Label Classification, Predictive Clustering Trees, Feature Extraction from Images

1. Introduction

Digital imaging in medicine is in constant growth due to the increasing availability of imaging equipment in hospitals. Average-sized radiology departments now produce several tera-bytes of data annually. This prompts for efficient systems for image annotation, storage, retrieval and mining. Typically, medical image databases are accessed via textual information through the

*Corresponding author (telephone: +389 2 3099 159)
Email addresses: ivicad@feit.ukim.edu.mk (Ivica Dimitrovski), Dragi.Kocev@ijs.si (Dragi Kocev), suze@feit.ukim.edu.mk (Suzana Loskovska), Saso.Dzeroski@ijs.si (Sašo Džeroski)

Preprint accepted for publication in Pattern Recognition

standard Picture Archiving and Communication System (PACS) [1], [2]. PACS integrates imaging modalities and interfaces with hospital and departmental information systems to manage storage and distribution of images to medical personnel, researchers, clinics, and imaging centers. An important requirement of PACS is the provision of an efficient search function to access the required images.

An universal format for PACS image storage and retrieval is the Digital Imaging and Communications in Medicine (DICOM) standard [3]. DICOM is a well known standard for handling, storing, printing, and transmitting information in medical imaging. The DICOM header contains tags to decode the body part examined, the patient position and the acquisition modality. Some of the tags are automatically set by the digital system according to the imaging protocol used to capture the pixel data. Other part of the tags are set manually by the physicians or radiologists during the routine documentation. This procedure cannot always be considered very reliable, since frequently happens that some entries are either missing, false, or do not describe the anatomic region precisely [4]. Furthermore, manual annotation of images is an expensive and time-consuming procedure, especially given the large and constantly growing databases of medical images. Thus, completely automated categorization in terms of DICOM tags is currently not possible, but is highly desirable.

Automatic image annotation or image classification is an important step in image retrieval. In the medical domain, using information directly extracted from images to annotate/categorize them will improve the quality of image annotation in particular, and more generally the quality of patient care. Properly classified medical image data can help medical professionals in fast and effective access to data in their teaching, research, training, and diagnostic problems. The results of the classification step can also be used for multilingual image annotation as well as for DICOM header correction [5].

Automatic image annotation can be used for retrospective annotation (pre DICOM). It can also be used as help for human annotators (i.e., radiologists), where the annotations that are suggested by the system are corrected/verified/confirmed by the human annotator. The limits of performance of an automated annotation system that learns from example images annotated by humans, is the rate/probability of operator error/agreement of annotators.

Automatic image annotation uses a computer system which automatically assigns metadata in the form of captions or keywords to a digital image. Typically, image analysis first extracts feature vectors. Then, together with the training annotations, they are used by a machine learning algorithm to learn to automatically assign annotations. The performance of the computer system largely depends on the availability of strongly representative visual features, able to characterize different visual properties of the images, and the use of effective algorithms for training classifiers for automatic image annotation.

A single image may contain different meanings organized in hierarchical semantics: hence, hierarchical multi-label classification (HMC) is strongly recommended for obtaining multi-label annotations. The task of multi-label classification is to assign multiple labels to each image. The assigned labels are a subset of a previously defined set or hierarchy of labels. HMC is used in various domains [6], such as text classification, scene and video classification, medical imaging and biological applications. One of the main issues involved in multi-label classification is the importance of detecting and incorporating the connections between the labels into the process of assigning multiple labels. A second and related issue is the additional complexity involved in learning multi-label classifiers, as compared to learning single-label classifiers.

In this paper, we present a HMC system for medical image annotation. This system consists of the two standard parts of image annotation systems, i.e., processing (feature extraction) and

classification of images. The image processing part uses state-of-the-art approaches to convert an image to a set of numerical features extracted directly from the pixel values. The image classification part, which labels and groups the images, contains the main novelty of our approach: The labels can be organized in a hierarchy and an image can be labeled with more than one label (an image can belong to more than one group).

First, we generate four different types of descriptors suitable for X-Ray medical images: raw pixel representation (RPR) [7], local binary patterns (LBP) [8], edge histogram descriptors (EHD) [9], and scale-invariant feature transform (SIFT) [10]. The features are generated using the medical X-ray images from the ImageCLEF2009 medical image annotation task [5]. Next, we use these features together with the annotations to train the classifiers. In particular, we use ensembles (bags and random forests) of PCTs for HMC and SVMs for single-label classification, the most widely used classifier in the area of image annotation. At the end, we assess the predictive performance of the classifiers using the hierarchical error measure (HEM) from ImageCLEF [5] and overall recognition rate (RR), commonly used for assessing the predictive performance over the database we use.

The main question that we address in our research is whether exploiting the semantic knowledge about the inter-class relationships among the image labels (organized in a hierarchical structure) can improve the predictive performance of a system for automatic image annotation. To this end, we compare the predictive performance of the ensembles of PCTs for HMC (that predict all labels simultaneously) to that of SVMs (each of them predicting a single label). We do this across all feature extraction techniques, thus evaluating the different feature extraction techniques and their use in HMC of medical X-ray images. Moreover, we investigate whether (and which type of) combination of feature extraction techniques yields better predictive performance. We consider low level (LL) and high level (HL) feature fusion/combination schemes [7].

To emphasize the generality of our approach, we have also tested it on the database of general images from the ImageCLEF@ICPR 2010 photo annotation task [11]. The images in this database are annotated with 53 visual concepts organized in a classification scheme with hierarchical structure, which we used to build ensembles of PCTs for HMC as classifiers. The 53 concepts include abstract categories (like partylife), the time of day (like day or night), persons (like no person visible, small or big group) and quality (like blurred or underexposed). A complete overview of the task is given by Nowak [11].

The remainder of the paper is organized as follows. In Section 2, we give an overview of related work. Section 3 introduces predictive clustering trees and their use for HMC. Section 4 describes the techniques for feature extraction from images. In Section 5, we explain the experimental setup for annotating medical images. The obtained results and a discussion thereof are given in Section 6. Section 7 describes the experiments in annotation of general images, as well as their results. Section 8 concludes the paper and points out some directions for further work.

2. Related work

In this section, we present some classification methods that are or can be used for image annotation. We begin by presenting the methods that are most widely used by the image annotation community. We then present some recent machine learning methods that can be used for hierarchical image annotation and discuss their relation to the method we propose.

Regardless of the number of visual concepts that have to be learned and their mutual connections, most of the present systems for annotation of general images (and medical images in

particular) learn a separate model for each visual concept (label), i.e., they treat the classes as completely separate and independent (both visually and semantically). This means that multi-label classification problems are transformed into several binary classification problems. For example, the methods with high predictive performance at recent challenges/competitions in detection and annotation tasks (such as the PASCAL Visual Object Classes challenge [12], the ImageCLEF medical image annotation task [13], [5] and the ImageCLEF visual concept detection and annotation task [14]) perform multi-label classification by building binary classifiers for each label. The instances associated with particular label are in one class and the rest are in another class. For solving the binary classification problems, is common to use a SVM with a χ^2 kernel [15]. This means that the increase of the number of labels used for annotation will linearly increase the complexity of such an approach.

To deal with a large number of labels/classes, many approaches combine binary classifiers using class hierarchies [16], [17]. This results in a logarithmic increase of complexity as the number of labels increases. The class hierarchies can be automatically constructed through analysis of visual similarities: this can proceed top-down by recursive partitioning of the set of classes [18] or bottom-up by agglomerative clustering [19]. The hierarchies could also be found by exhaustive search or random sampling followed by cross-validation [20].

An alternative method for automatic construction of hierarchies is to query an external semantic network with class labels [17]. Since semantic networks model concepts and relations between them, a subgraph in the form of a hierarchy can be easily extracted. Such an approach allows to incorporate prior knowledge about object identity into the visual recognition system. Our approach to automatic image annotation is based on this idea. We exploit the semantic knowledge about the inter-class relationships among the image labels organized in a hierarchical structure. We build one classifier that can simultaneously predict all annotations of an image, instead of building one binary classifier for each node in the hierarchy.

Another popular approach to image annotation is TagProp [21]. TagProp is a discriminatively trained nearest neighbor model. Tags of test images are predicted using a weighted nearest-neighbor model to exploit labeled training images. Neighbor weights are based on neighbor rank or distance. TagProp allows the integration of metric learning by directly maximizing the log-likelihood of the tag predictions in the training set. However, in a recent study, Mensink et al.[22] showed that per-label-trained linear SVM classifiers outperform TagProp.

So far, we presented the most widely used methods for image annotation and concluded that SVMs with a χ^2 kernel trained per label are the preferred method by the image annotation community. In the remainder of this section, we discuss recent machine learning methods that can be used in the context of hierarchical image annotation: SVMs for structured prediction, PCTs, ensembles of PCTs and ensembles of SVMs. To begin with, SVMs for predicting structured outputs can be considered as classifiers for hierarchical image annotation. Unfortunately, the most well-known system for predicting structured outputs based on SVMs, SVMstruct [23], does not offer facilities for HMC. Those that do are very recent [24], have high computational complexity and are not used by the image annotation community.

We can also apply PCTs for HMC to the task of hierarchical image annotation. Vens et al. [25] describe in detail PCTs that are able to perform hierarchical multi-label classification and perform extensive experimental evaluation on functional genomics datasets. They show that PCTs for HMC achieve very good predictive performance and are very efficient.

Ensemble methods are a popular approach that generates a set of classifiers (called base classifiers) and combine their predictions into a single prediction [26]. Many practical and theoretical studies show that ensembles achieve high predictive performance and lift the predictive

performance of a single classifier [27, 28]. This is especially true for base classifiers that are unstable, i.e.,can change drastically due to small changes in the training data: Decision trees are typical example of unstable classifiers. Having this in mind, we extend the PCT framework in the context of ensemble learning, i.e., we construct ensembles of PCTs for HMC. We apply this approach to hierarchical image annotation: The ensembles of PCTs for HMC achieve better predictive performance than a single PCT and can be constructed efficiently.

Given that SVMs are the most widely used machine learning approach to image annotation, and that ensembles improve the performance of individual classifiers, one might also consider ensembles of SVMs. However, SVMs are relatively stable classifiers and less likely to benefit from an ensemble extension. Consequently, there is much less community consensus on whether and how ensembles of SVMs should be constructed (as compared to ensembles of decision trees). Evgeniou et al. [29] performed theoretical and empirical evaluation of ensembles from SVMs. The main finding in their study is that a single SVM classifier with tunned parameters performs similar to an ensemble of SVM classifiers. On the other hand, Valentini and Dietterich [30] and Wang et al. [31] show that ensembles of SVMs do lift the predictive performance of a single SVM. They also discuss practical issues in constructing such an ensemble and obtaining a prediction for unseen instance. A recent study by Ting and Zhu [32] proposed a boosting algorithm that uses a hybrid between decision trees and SVMs as base classifier. Their findings reveal that such an ensemble has better predictive performance than a single SVM and it is efficient. However, all these studies were performed in the context of binary or multi-class classification and their extension for the task of HMC is not straightforward. The base classifiers would need to be either SVMs for HMC or collections of SVMs as discussed above. In addition, ensembles of SVMs are not in widespread use in the image annotation community.

3. Ensembles of PCTs for HMC

This section presents our approach for building ensembles of PCTs. We first present the task of HMC. Next, we describe the predictive clustering trees and their instantiation for the task of HMC. Finally, we present ensembles of PCTs for HMC and methods for building them.

The development of this approach is motivated by the fact that ensembles lift the predictive performance of a single predictive model. This is well known in the case when the single predictive model is a classification or a regression tree. However, it is not obvious that the lift carries over to PCTs for HMC. When the base classifiers are decision trees, Bauer and Kohavi [33] conclude that the increase in performance is related to the trees being unpruned, i.e., overfitting. On the other hand, Blockeel et al. [34] state that PCTs for HMC overfit less as compared to individual trees for each class in the hierarchy. Having in mind these two conflicting influences, it is not obvious whether an ensemble of PCTs will significantly increase the predictive performance of a single PCT. Hence, this is an interesting issue to investigate. A further motivation for our study is provided by the fact that PCTs for HMC (and potentially ensembles thereof) are efficient to construct and perform well, yet their use in the context of hierarchical image annotation has so far not been investigated.

3.1. The task of HMC

Hierarchical multi-label classification is a variant of classification where (1) a single example may belong to multiple classes at the same time and (2) the possible classes are organized 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. For a more detailed overview of the possible application areas we refer the reader to Silla and Freitas[6].

image	features/descriptors						annotations/labels
	|	—	/	\	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
…	…	…	…	…	…	…	…

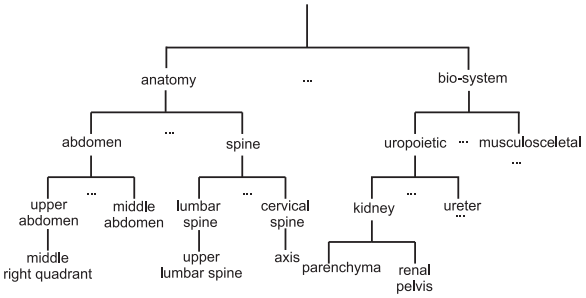


Figure 1: An example task of HMC in a medical domain. The table (on the left-hand side) contains a set of images with their visual descriptors and annotations. The annotations are part of the IRMA [35] hierarchical classification scheme (of which a small part is shown on the right hand side).

In medical image classification, the application domain on which we focus, an important problem is the development of an automatic image annotation system, which can specify the image modality, body orientation, body region, or the biological system examined. In this domain, the predefined set of labels might be organized in a semantic hierarchy, such as the one shown in Fig. 1. Each image is represented with: (1) a set of descriptors (in this example, the descriptors are histograms of five types of edges encountered in the image) and (2) a set of labels/annotations. A single image can be annotated with multiple labels at different levels of the predefined hierarchy.

For example, the image in the second row of the table in Fig. 1 has two labels, middle abdomen and renal pelvis, listed explicitly. Note that this image is also implicitly labeled with the labels: anatomy, abdomen, kidney, uropoietic and bio-system. These labels are all ancestors of the explicitly listed labels in the given hierarchy.

The data, as presented in the table in the left-hand side of Fig. 1, constitute a data set for HMC. This set can be used by a machine learning algorithm to train a classifier for HMC. For images in the testing set only the descriptors are given and no *a priori* annotations.

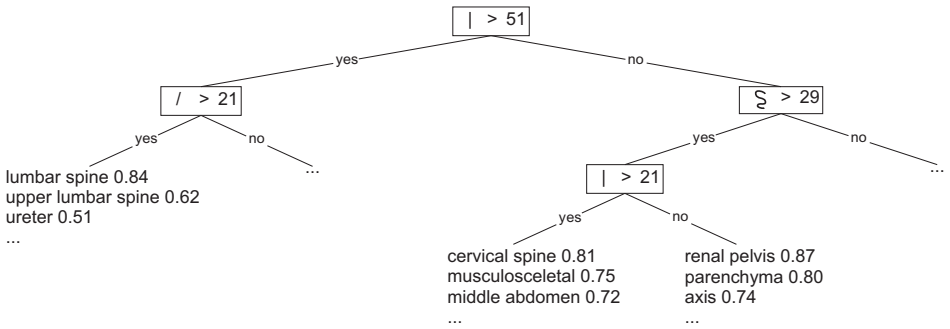


Figure 2: An example of a predictive clustering tree constructed using the descriptors from Fig. 1. The internal nodes contain tests on the descriptors, while the leafs store the probabilities that an image is annotated with a given label from the hierarchy.

3.2. Predictive clustering trees

Predictive Clustering Trees (PCTs) [36] [1] generalize decision trees [37] and can be used for a variety of learning tasks including different types of prediction and clustering. The PCT framework views a decision tree as a hierarchy of clusters: the top-node of a PCT corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The leaves represent the clusters at the lowest level of the hierarchy and each leaf is labeled with its cluster's prototype (prediction). Note that the hierarchical structure of the PCT (Fig. 2) does not necessary reflect the hierarchical structure of the annotations (Fig. 1).

PCTs are built with a greedy recursive top-down induction (TDI) algorithm, similar to that of C4.5 [38] or CART [37]. The learning algorithm starts by selecting a test for the root node. Based on this test, the training set is partitioned into subsets according to the test outcome. This is recursively repeated to construct the subtrees. The partitioning process stops when a stopping criterion is satisfied (e.g., the number of records in the induced subsets is smaller than some predefined value; the length of the path from the root to the current subset exceeds some predefined value etc.). In that case, the prototype is calculated and stored in a leaf.

One of the most important steps in the TDI algorithm is the test selection procedure. For each node, a test is selected by using a heuristic function computed on the training examples. The goal of the heuristic is to guide the algorithm towards small trees with good predictive performance. The heuristic used in this algorithm for selecting the attribute tests in the internal nodes is the reduction in variance caused by partitioning the instances, where the variance $Var(S)$ is defined by (Equation 1). Maximizing the variance reduction maximizes cluster homogeneity and improves predictive performance.

The main difference between the algorithm for learning PCTs and an algorithm for learning decision trees (such as C4.5 [38] and CART [37]) is that 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 [39], [40], prediction of time-series [41] and hierarchical-multi label classification [25]. In this article, we focus on the last of these tasks.

3.3. PCTs for hierarchical multi-label classification

To apply PCTs to the task of HMC, the example labels are represented as vectors with Boolean components. Components in the vector correspond to labels in the hierarchy traversed in a depth-first manner. The i-th component of the vector is 1 if the example belongs to class c_i and 0 otherwise. If $v_i = 1$, then $v_j = 1$ for all v_j's on the path from the root to v_i.

The variance of a set of examples (S) is defined as the average squared distance between each example's label v_i and the mean label $\bar{v}$ of the set, i.e.,

$$Var(S) = \frac{\sum_i d(v_i, \bar{v})^2}{|S|} \tag{1}$$

[1] The PCT framework is implemented in the CLUS system, which is available at `http://www.cs.kuleuven.be/~dtai/clus`.

The higher levels of the hierarchy are more important: an error at the upper levels costs more than an error at the lower levels. Considering this, a weighted Euclidean distance is used:

$$d(v_1, v_2) = \sqrt{\sum_i w(c_i)(v_{1,i} - v_{2,i})^2} \quad (2)$$

where $v_{k,i}$ is the i'th component of the class vector v_k of an instance x_k, and $w(c_i)$ are the class weights. The class weights decrease with the depth of the class in the hierarchy, $w(c_i) = w_0 \cdot w(c_j)$, where c_j is the parent of c_i. Each leaf in the tree stores the mean $\bar{v}$ of the vectors of the examples that are sorted into that leaf (Fig. 2). Each component of $\bar{v}$ is the proportion of examples $\bar{v}_i$ in the leaf that belong to class c_i. An example arriving in the leaf can be predicted to belong to class c_i if $\bar{v}_i$ is above some threshold t_i. The threshold can be chosen by a domain expert.

The PCTs are also extended for predicting hierarchies organized as directed acyclic graphs (DAGs). In this case, the depth of a class is not unique as classes do not have single path from the hierarchy's root. To resolve this issue, Vens et al. [25] suggest four aggregation schemes of the possible paths from the top-node to a given class: average, maximum, minimum and sum. After an extensive experimental evaluation, they recommend to use the average as aggregation function. For a detailed description of PCTs for HMC we refer the reader to Vens et al. [25]. Next, we explain how PCTs are used in the context of an ensemble classifier, in order to further improve the performance of PCTs.

3.4. Ensemble methods

An ensemble is a set of (base) classifiers. A new example is classified by the ensemble by combining the predictions of the member classifiers. The predictions can be combined by taking the average (for regression tasks), the majority vote (for classification tasks) [42],[43], or more complex combinations.

We use PCTs for HMC as base classifiers. Averaging is applied to combine the predictions of the different trees: the leaf's prototype is the proportion of examples of different classes that belong to it. Just like for the base classifiers, a threshold should be specified to make a prediction.

We consider two ensemble learning techniques that have primarily been used in the context of decision trees: bagging and random forests. Bagging [42] constructs the different classifiers by making bootstrap replicates of the training set and using each of these replicates to construct one classifier. Each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set, until a number of instances is obtained equal to the size of the training set. Bagging is applicable to any type of learning algorithm.

A random forest [43] is an ensemble of trees, obtained both by bootstrap sampling, and by randomly changing the feature set during learning. More precisely, at each node in the decision tree, a random subset of the input attributes is taken, and the best feature is selected from this subset (instead of the set of all attributes). The number of attributes that are retained is given by a function f of the total number of input attributes x (e.g., $f(x) = x$, $f(x) = \sqrt{x}$, $f(x) = \lfloor \log_2 x \rfloor + 1$, ...). By setting $f(x) = x$, we obtain the bagging procedure.

4. Feature extraction from images

Collections of medical images can contain various images obtained using different imaging techniques. Different feature extraction techniques are able to capture different aspects of an

image (e.g., texture, shapes, color distribution...). In this study, we focus on X-ray images, hence, we use texture (LBP and EHD) and local (SIFT) features as most promising for describing X-ray images [5],[44].

Texture is especially important, because it is difficult to classify medical images using shape or gray level information. Effective representation of texture is needed to distinguish between images with equal modality and layout. Local image characteristics are fundamental for image interpretation: while global features retain information on the whole image, the local features capture the details. They are thus more discriminative concerning the problem of inter and intra-class variability, an open challenge in automatic annotation of medical images [7].

4.1. Raw pixel representation

The most straightforward approach to image classification is the direct use of the image pixel values as features. The images are scaled to a common size and represented by a feature vector that contains image pixel values. It has been shown that for classification and retrieval of medical radiographs, this method serves as a reasonable baseline [45]. We used a 32x32 down-sampled representation of the images as recommended by Tommasi et al. [7]. The obtained 1024 pixel values were then used as input features. Fig. 3 shows how we built the raw pixel representation for each image.

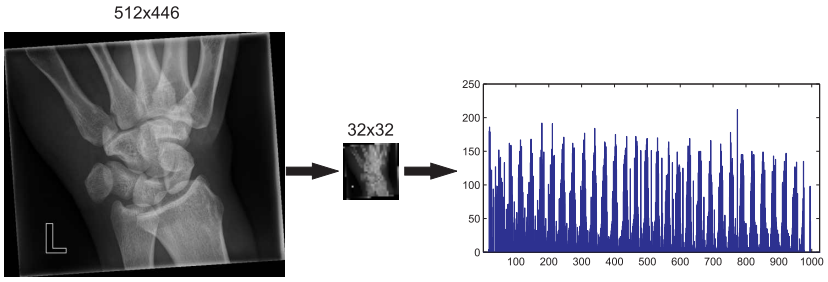


Figure 3: Down-sampling for raw pixel representation

4.2. Local binary patterns

Local binary patterns (LBP) are one of the best representations of texture content in images [8]. They are invariant to monotonic changes in gray-scale images and fast to compute. Furthermore, they are able to detect different micro patterns, such as edges, points and constant areas.

The basic idea behind the LBP approach is to use the information about the texture from a local neighborhood. First, we define the radius R of the local neighborhood under consideration. The LBP operator then builds a binary code that describes the local texture pattern in the neighborhood set of P pixels. The binary code is obtained by applying the gray value of the neighborhood center as a threshold. The binary code is then converted to a decimal number which represents the LBP code. Formally, given a pixel at position (x_c, y_c) the resulting LBP code can be expressed as follows:

$$LPB_{(P,R)}(x_c, y_c) = \sum_{n=0}^{P-1} S(i_n - i_c)2^n \tag{3}$$

where n ranges over the P neighbors of the central pixel (x_c, y_c), i_c and i_n are the gray-level values of the central pixel and the neighbor pixel, and $S(x)$ is defined as:

$$S(x) = \begin{cases} 1, & \text{if } x \geq 0 \quad (4a) \\ 0, & \text{otherwise} \quad (4b) \end{cases}$$

The image is traversed with the LBP operator pixel by pixel and the outputs are accumulated into a discrete histogram. However, not all LBP codes are informative. Certain LBP codes capture fundamental properties of the texture and are called uniform patterns because they constitute the vast majority, sometimes over 90 percent, of all patterns present in the observed textures [8]. These patterns have one thing in common, namely, a uniform circular structure that contains very few spatial transitions. They function as templates for micro-structures such as bright spot, flat area or dark spot.

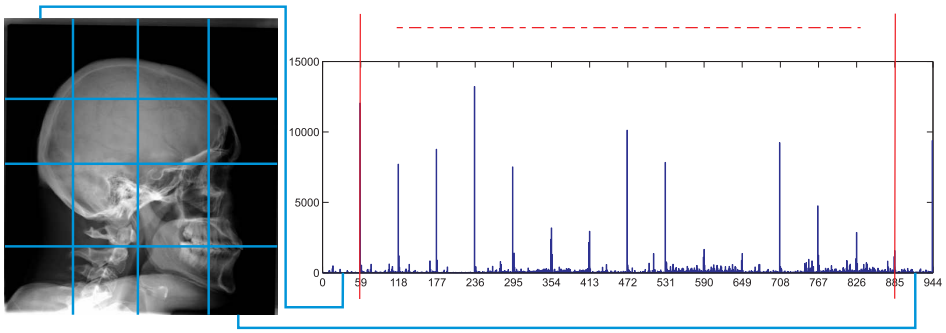


Figure 4: The image is divided into 4x4 non-overlapping sub-images from which LBP histograms are extracted and concatenated into a single, spatially enhanced histogram

In our experiments, we used the patterns $LBP^{u2}_{8,1}$, where the superscript $u2$ reflects the use of uniform patterns that have a U value of at most 2 on a neighborhood of size 8 and radius 1. The U value is the number of spatial transitions (bitwise 0/1 changes) in the pattern. The non-uniform patterns (patterns that have U value larger than 2) are grouped under one bin in the resulting histogram. With the $LBP^{u2}_{8,1}$ operator, the number of bins in the histogram is reduced from 256 to 59 (58 bins for uniform patterns and one bin for non-uniform/noisy patterns).

To spatially enhance the descriptors and improve the performance, it has been suggested to repeatedly sample predefined sub-regions of an image (e.g., 1x1, 2x2, 4x4 or 1x3) [46]. The different resolutions are then aggregated into a spatial pyramid which allows for region-specific weighting. Following these approaches, we divide the images into 4x4 non-overlapping sub-images (blocks) and concatenate the LBP histograms extracted for each sub-image into a single, spatially enhanced feature histogram. This approach aims at obtaining a more local description of the images. Fig. 4 shows how we build the LBP histogram with 944 bins in total for each image (16 blocks with 59 bins each).

4.3. Edge histogram descriptors

Edge detection is a fundamental problem of computer vision and has been widely investigated [47]. The goal of edge detection is to mark the points in a digital image at which the luminous intensity changes sharply. An edge representation of an image drastically reduces the amount of

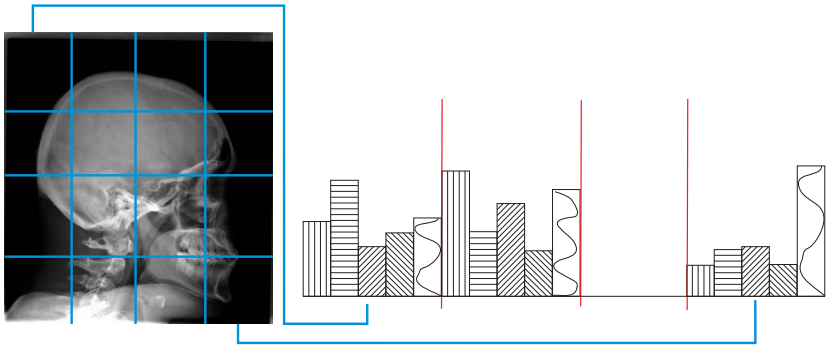


Figure 5: The image is divided into 4x4 non-overlapping sub-images. For each sub-image, five types of edge bins are calculated and concatenated into a single, spatially enhanced histogram

data to be processed, yet it retains important information about the shapes of objects in the scene. Edges in images constitute important features to represent their content.

The edge histogram in the image space represents the frequency and the directionality of the brightness changes in the image. To represent it, the MPEG-7 standard defines the edge histogram descriptor (EHD) [9]. The edge histogram descriptor basically represents the distribution of five types of edges (vertical, horizontal, two types of diagonal and non-directional edges; see Fig. 2). We divide the image space into 4x4 non-overlapping blocks, yielding 16 equal-sized sub-images and count the edges on each one of them (as shown in Fig. 5).

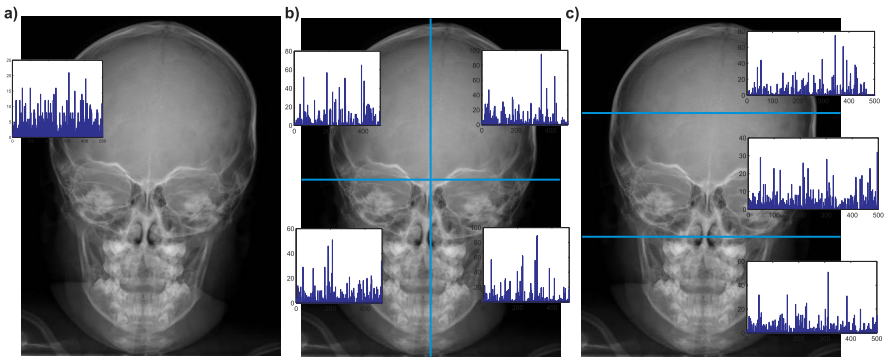


Figure 6: Three different spatial pyramids used in our experiments, a) 1x1, b) 2x2 and c) 1x3. The spatial pyramid constructs feature vectors for each of the specific part of the image.

To characterize the sub-images, a histogram of edge distribution for each sub-image is generated. Edges in the sub-images are categorized into five types: vertical, horizontal, 45-degree diagonal, 135-degree diagonal and non-directional edges, as presented in Fig. 5. The histogram for each sub-image represents the relative frequency of occurrence of the five types of edges in the corresponding sub-image and thus contains five bins.

Since there are 16 sub-images in the image and 5 types of edges, a total of 80 histogram bins are required. Note that each of the 80-histogram bins has its own semantics in terms of location and edge type. In our experiments, the edge detection is performed using the Canny edge detection algorithm [48].

4.4. SIFT descriptors

We employ the bag of features approach commonly used in many state of the art approaches in image classification [49]. The basic idea of this approach is to sample a set of local image patches using some method (densely, randomly or using a key-point detector) and calculate a visual descriptor on each patch (SIFT descriptor, normalized pixel values). The resulting distribution of descriptors is then quantified against a pre-specified visual codebook which converts it to a histogram. The main issues that need to be considered when applying this approach are: sampling of the patches, selection of the visual patch descriptor and building the visual codebook.

We use dense sampling of the patches, which samples an image grid in a uniform fashion using a fixed pixel interval between patches. We use an interval distance of 6 pixels and sample at multiple scales ($\sigma = 1.2$ and $\sigma = 2.0$). Due to the low contrast of the radiographs, it would be difficult to use any detector for points of interest. Also, it has been pointed by Zhang et al. [49], that a dense sampling is always superior to any strategy based on detectors for points of interest. We calculate a SIFT descriptor [10] for each image patch.

The crucial aspects of a codebook representation are the codebook construction and assignment. An extensive comparison of codebook representation variables is given by van Gemert et al. [50]. We employ k-means clustering (as implemented in the *R* environment) [51] on 400000 randomly chosen descriptors from the set of images available for training. k-means partitions the visual feature space by minimizing the variance between a predefined number of k clusters. Here, we set k to 500 and thus define a codebook with 500 codewords [7].

Dense sampling gives an equal weight to all key-points, irrespective of their spatial location in the image. To overcome this limitation, we follow the spatial pyramid approach which we applied for the LBP descriptor. For this descriptor, we used a spatial pyramid of 1x1, 2x2, and 1x3 regions. Since every region is an image in itself, the spatial pyramid can easily be used in combination with dense sampling. The resulting vector with 4000 bins ((1x1 + 2x2 + 1x3)x500) was obtained by concatenation of the eight histograms. Fig. 6 shows an example of the histograms extracted from an image for the spatial pyramids of 1x1, 2x2 and 1x3.

4.5. Feature fusion schemes

Different visual features bringing different information about the visual content of the images clearly outperform single feature approaches [5], [7]. Following these findings, we combine the different visual features described above. We investigate two different feature fusion schemes: low level (LL) and high level (HL). These fusion schemes are depicted in Fig. 7.

For the low level feature fusion scheme, the descriptors are concatenated in a single feature vector and a classifier is trained on the joint feature vector. The high level fusion scheme averages the predictions from the individual classifiers trained on the separate descriptors.

5. Experimental setup

In this section, we present the experimental setup we used to evaluate the proposed system and compare it to other approaches. First, we present the databases of images that we use. Next, we describe the evaluation metrics we use to assess the predictive performance of the classifiers. We then state the experimental questions that we investigate in this study. We specify the parameter instantiations for the algorithms and the design of the experiments.

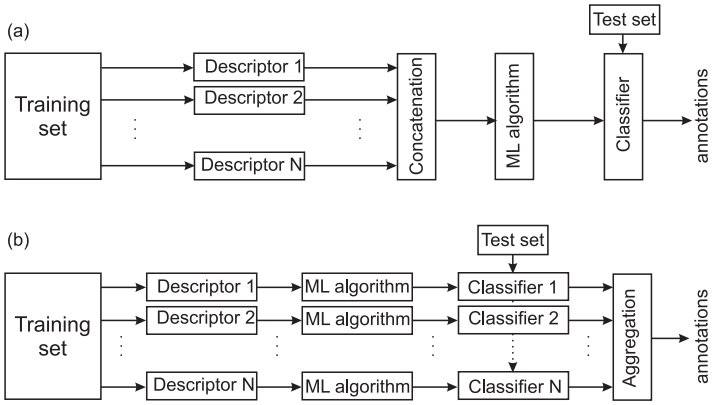
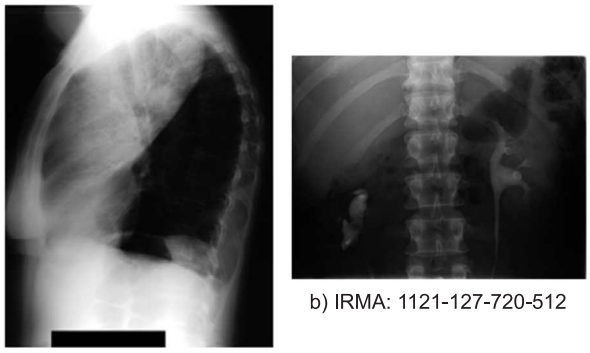


Figure 7: Fusion schemes for the different descriptors. a) Low level fusion, b) High level fusion.

5.1. IRMA database

We evaluated our system by applying it to the database for the ImageCLEF2009 medical image annotations task [5]. This database is provided by the IRMA group from the University Hospital of Aachen, Germany [35]. The database contains 12677 fully annotated radiographs, taken randomly from medical routine, which should be used to train a classifier. The dataset contains two parts: ImageCLEF2007 (12339 training and 1353 testing images) and ImageCLEF2008 (12667 training and 1733 testing images). These datasets present a difficult classification problem. First, the classes in the training set are extremely imbalanced (e.g. there are classes with less than 10 images and classes with more than 2000 images). Second, the distribution of the classes in the training set is different from the one on the testing set.



a) IRMA: 1123-211-520-3a0

b) IRMA: 1121-127-720-512

Figure 8: IRMA-coded chest and abdomen radiograph. For instance, the code for the biological axis (512) on the subfigure b) is translated as follows: 5 is for uropoietic system, 51 is for uropoietic system, kidney and 512 is uropoietic system, kidney, renal pelvis. The renal pelvis is an element of the kidney, which in turn is an element of the uropoietic system

The images are labeled according to the four annotation label sets [5]. We used the ImageCLEF2007 label set with 116 IRMA codes and the ImageCLEF2008 label set with 193 IRMA codes, both with a hierarchical nature of the coding scheme [35]. The goal is to correctly annotate 1353 (for 2007) and 1733 (for 2008) images that are provided without labels, using the different respective annotation label sets in turn.

The IRMA coding scheme consists of four axes with three to four positions, each position

taking a value from the set 0,..., 9, a,..., z, where '0' denotes 'unspecified' and determines the end of a path along an axis. The four axes are: technical axis (T, image modality), directional axis (D, body orientation), anatomical axis (A, body region examined) and biological axis (B, biological system examined). This allows a short and unambiguous notation (IRMA: TTTT-DDD-AAA-BBB), where T, D, A, and B denotes a coding or sub-coding digit of the respective axis. A small part of the IRMA coding hierarchy is presented in Fig. 1. Fig. 8 gives two examples of unambiguous image classification using the IRMA code.

The IRMA code is hierarchical in its nature and it allows us to exploit the hierarchy of the code. This means that we can construct an automatic image annotation system based on predictive clustering trees for HMC.

5.2. Evaluation metrics

In this study, we use two evaluation metrics: the ImageCLEF hierarchical evaluation measure [5] and overall recognition rate. The ImageCLEF hierarchical evaluation measure takes into account the depth and the difficulty of the predictive problem ('branching factor') at which an error has occurred (Equation 5). It can be calculated using the following formula:

$$\sum_{i=1}^{I} \frac{1}{b_i} \frac{1}{i} \delta(v_i, \bar{v}_i), \tag{5}$$

$$\delta(v_i, \bar{v}_i) = \begin{cases} 0, & \text{if } v_j = \bar{v}_j \forall j \leq i & \text{(6a)} \\ 0.5, & \text{if } v_j = * \exists j \leq i & \text{(6b)} \\ 1, & \text{if } v_j \neq \bar{v}_j \exists j \leq i & \text{(6c)} \end{cases}$$

where I is the depth of the hierarchy, b_i is the number of possible labels at the error ('branching factor') and i is the depth at which the error occurred. This measure allows the classifier not to predict the complete code/annotation, that is, the classifier can predict the first 2 nodes of the code (level of the hierarchy) and then say 'don't know' (encoded by *) for the next node/level. The ImageCLEF evaluation measure can range from 0 to the number of testing images. If this measure is closer to 0, then the classifier is more accurate.

The overall recognition rate is a very common and widely used evaluation measure. It is the fraction of the test images whose complete IRMA code was predicted correctly.

5.3. Experimental questions

The goal of this study is to answer the following questions:

1. Does the use of the hierarchy (in ensembles of PCTs) improve the predictive performance over flat classification (SVMs)?
2. How is the relative performance of the two techniques affected by the:
   (a) Use of PCT ensembles versus single PCTs in the domain of image annotation?
   (b) Different ensemble methods: bagging or random forests?
   (c) Different feature extraction techniques for medical X-Ray images?
   (d) Schemes for fusion of the descriptors from the feature extraction techniques?
3. Is the proposed system with ensembles of PCTs for HMC scalable and efficient?

For the first three questions (1, 2a and 2b), we evaluate the performance of PCTs for HMC and ensembles (bagging and random forest) of PCTs. After that, we compare the best method for HMC with SVMs. It has been shown [25] that exploiting the structure of the hierarchy in tree classifiers yields better predictive performance in the domain of functional genomics. Here, we compare the performance of the ensemble classifiers with SVMs for flat classification - the most widely used classifiers for medical image annotation [7].

To check which feature extraction technique is most suitable for medical X-Ray images (question 2c), we compare the performance of the classifiers on each type of visual descriptors. For this purpose, we discuss only the results from the separate runs of the descriptors (first four rows from Table 1 and Table 2).

The various feature extraction techniques capture different aspects of an image. We also investigate whether the combination of feature extraction techniques can increase the predictive performance (question 2d). The results from the fusion schemes are presented in the last 10 rows in Table 1 and Table 2.

We compare the execution times of the different classifiers to assess the efficiency and scalability of the system (question 3). We measure the time needed to train the classifiers; for SVMs this includes also the time needed to optimize the parameters.

5.4. Experimental design

In this section, we describe the experimental setup that we used. First, we describe an adaptation of the hierarchy of the IRMA code and then the parameter instantiations of the learning algorithms. Note that we stated the parameters for the feature extraction techniques while explaining them (see Section 4).

The IRMA coding scheme was proposed by Lehmann et al. in [35]: It consists of four axes which are strictly hierarchical (tree-shaped hierarchies). The literature [5],[35] suggests that these four axes are independent. We conducted a series of experiments predicting the four axes simultaneously (combined in a single hierarchy) and separately. The predictive performance when using all four axes simultaneously was higher as compared to using each axis separately. This leads us to believe that these axes are not-independent. In a separate study, Tommasi et al. [7] come to a similar conclusion. To address this issue, we adapted the IRMA coding hierarchy as follows.

We take the code of the first position for the biological axis and add it in front of the codes for the anatomical and directional axes. The inclusion of the biological code in the first level in the hierarchy helps us to initially filter the images resulting in large visual differences in the first level of the hierarchy. In the context of the axis A, the first level of axis B is necessary because the examined body region insufficiently describes the content and structure of the images. For example, fluoroscopy of the abdominal region may access the vascular or the gastrointestinal system depending on the way the contrast agent is administered, which results in different image textures. For the directional axis, this is even more obvious. For instance, an image of a chest and an image of a hand can have the same directional code, but are visually very different.

The hierarchy of the IRMA code was adapted in order to increase the inter-class variability and decrease the intra-class variability of the images. Fig. 9 shows the adapted hierarchy of classes that we use in the experiments. Note that this hierarchy was only used to train the classifier. The evaluation was performed by using the original IRMA hierarchy.

In the following, we state the parameter instantiations that we used to train the classifiers: PCTs, ensembles and SVMs. The algorithm for learning PCTs requires as input the weight of

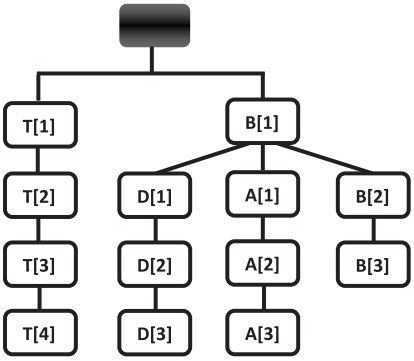


Figure 9: The adapted hierarchy of the classes in the IRMA code

the depth in the hierarchy. We set w_0 to 0.75 to force the algorithm to make better predictions on the upper levels of the hierarchy. Also, we performed F-test pruning to prevent over-fitting of the trees [25].

We trained ensembles of 100 un-pruned trees (PCTs). For the base PCTs, we used the same weight (0.75) used to train the single PCTs. The size of the feature subset that is retained at each node, when training a random forest, was set to 10% of the number of descriptive attributes. Remember that the output of the classifier is a probability that a given example is annotated with a given label. If the probability is higher than a given threshold (obtained during the training of the classifier), then the example is annotated with the given label. Since the hierarchical evaluation measure allows the classifier to predict a portion of the code, different thresholds for the different levels of the hierarchy were selected. If a probability for a given code was lower than the threshold, then for this code and its sub-codes the classifier predicts 'don't know'.

For training the SVMs, we used a custom developed application . This application uses the LibSVM library [52]. We apply the *One-against-All* (OvA) approach to solve the partial binary classification problems. Each of the SVMs was trained with a χ^2 kernel. We optimize the cost parameter C of the SVMs using an automated parameter search procedure. For the parameter optimization, we separate 20% of the training set and use it as validation set. After finding the optimal C value, the SVM was finally trained on the whole set of training images.

For the evaluation of the SVMs using the hierarchical error measure, we applied confidence based opinion fusion [7]. Let us assume that there are N classes. Then, using the OvA approach, N SVMs are trained – each separating a single class from all remaining ones. The decision is based on the distances of the test sample to the N hyperplanes. The prediction then corresponds to the hyperplane for which the distance is largest. The confidence based opinion fusion, however, takes into account the difference of the predictions with the two largest distances reported from the SVMs classifiers. This difference is computed only if their distances differ less than a threshold value (obtained during training using the validation data set). In that case, the final prediction will contain 'don't know' starting from the position where the two underlying predictions begin to differ. For example, if the two predictions for the anatomical axis are 411 and 421 then the final prediction will be 4**. This approach improves the hierarchical error measure for the SVMs classifier by 10 to 20 points depending on the used descriptors.

6. Results and discussion

Table 1 and Table 2 present the results obtained using the experimental setup described in Section 5 in terms of the hierarchical evaluation measure (HEM) and overall recognition rate (RR) respectively. In the discussion of the results, we first compare the performance of single PCTs and ensembles of PCTs. We then compare the performance of the best ensemble method (random forests) and SVMs. We focus on the first evaluation measure HEM (Table 1), since the two show similar behavior; the conclusions for HEM are also valid for RR.

Table 1: Predictive performance of the models learned from descriptors produced by different feature extraction algorithms and their combinations. The best results are shown in boldface. Performance is given in terms of the ImageCLEF hierarchical evaluation measure HEM, where smaller values mean better performance. The low-level fusion results are in rows that end with 'LL' and high-level fusion results are in rows that end with 'HH'.

	Hierarchical Error Measure							
	ImageCLEF2007				**ImageCLEF2008**			
	SVM	RF	Bag	PCTs	SVM	RF	Bag	PCTs
SIFT	75.00	**58.90**	59.78	180.00	179.88	**161.67**	161.47	320.90
LBP	124.44	95.71	95.71	210.40	257.92	209.47	208.97	360.00
EHD	127.41	105.12	105.12	222.39	265.95	249.44	249.74	380.12
32x32	202.94	195.78	200.12	310.90	376.93	361.21	361.31	530.11
LBP+EHD_LL	99.48	85.56	86.80	200.12	221.96	190.12	190.22	347.89
LBP+SIFT_LL	72.71	52.89	53.22	178.29	175.65	157.38	157.48	317.12
EHD+SIFT_LL	72.37	56.11	57.11	179.12	170.97	159.30	159.33	318.87
LBP+EHD+SIFT_LL	70.45	**51.90**	52.33	177.23	170.87	**153.21**	153.41	317.00
LBP+EHD+SIFT+32x32_LL	69.46	52.23	53.00	178.12	169.11	154.23	154.63	318.50
LBP+EHD_HL	100.37	87.90	89.21	201.30	223.73	195.96	196.06	347.90
LBP+SIFT_HL	73.72	54.21	54.56	178.90	177.12	159.73	160.03	318.00
EHD+SIFT_HL	72.70	59.12	61.71	179.50	174.44	161.85	162.05	318.80
LBP+EHD+SIFT_HL	71.58	52.54	53.00	177.90	174.18	156.21	156.31	317.90
LBP+EHD+SIFT+32x32_HL	70.46	53.90	54.50	178.58	173.28	156.50	156.70	318.30

The results clearly show that ensemble methods outperform single PCTs on all datasets: random forests are significantly better (according to the non-parametric Wilcoxon test for statistical significance) than single PCTs ($p < 4 \cdot 10^{-6}$) and bagging is better than single PCTs ($p < 4 \cdot 10^{-6}$). A comparison between the two ensemble methods shows that random forests outperforms bagging and that the difference is statistically significant ($p < 1 \cdot 10^{-4}$).

While extremely efficient, individual PCTs have the drawback of only using a small number of the available features, which results in low predictive performance. The PCTs trade predictive performance for interpretability. However, in the domains where interpretability of the model is a necessity, PCTs are the models that should be considered.

We next compare the performance of random forests to the performance of SVMs. On all datasets, random forests perform better than SVMs; the difference on average is ~ 17 points for the ImageCLEF2007 and ~ 20 points for ImageCLEF2008 datasets (note that a point in the HEM roughly corresponds to one completely misclassified image). The difference in performance is statistically significant (with $p < 4 \cdot 10^{-6}$). This shows that exploiting the structure of the hierarchy does help in improving the predictive performance.

We then analyze the results for the individual feature extraction algorithms (top 4 rows from Table 1 and Table 2). We can note the high predictive performance of the SIFT histogram: it is

Table 2: Predictive performance of the models learned from descriptors produced by different feature extraction algorithms and their combinations. The best results are shown in boldface. Performance is given in terms of the overall recognition rate evaluation measure, where larger values mean better performance. The low-level fusion results are in rows that end with 'LL' and high-level fusion results are in rows that end with 'HH'.

	Overall Recognition Rate							
	ImageCLEF2007				**ImageCLEF2008**			
	SVM	RF	Bag	PCTs	SVM	RF	Bag	PCTs
SIFT	77.31	**79.37**	79.08	63.04	62.44	**64.91**	64.80	52.04
LBP	70.36	75.24	75.24	56.02	56.26	60.99	60.70	47.02
EHD	68.37	72.28	72.21	55.06	54.53	54.99	54.81	45.00
32x32	57.35	58.01	57.64	45.97	45.47	45.52	45.47	36.98
LBP+EHD_LL	75.09	76.97	75.75	58.98	60.53	61.51	61.39	48.99
LBP+SIFT_LL	77.90	81.00	80.93	64.52	62.26	65.49	65.43	53.49
EHD+SIFT_LL	78.20	79.97	79.82	64.00	63.19	64.97	64.80	52.97
LBP+EHD+SIFT_LL	78.42	**81.96**	81.67	64.89	63.30	**65.95**	65.83	53.72
LBP+EHD+SIFT+32x32_LL	78.49	81.22	81.00	64.30	63.53	65.78	65.55	52.97
LBP+EHD_HL	74.87	76.01	76.64	58.38	60.13	61.45	61.39	48.87
LBP+SIFT_HL	77.46	79.97	79.97	64.22	62.26	65.32	65.14	53.49
EHD+SIFT_HL	77.90	79.00	78.86	63.93	62.44	64.80	64.62	52.79
LBP+EHD+SIFT_HL	78.05	81.00	80.93	64.59	62.78	65.78	65.72	53.66
LBP+EHD+SIFT+32x32_HL	78.42	80.70	80.56	64.37	63.13	65.60	65.49	52.97

most capable of capturing the hierarchical structure of the X-ray images. The other feature extraction algorithms follow after and are ordered by performance as follows: LBP, then EHD and the simplest descriptor RPR, which has the worst performance. The difference of performance to the LBP operator is very noticeable and larger for SVMs than for random forests: on the ImageCLEF2007 dataset, random forests are better by $\sim$ 30 points and on ImageCLEF2008 by $\sim$ 50 points and on the ImageCLEF2007 dataset, SVMs are better by $\sim$ 50 and on ImageCLEF2008 by $\sim$ 80 points. The LBP descriptors capture information that is more easily utilized by the random forests than by the SVMs.

The experimental results show that the features that describe the image content in a local manner (i.e., SIFT descriptors) outperform the ones that provide global descriptions. The local features capture the details in an image, while the global features are able to retain information on the whole image as a source of context. Furthermore, the SIFT descriptor is robust to noise, illumination, scale, translation and rotation changes. Hence, it can better resolve the inter and intra-class variability, thus it can offer better information to the classifier. We can conclude that the local features are generally more informative than global features for the medical image annotation task at hand.

We also compare the results of the experiments conducted with different feature fusion schemes. Inclusion of more than one type of features in the classification process contributes to better representation of the hierarchical nature of the images and helps to further improve the predictive performance. Low level fusion (concatenation) yields slightly better predictive performance than high level fusion. This is valid for all algorithms used in this study.

The classifiers on the fused feature sets use more information about the different aspects of an image that are captured by the different descriptors. Namely, they can consider combinations of features from different descriptors. This additional information is orthogonal and helps the classifiers to produce better annotations. Moreover, the ensembles of trees, such as random forests, can effectively exploit the information provided by the large number of features. Thus,

low-level fusion yields better performance than high-level fusion.

The best results are achieved by using random forests on the concatenated SIFT, LBP and EHD descriptors (boldface in Table 1 and Table 2). This holds for both datasets, ImageCLEF2007 and ImageCLEF2008. The best results for overall recognition rate (81.96) are close to the error rate for the DICOM header. Namely, Guld et al [4] reported 15.5% disagreement between the data for the DICOM header and the radiologists' reference categorization. Moreover, our best results are better than the best results reported so far on this database [5]. Our score of 153.2 for ImageCLEF2008 is by 16.3 points better than the best result, and the score of 51.9 for ImageCLEF2007 is by 12.4 points better than the best result.

From the results, we can also notice the worse performance of all algorithms on the ImageCLEF2008 dataset, as compared to the ImageCLEF2007 dataset. This is mainly due to the larger hierarchy of the ImageCLEF2008 dataset (195 nodes as compared to 140 nodes for the ImageCLEF2007 dataset). In addition, the difference of the distribution of images in the training and the testing set is bigger for ImageCLEF2008 than for ImageCLEF2007.

Table 3: Running times of the algorithms: time needed to construct the classifier and time needed to produce an annotation for an unseen image. Note that this table only lists the results for the low-level fusion scheme (the results that end with 'LL'). The running times for the high-level fusion are the sum of running times for its constitutive runs. The experiments were executed on a Linux server with two Intel Quad-Core Processors@2.5GHz and 64GB of RAM.

		ImageCLEF 2007				**ImageCLEF 2008**			
		SVM	RF	Bag	PCTs	SVM	RF	Bag	PCTs
Training time [sec]	**EHD**	2820.873	92.668	566.880	4.667	3113.320	115.129	716.606	5.446
	LBP	4323.681	1909.510	21684.124	127.889	4406.340	2631.485	28612.105	158.955
	32x32	4745.630	1909.427	21458.823	110.436	5467.686	2614.089	28410.495	151.317
	SIFT	12451.760	2886.417	31611.480	227.709	13219.039	3717.713	40567.323	248.920
	LBP+EHD_LL	4824.592	2315.010	21629.071	231.516	4480.761	3012.840	28106.304	254.442
	LBP+SIFT_LL	14871.131	5095.170	55476.671	502.794	15788.345	6508.022	70057.262	487.347
	EHD+SIFT_LL	12656.792	3299.330	36001.937	337.784	13430.779	4165.986	45921.571	393.629
	LBP+EHD+SIFT_LL	15076.162	5094.305	55724.765	504.575	16006.638	6460.307	70462.933	500.873
	LBP+EHD+SIFT+32x32_LL	17700.564	6936.030	73786.231	591.772	18800.790	9128.094	95792.121	679.572
Testing time per image [sec]	**EHD**	0.016	0.002	0.003	0.001	0.019	0.004	0.003	0.001
	LBP	0.172	0.002	0.003	0.001	0.179	0.003	0.003	0.001
	32x32	0.189	0.002	0.003	0.001	0.192	0.002	0.002	0.001
	SIFT	0.551	0.002	0.003	0.001	0.591	0.003	0.004	0.001
	LBP+EHD_LL	0.175	0.003	0.002	0.001	0.176	0.002	0.003	0.001
	LBP+SIFT_LL	0.569	0.002	0.002	0.001	0.565	0.003	0.003	0.001
	EHD+SIFT_LL	0.552	0.002	0.003	0.001	0.552	0.003	0.003	0.001
	LBP+EHD+SIFT_LL	0.570	0.002	0.002	0.001	0.569	0.002	0.002	0.001
	LBP+EHD+SIFT+32x32_LL	0.600	0.002	0.002	0.002	0.590	0.003	0.003	0.002

Additionally, we assess the efficiency of the algorithms by measuring the time needed to learn the classifier and time needed to produce an annotation for an unseen image. The running times for the algorithms are presented in Table 3. The random forests are the fastest method; they are $\sim$ 10 times faster than bagging and $\sim$ 5.5 times than the SVMs (including the optimization of the SVM parameters). Recall that the random forests are ensembles of PCTs that predict the complete hierarchy (a single model), while the SVMs construct a classifier for each node of the hierarchy separately. Hence, the increase of the hierarchy will significantly increase the training time of SVMs (additional classifiers should be trained), while the training time for random forests will increase only slightly. The efficiency of the random forests of PCTs is even more prominent when producing annotations for unseen images. The random forests in this case are $\sim$ 165 times faster than the SVMs. In this respect, bagging performs comparably to random forests. This is due to the fact that passing through the tree has logarithmic complexity with respect to the

number of leafs in the tree. Since random forests and bagging produce trees with similar sizes, these times will be similar. All in all, random forests of PCTs significantly outperform SVMs as compared by their training and testing times.

7. Experiments on photo annotation

To show the generality of the proposed system, we perform experiments on annotation of general images. In this section, we first present the experimental setup that we used (the data, evaluation metrics and the experimental design). We then present the results and compare them to those of state-of-the-art approaches used in image annotation.

7.1. Experimental setup

This set of experiments was performed using the database from the ImageCLEF@ICPR photo annotation task [53]. The database consists of 5000 train, 3000 validation, and 10000 test images annotated with 53 visual concepts organized in a small hierarchy with tree structure (see Fig. 10 for an example). The average number of annotations per image is 8.68 (including both leaf and internal nodes from the hierarchy). The visual concepts also contain abstract categories like Family/Friends, Partylife, Quality (blurred, underexposed, ...) and etc., thus making the annotation/classification task very challenging.

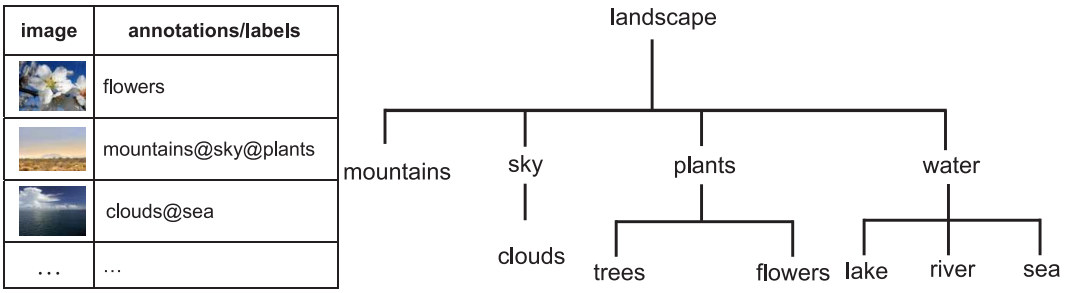


Figure 10: A fragment of the hierarchy for image annotation. The annotations are part of the hierarchical classification scheme for the ICPR 2010 photo annotation task (right). The table contains a set of images with their annotations (left).

The measures that we used to evaluate the performance of the algorithms on the medical X-ray images are specific for the problem of annotation of medical images using the IRMA coding scheme [2]. Here, we use the most widely used evaluation measure in the area of 'general photo annotation'/'visual concept detection': mean average precision (MAP) [12]. For a given target visual concept, the average precision can be calculated as the area under the precision-recall curve for that target. Hence, it combines both precision and recall into a single performance value. The average precision is calculated for each visual concept separately and the obtained values are then averaged to obtain the mean average precision. Because the true labels of the

[2]Note that the hierarchical error measure allows the algorithm to say 'don't know' for some classes, since the maximum number of labels per image with the IRMA coding scheme is known. In the case of general images, an image can be annotated with zero or $|C|$ classes. Also, for the Overall recognition rate, for the case of IRMA coding scheme, the number of possible combinations of labels is limited, while in the case of general images, this number is $2^{|C|}$. This makes overall recognition rate not suitable for measuring the predictive performance of algorithms in annotating general images.

test images from the ImageCLEF@ICPR 2010 database are not publicly available, we report the MAP value obtained on the validation dataset.

For the images from this database, we use SIFT features, which were the best performing features in previous experiments (also SIFT features are typically used in this type of problem [14]). The SIFT features for this set of experiments were constructed using a visual codebook with 4000 instead of 500 words (see Section 4.4). This modification was made because most of the state-of-the-art approaches for image classification of general photos use a visual codebook with 4000 words [14], [12]. In the previous experiments, random forests were the best performing method, so again we train random forests with 100 un-pruned PCTs for HMC. For the base PCTs, we used the same weight (0.75) and the size of the feature subset that is retained at each node was set to 10% of the number of descriptive attributes (same as in the experiments from the Section 5).

To train the SVMs, we use the LibSVM implementation with probabilistic outputs [54]. To solve the multiple classification problems, we employ again the *One-against-All* approach. For each visual concept, we build a binary classifier where instances associated with that visual concept are in one class (positive) and the rest are in another class (negative). To handle the imbalance in the number of positive versus negative training examples, we fix the weights of the positive and negative class. The weight of the positive class is set to $\frac{\#pos+\#neg}{\#pos}$ and the weight of the negative class is set to $\frac{\#pos+\#neg}{\#neg}$, with $\#pos$ the number of positive instances in the train set and $\#neg$ the number of negative instances [15]. As in the previous experiments, we optimize the value of the cost parameter C of the SVMs.

7.2. Results and discussion

The results from the photo annotation experiments are shown in Table 4. The table also contains the total training time and testing time per image for both SVMs and random forests of PCTs for HMC. From the presented results we can note that the random forests of PCTs for HMC outperform the SVMs both in terms of predictive performance and efficiency. The latter holds especially for the time needed to produce an annotation for a given test image: our approach is more than 500 times faster than the SVMs.

Table 4: Results of the photo annotation experiments evaluated using Mean Average Precision (larger values of MAP mean better performance).

	MAP	**Train time**	**Test time per image**
RF	0.450	9113.516	0.002
SVM	0.428	11821.227	1.078

Following the results from the study performed by Mensink et al. [22], this means that our system also outperforms the TagProp [21] approach for image annotation. The results show that our system offers better predictive performance and efficiency than systems that are most widely used for annotation of images. All in all, the proposed system has high predictive performance and efficiency, is general and is easily applicable to other domains.

8. Conclusions

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 this paper, we propose to exploit the annotation hierarchy in image annotation by using ensembles of trees for HMC. Our approach to HMC exploits the annotation hierarchy by building a single classifier that simultaneously predicts all labels in the hierarchy. A substantial performance improvement is achieved by building ensembles of HMC trees, such as random forests.

We apply our approach to two benchmark tasks of hierarchical annotation of medical (X-ray) images and an additional task of photo annotation (i.e., visual concept detection). We compare it to a collection of SVMs (trained with a χ^2 kernel), each predicting one label at the lowest level of the hierarchy, 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. 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 trees 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 trees 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 representations (we consider raw pixel representation, local binary patterns, edge histogram descriptors and SIFT histograms), as well as combinations thereof: The SIFT histograms are the best individual descriptors. Moreover, combinations of different descriptors yield better predictive performance than the individual descriptors. 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 trees for HMC are a superior alternative to using collections of SVMs, which are most-commonly applied in this context.

We expect it is possible to further improve the predictive performance of our system. We could try to adapt our tree-learning approach to tackle the shift in distribution of images between the training and the testing set. Better performance may also be obtained by including high level feature extraction algorithms able to give more understandable and compact representation of the visual content of the images (segmented objects with relations among them).

Let us conclude by emphasizing 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.

Acknowledgment

The authors would like to thank Donato Malerba, Department of Computer Science, University of Bari for his valuable comments and suggestions.

References

[1] R. Choplin, J. Boehme, C. Maynard, Picture archiving and communication systems: an overview, Radiographics 12 (1) (1992) 127–129.

[2] S. Becker, R. Arenson, Costs and benefits of picture archiving and communication systems, Journal of the American Medical Informatics Association 1 (5) (1994) 361–371.

[3] N. E. M. Association, Digital imaging and communications in medicine - DICOM, http://dicom.nema.org/ (2009). URL http://dicom.nema.org/

[4] M. O. Guld, M. Kohnen, D. Keysers, H. Schubert, B. B. Wein, J. Bredno, T. M. Lehmann, Quality of DICOM header information for image categorization, in: SPIE vol. 4685 - Medical Imaging 2002: PACS and Integrated Medical Information Systems: Design and Evaluation, 2002, pp. 280–287.

[5] T. Tommasi, B. Caputo, P. Welter, M. O. Guld, T. M. Deserno, Overview of the CLEF 2009 medical image annotation track, in: CLEF 2009 Workshop - LNCS 6242, 2010, pp. 85–93.

[6] C. Silla, A. Freitas, A survey of hierarchical classification across different application domains, Data Mining and Knowledge Discovery 22 (1-2) (2011) 31–72.

[7] T. Tommasi, F. Orabona, B. Caputo, Discriminative cue integration for medical image annotation, Pattern Recognition Letters 29 (15) (2008) 1996–2002.

[8] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (7) (2002) 971–987.

[9] D. K. Park, Y. S. Jeon, C. S. Won, Efficient use of local edge histogram descriptor, in: ACM workshops on Multimedia, 2000, pp. 51–54.

[10] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[11] S. Nowak, ImageCLEF@ICPR contest: Challenges, methodologies and results of the photo annotation task, in: International Conference on Pattern Recognition, 2010, pp. 489–492.

[12] M. Everingham, L. V. Gool, C. Williams, A. Zisserman, The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results, http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html (2009).

[13] T. Deselaers, H. Muller, P. Clough, H. Ney, T. M. Lehmann, The CLEF 2005 automatic medical image annotation task, International Journal of Computer Vision 74 (1) (2005) 51–58.

[14] S. Nowak, P. Dunker, Overview of the clef 2009 large-scale visual concept detection and annotation task, in: CLEF 2009 Workshop - LNCS 6242, 2010, pp. 94–109.

[15] K. van de Sande, T. Gevers, C. Snoek, Evaluating color descriptors for object and scene recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (9) (2010) 1582–1596.

[16] M. Marszałek, C. Schmid, Constructing category hierarchies for visual recognition, in: European Conference on Computer Vision - LNCS 5305, Springer, 2008, pp. 479–491.

[17] M. Marszałek, C. Schmid, Semantic hierarchies for visual object recognition, in: IEEE Conference on Computer Vision & Pattern Recognition, 2007, p. 7.

[18] S. Liu, H. Yi, L.-T. Chia, D. Rajan, Adaptive hierarchical multi-class SVM classifier for texture-based image classification, in: IEEE International Conference on Multimedia and Expo, 2005, p. 4.

[19] L. Zhigang, S. Wenzhong, Q. Qianqing, L. Xiaowen, X. Donghui, Hierarchical support vector machines, in: IEEE International Geoscience and Remote Sensing Symposium, 2005, pp. 186–189.

[20] X. Yuan, W. Lai, T. Mei, X.-S. Hua, X. qing Wu, S. Li, Automatic video genre categorization using hierarchical SVM, in: IEEE International Conference on Image Processing, 2006, pp. 2905–2908.

[21] M. Guillaumin, T. Mensink, J. Verbeek, C. Schmid, Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation, in: 12th International Conference on Computer Vision, 2009, pp. 309–316.

[22] T. Mensink, G. Csurka, F. Perronnin, J. Sanchez, J. J. Verbeek, LEAR and XRCE's participation to visual concept detection task - ImageCLEF 2010, in: CLEF 2010 LABs and Workshops, Notebook Papers, 2010, p. 12.

[23] T. Joachims, SVMStruct – support vector machine for complex outputs (2011). URL http://www.cs.cornell.edu/People/tj/svm_light/svm_struct.html

[24] T. Gärtner, S. Vembu, On structured output training: hard cases and an efficient alternative, Machine Learning 76 (2009) 227–242.

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

[26] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.

[27] S. Džeroski, P. Panov, B. Ženko, Ensemble methods in machine learning, in: Encyclopedia of complexity and systems science, Springer New York, 2009, pp. 5317–5325.

[28] G. Seni, J. F. Elder, Ensemble methods in data mining: Improving accuracy through combining predictions, Morgan & Claypool Publishers, 2010.

[29] T. Evgeniou, L. Perez-Breva, M. Pontil, T. Poggio, Bounds on the generalization performance of kernel machine ensembles, in: 17th International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., 2000, pp. 271–278.
[30] G. Valentini, T. Dietterich, Biasvariance analysis and ensembles of svm, in: Multiple Classifier Systems - LNCS 2364, Springer Berlin / Heidelberg, 2002, pp. 27–38.
[31] C. Wang, H. Yuan, J. Liu, T. Zhou, H. Lu, A novel support vector machine ensemble based on subtractive clustering analysis, in: Advances in Knowledge Discovery and Data Mining - LNCS 4426, Springer Berlin / Heidelberg, 2007, pp. 849–856.
[32] K. M. Ting, L. Zhu, Boosting support vector machines successfully, in: MCS '09: Proceedings of the 8th International Workshop on Multiple Classifier Systems - LNCS 5519, Springer-Verlag, 2009, pp. 509–518.
[33] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine Learning 36 (1) (1999) 105–139.
[34] H. Blockeel, L. Schietgat, J. Struyf, S. Džeroski, A. Clare, Decision trees for hierarchical multilabel classification: A case study in functional genomics, in: Knowledge Discovery in Databases: PKDD 2006 - LNCS 4213, Springer Berlin / Heidelberg, 2006, pp. 18–29.
[35] T. M. Lehmann, H. Schubert, D. Keysers, M. Kohnen, B. B. Wein, The IRMA code for unique classification of medical images, in: SPIE vol. 5033 - Medical Imaging 2003: PACS and Integrated Medical Information Systems: Design and Evaluation, 2003, pp. 440–451.
[36] H. Blockeel, L. D. Raedt, J. Ramong, Top-down induction of clustering trees, in: International Conference on Machine Learning, Morgan Kaufmann, 1998, pp. 55–63.
[37] L. Breiman, J. Friedman, R. Olshen, C. J. Stone, Classification and Regression Trees, Chapman & Hall/CRC, 1984.
[38] R. J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
[39] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Ensembles of multi-objective decision trees, in: European conference on Machine Learning - LNCS 4701, Springer, 2007, pp. 624–631.
[40] J. Struyf, S. Džeroski, Constraint based induction of multi-objective regression trees, in: 4th Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 3933, Springer, 2006, pp. 222–233.
[41] I. Slavkov, V. Gjorgjioski, J. Struyf, S. Džeroski, Finding explained groups of time-course gene expression profiles with predictive clustering trees, Molecular BioSystems 6 (4) (2010) 729–740.
[42] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[43] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[44] I. Dimitrovski, S. Loskovska, Content-based retrieval system for X-ray images, in: International Congress on Image and Signal Processing, 2009, pp. 2236–2240.
[45] D. Keysers, T. Deselaers, C. Gollan, H. Ney, Deformation models for image recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (8) (2007) 1422–1435.
[46] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: IEEE conference on Computer Vision and Pattern Recognition, 2006, pp. 2169–2178.
[47] D. Ziou, S. Tabbone, Edge detection techniques an overview, Pattern Recognition and Image Analysis 8 (24) (1998) 537–559.
[48] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 8 (6) (1986) 679–698.
[49] J. Zhang, M. Marszalek, S. Lazebnik, C. Schmid, Local features and kernels for classification of texture and object categories: A comprehensive study, International Journal of Computer Vision 73 (2) (2007) 213–238.
[50] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, J. M. Geusebroek, Visual word ambiguity, IEEE Transactions on Pattern Analysis and Machine Intelligence 99 (1).
[51] R. D. C. Team, R: A language and environment for statistical computing (2009).
URL `http://www.R-project.org`
[52] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm` (2001).
[53] S. Nowak, Imageclef@icpr2010 – photo annotation task, http://www.imageclef.org/2010/ICPR/ (2010).
URL `http://www.imageclef.org/2010/ICPR/`
[54] H.-T. Lin, C.-J. Lin, R. C. Weng, A note on Platt's probabilistic outputs for support vector machines, Machine Learning 68 (2007) 267–276.

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.

Schietgat et al. BMC Bioinformatics 2010, 11:2
http://www.biomedcentral.com/1471-2105/11/2

BMC Bioinformatics

RESEARCH ARTICLE Open Access

Predicting gene function using hierarchical multi-label decision tree ensembles

Leander Schietgat[1*], Celine Vens[1*], Jan Struyf[1], Hendrik Blockeel[1], Dragi Kocev[2], Sašo Džeroski[2]

Abstract

Background: *S. cerevisiae*, *A. thaliana* and *M. musculus* are well-studied organisms in biology and the sequencing of their genomes was completed many years ago. 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.

Results: We study the use of decision tree based models for predicting the multiple functions of ORFs. First, we describe an algorithm for learning hierarchical multi-label decision trees. These can simultaneously predict all the functions of an ORF, while respecting a given hierarchy of gene functions (such as FunCat or GO). We present new results obtained with this algorithm, showing that the trees found by it exhibit clearly better predictive performance than the trees found by previously described methods. Nevertheless, the predictive performance of individual trees is lower than that of some recently proposed statistical learning methods. We show that ensembles of such trees are more accurate than single trees and are competitive with state-of-the-art statistical learning and functional linkage methods. Moreover, the ensemble method is computationally efficient and easy to use.

Conclusions: Our results suggest that decision tree based methods are a state-of-the-art, efficient and easy-to-use approach to ORF function prediction.

Background

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, which is the focus of our paper, is often guided by automatic discovery processes which interact with the laboratory experiments.

More precisely, machine learning techniques are used to predict gene functions from a predefined set of possible functions (e.g., the functions in the Gene Ontology). Afterwards, the predictions with highest confidence can be tested in the lab. There are two characteristics of the function prediction task that distinguish it from common machine learning tasks: (1) a single gene may have multiple functions; and (2) the functions are organized in a hierarchy: a gene that is related to some function is automatically related to all its ancestor functions (this is called the hierarchy constraint). This particular problem setting is known in machine learning as hierarchical multi-label classification (HMC) and recently, many approaches have been proposed to deal with it [1-19]. These approaches differ with respect to a number of characteristics: which learning algorithm they are based on, whether the hierarchy constraint is always met and whether they can deal with hierarchies structured as a directed acyclic graph (DAG), such as the Gene Ontology, or are restricted to hierarchies structured as a rooted tree, like MIPS's FunCat.

Decision trees are a well-known type of classifiers that can be learned efficiently from large datasets, produce accurate predictions and can lead to knowledge that provides insight in the biology behind the predictions, as demonstrated by Clare et al. [3]. They have been applied to several machine learning tasks [20]. In earlier work [14], we have investigated how they can be extended to the HMC setting: we presented an HMC

* Correspondence: leander.schietgat@cs.kuleuven.be; celine.vens@cs.kuleuven.be
[1]Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

decision tree learner that takes into account the hierarchy constraint and that is able to process DAG structured hierarchies.

In this article, we show that our HMC decision tree method outperforms previously published approaches applied to *S. cerevisiae* and *A. thaliana*. Our comparisons primarily use precision-recall curves. This evaluation method is well-suited for the HMC tasks considered here, due to the large class skew present in these tasks.

Moreover, we show that by upgrading our method to an ensemble technique, classification performance improves further. Ensemble techniques are learning methods that construct a set of classifiers and classify new data instances by taking a vote over their predictions. Experiments show that ensembles of decision trees outperform Bayesian corrected support vector machines [10], a statistical learning method for gene function prediction, on *S. cerevisiae* data, and methods participating in the MouseFunc challenge [21,22] on *M. musculus* data.

Related work

A number of machine learning approaches have been proposed in the area of functional genomics. They have been applied in the context of gene function prediction in *S. cerevisiae*, *A. thaliana* or *M. musculus*. We have grouped them according to the learning approach they use.

Network based methods

Several 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 [2,4,5,8,15-17]. GENEFAS [4], for example, predicts functions of unannotated yeast genes based on known functions of genes that are nearby in a functional association network. GENEMANIA [15] calculates per gene function a composite functional association network from multiple networks derived from different genomic and proteomic data sources.

These approaches are based on label propagation and do not return a global predictive model. However, a number of approaches were proposed to combine predictions of functional networks with those of a predictive model. Kim et al. [16] combine them with predictions from a Naive Bayes classifier. The combination is based on a simple aggregation function. The Funckenstein system [17] uses logistic regression to combine predictions made by a functional association network with predictions from a random forest.

Kernel based methods

Deng et al. [1] predict gene functions with Markov random fields using protein interaction data. They learn a model for each gene function separately and ignore the hierarchical relationships between the functions. Lanckriet et al. [6] represent the data by means of a kernel function and construct support vector machines for each gene function separately. They only predict top-level classes in the hierarchy. Lee et al. [13] have combined the Markov random field approach of [1] with the SVM approach of [6] by computing diffusion kernels and using them in kernel logistic regression.

Obozinski et al. [19] present a two-step approach in which SVMs are first learned independently for each gene function separately (allowing violations of the hierarchy constraint) and are then reconciliated to enforce the hierarchy constraint. Barutcuoglu et al. [10] have proposed a similar approach where unthresholded support vector machines are learned for each gene function and then combined using a Bayesian network so that the predictions are consistent with the hierarchical relationships. Guan et al. [18] extend this method to an ensemble framework that is based on three classifiers: a classifier that learns a single support vector machine for each gene function, the Bayesian corrected combination of support vector machines mentioned above, and a classifier that constructs a single support vector machine per gene function and per data source and forms a Naive Bayes combination over the data sources.

Methods that learn a separate model for each function have several disadvantages. Firstly, they are less efficient, because n models have to be built (with n the number of functions). Secondly, they often learn from strongly skewed class distributions, which is difficult for many learners.

Decision tree based methods

Clare [23] presents an HMC decision tree method that learns a single tree for predicting gene functions of *S. cerevisiae*. She adapts the well-known decision tree algorithm C4.5 [20] to cope with the issues introduced by the HMC task. First, where C4.5 normally uses class entropy for choosing the best split, her version uses the sum of entropies of the class variables. Second, she extends the method to predict classes on several levels of the hierarchy, assigning a larger cost to misclassifications higher up in the hierarchy. The resulting tree is transformed into a set of rules, and the best rules are selected, based on a significance test performed on a separate validation set. Note that this last step violates the hierarchy constraint, since rules predicting a class can be dropped while rules predicting its subclasses are kept. The non-hierarchical version of her method was later used to predict GO terms for *A. thaliana* [9]. Here, the annotations are predicted for each level of the hierarchy separately.

Hayete and Bienkowska [7] build a decision tree for each GO function separately using information about protein assignments in the same functional domain. As

mentioned earlier, methods that learn separate models for each function have several disadvantages. Moreover, Vens et al. [14] show that in the context of decision trees, separate models are less accurate than a single HMC tree that predicts all functions at once.

Blockeel et al. [24] present to our knowledge the first decision tree approach to HMC that exploits the given class hierarchy and predicts all classes with a single decision tree. Their method is based on the predictive clustering tree framework [25]. This method was first applied to gene function prediction by Struyf et al. [26]. Later, Blockeel et al. [27] propose an improved version of the method and evaluate it on yeast functional genomics data. Vens et al. [14] extend the algorithm towards hierarchies structured as DAGs and show that learning one decision tree for simultaneously predicting all functions outperforms learning one tree per function (even if those trees are built taking into account the hierarchy).

Methods

We first discuss the approach to building HMC trees presented in [14] and then extend it to build ensembles of such trees.

Using predictive clustering trees for HMC tasks

The approach that we present is based on decision trees and is set in the predictive clustering tree (PCT) framework [25]. This framework views a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all training examples, which is recursively partitioned into smaller clusters while moving down the tree. PCTs can be applied to both clustering and prediction tasks. The PCT framework is implemented in the CLUS system, which is available at http://www.cs.kuleuven.be/~dtai/clus.

Before explaining the approach in detail, we show an example of a (partial) predictive clustering tree predicting the functions of *S. cerevisiae* genes from homology data [23] (Figure 1). The homology features are based on a sequence similarity search performed for each yeast gene against all the genes in SwissProt. The functions are taken from the FunCat classification scheme [28]. Each internal node of the tree contains a test on one of the attributes in the dataset. Here, the attributes are binary and have been obtained after preprocessing the relational homology data with a frequent pattern miner. The root node, for instance, tests whether there exists a SwissProt protein that has a high similarity (e-value < $1.0 \cdot 10^{-8}$) with the gene under consideration *G*, is classified into the rhizobiaceae group and has references to the InterPro database. In order to predict the functions of a new gene, the gene is routed down the tree according to the outcome of the tests. When a leaf node is reached, the gene is assigned the functions that are stored in it. Only the

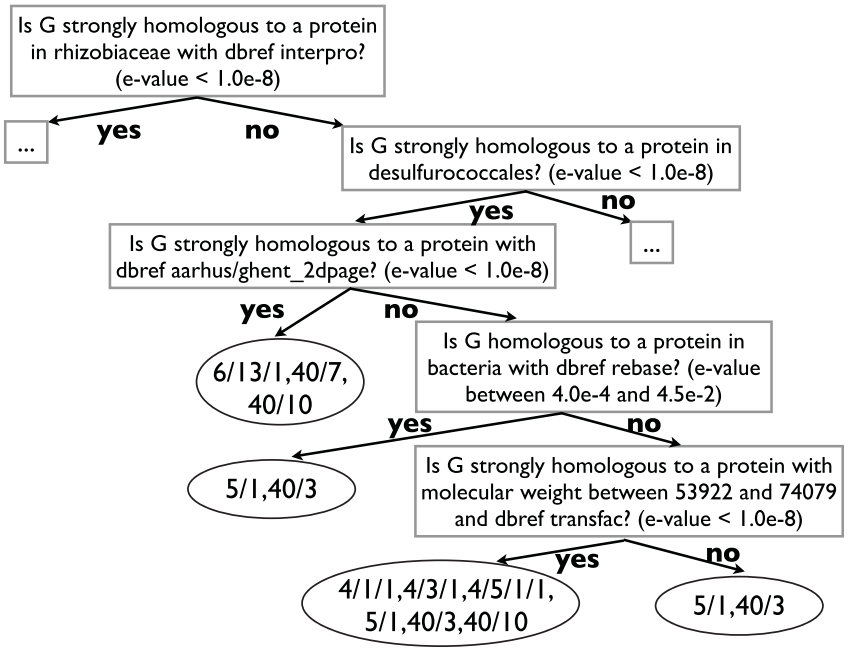


Figure 1 Example of a predictive clustering tree. This tree predicts the functions of a gene *G*, based on homology data. The functions are taken from the FunCat classification scheme and are hierarchical: if for example function 4/3/1 (tRNA synthesis) is predicted, then function 4/3 (tRNA transcription) and function 4 (transcription) are predicted as well.

most specific functions are shown in the figure. In the rest of this section, we explain how PCTs are constructed. A detailed explanation is given in [14].

PCTs [25] can be constructed with a standard "top-down induction of decision trees" (TDIDT) algorithm, similar to CART[29] or C4.5 [20]. The algorithm takes as input a set of training instances (i.e., the genes and their annotations). It searches for the best acceptable test that can be put in a node. If such a test can be found then the algorithm creates a new internal node and calls itself recursively to construct a subtree for each subset (cluster) in the partition induced by the test on the training instances. To select the best test, the algorithm scores the tests by the reduction in variance (which is defined below) they induce on the instances. Maximizing variance reduction maximizes cluster homogeneity and improves predictive performance. If no acceptable test can be found, that is, if no test significantly reduces variance (as measured by a statistical *F*-test), then the algorithm creates a leaf and labels it with a representative case, or prototype, of the given instances.

To apply PCTs to the task of hierarchical multi-label classification, the variance and prototype are defined as follows [14].

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. We define the variance of a set of examples S as the average squared distance between each example's class vector v_k and the set's mean class vector $\bar{v}$, i.e.,

$$Var(S) = \frac{\sum_k d(v_k, \bar{v})^2}{|S|}.$$

In the HMC context, it makes sense to consider similarity at higher levels of the hierarchy more important than similarity at lower levels. To that aim, we use a weighted Euclidean distance

$$d(v_1, v_2) = \sqrt{\sum_i w(c_i) \cdot (v_{1,i} - v_{2,i})^2},$$

where $v_{k,i}$ is the *i*'th component of the class vector v_k of an instance x_k, and the class weights $w(c)$ decrease with the depth of the class in the hierarchy. We choose $w(c) = w_0 \cdot \text{avg}_j \{w(p_j(c))\}$, where $p_j(c)$ denotes the *j*'th parent of class c and $0 < w_0 < 1$). Consider, for example, the class hierarchy shown in Figure 2, and two examples (x_1, S_1) and (x_2, S_2) with $S_1 = \{1, 2, 2/2\}$ and $S_2 = \{2\}$. Using a vector representation with consecutive components representing membership of class 1, 2, 2/1, 2/2 and 3, in that order, we have

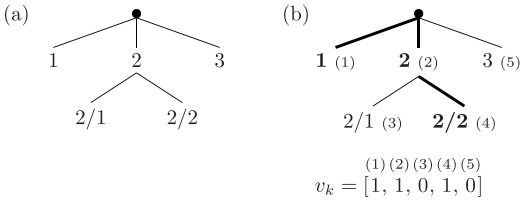


Figure 2 A toy hierarchy. (a) Class label names reflect the position in the hierarchy, e.g., '2/1' is a subclass of '2'. (b) The set of classes {1,2,2/2}, indicated in bold in the hierarchy, and represented as the vector v_k.

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

The heuristic for choosing the best test for a node of the tree is to maximize the variance reduction as discussed before, with the above definition of variance. Note that our definition of $w(c)$ allows the classes to be structured in a DAG, as is the case with the Gene Ontology.

Second, a classification tree stores in a leaf the majority class for that leaf; this class will be the tree's prediction for examples arriving in the leaf. But in our case, since an example may have multiple classes, the notion of "majority class" does not apply in a straightforward manner. Instead, the mean $\bar{v}$ of the class vectors of the examples in that leaf is stored. Recall that $\bar{v}_i$ is the proportion of examples in the leaf belonging to c_i. An example arriving in the leaf can therefore be predicted to belong to class c_i if $\bar{v}_i$ is above some threshold t_i, which can be chosen by the user. To ensure that the predictions obey the hierarchy constraint (whenever a class is predicted its superclasses are also predicted), it suffices to choose $t_i \leq t_j$ whenever c_i is a superclass of c_j. The PCT in Figure 1 has a threshold of $t_i = 0.4$ for all *i*.

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

Ensembles of PCTs

Ensemble methods are learning methods that construct a set of classifiers for a given prediction task and classify new examples by combining the predictions of each classifier. In this paper we consider bagging, an ensemble learning technique that has primarily been used in the context of decision trees. In preliminary experiments, we also considered two other ensemble learning techniques: random forests [30] and an adapted version of the boosting approach for regression trees by Drucker [31]. However, neither method performed better than simple bagging.

Bagging [32] is an ensemble method where the different classifiers are constructed by making bootstrap replicates of the training set and using each of these replicates to construct one classifier. Each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set, until the sample contains the same number of instances as the original training set. The individual predictions given by each classifier can be combined by taking the average (for numeric targets) or the majority vote (for nominal targets).

Breiman has shown that bagging can give substantial gains in the predictive performance of decision tree learners [32]. Also in the case of learning PCTs for predicting multiple targets at once (multi-task learning [33]), decision tree methods benefit from the application of bagging [34]. However, it is clear that, by using bagging on top of the PCT algorithm, the learning time of the model increases significantly, resulting in a clear trade-off between predictive performance and efficiency to be considered by the user.

The algorithm for bagging PCTs takes as input the parameter k, denoting the number of trees in the ensemble. In order to make predictions, the average of all class vectors predicted by the k trees in the ensemble is computed, and then the threshold is applied as before. This ensures that the hierarchy constraint holds. We call the resulting instantiation of the bagging algorithm around the CLUS-HMC algorithm CLUS-HMC-ENS.

Results and discussion

In this section, we address the following questions:

1. How well does CLUS-HMC perform on functional genomics data and what is the improvement, if any, that can be obtained by using CLUS-HMC-ENS on such tasks?

2. How does the predictive performance of the proposed algorithms compare to results reported in the biomedical literature?

In order to answer these questions, we compare our results to the results reported by Clare and King [3] and Barutcuoglu et al. [10] on *S. cerevisiae*, to the results reported by Clare et al. [9] on *A. thaliana*, and to the results of the groups participating in the MouseFunc challenge [21,22] on *M. musculus*. The methods used in these studies were discussed in the "Related work" section.

Datasets

For *S. cerevisiae* and *A. thaliana*, the datasets that we use in our evaluation are exactly those datasets that are used in the cited articles. They are available, together with the parameter settings that can be used to reproduce the results, at the following webpage: http://www.cs.kuleuven.be/~dtai/clus/hmc-ens. For *M. musculus*, the (raw) data is available at http://hugheslab.med.utoronto.ca/supplementary-data/mouseFunc_I/, while the dataset we assembled from it is available at the former webpage.

Next to predicting gene functions of three organisms (*S. cerevisiae*, *A. thaliana*, and *M. musculus*), we consider two annotation schemes in our evaluation: FunCat (developed by MIPS [28]), which is a tree-structured class hierarchy and the Gene Ontology (GO) [35], which forms a directed acyclic graph instead of a tree: each term can have multiple parents.

Saccharomyces cerevisiae

The first dataset we use ($\mathbf{D_0}$) was described by Barutcuoglu et al. [10] and is a combination of different data sources. The input feature vector for a gene consists of pairwise interaction information, membership to colocalization locale, possession of transcription factor binding sites and results from microarray experiments, yielding a dataset with in total 5930 features. The 3465 genes are annotated with function terms from a subset of 105 nodes from the Gene Ontology's *biological process* hierarchy.

We also use the 12 yeast datasets ($\mathbf{D_1}$ - $\mathbf{D_{12}}$) from [23]. The datasets describe different aspects of the genes in the yeast genome. They include five types of bioinformatics data: sequence statistics, phenotype, secondary structure, homology and expression. The different sources of data highlight different aspects of gene function. The genes are annotated with functions from the FunCat classification schemes. Only annotations from the first four levels are given.

$\mathbf{D_1}$ (seq) records sequence statistics that depend on the amino acid sequence of the protein for which the gene codes. These include amino acid frequency ratios, sequence length, molecular weight and hydrophobicity.

$\mathbf{D_2}$ (pheno) contains phenotype data, which represents the growth or lack of growth of knock-out mutants that are missing the gene in question. The gene is removed or disabled and the resulting organism is grown with a variety of media to determine what the modified organism might be sensitive or resistant to.

$\mathbf{D_3}$ (struc) stores features computed from the secondary structure of the yeast proteins. The secondary structure is not known for all yeast genes; however, it can be predicted from the protein sequence with reasonable accuracy, using Prof [36]. Due to the relational nature of secondary structure data, Clare performed a preprocessing step of relational frequent pattern mining; D_3 includes the constructed patterns as binary attributes.

$\mathbf{D_4}$ (hom) includes for each yeast gene, information from other, homologous genes. Homology is usually determined by sequence similarity; here, PSI-BLAST [37] was used to compare yeast genes both with other yeast genes and with all genes indexed in SwissProt v39.

This provided for each yeast gene a list of homologous genes. For each of these, various properties were extracted (keywords, sequence length, names of databases they are listed in, ...). Clare preprocessed this data in a similar way as the secondary structure data to produce binary attributes.

$\mathbf{D_5}$, ..., $\mathbf{D_{12}}$. Many microarray datasets exist for yeast and several of these were used [23]. Attributes for these datasets are real valued, representing fold changes in expression levels.

Arabidopsis thaliana

We use six datasets from [9], originating from different sources: sequence statistics, expression, predicted SCOP class, predicted secondary structure, InterPro and homology. Each dataset comes in two versions: with annotations from the FunCat classification scheme and from the Gene Ontology's *molecular function* hierarchy. Again, only annotations for the first four levels are given. We use the manual annotations for both schemes.

$\mathbf{D_{13}}$ (seq) records sequence statistics in exactly the same way as for *S. cerevisiae*. $\mathbf{D_{14}}$ (exprindiv) contains 43 experiments from NASC's Affymetrix service "Affywatch" http://affymetrix.arabidopsis.info/AffyWatch.html, taking the signal, detection call and detection *p*-values. $\mathbf{D_{15}}$ (scop) consists of SCOP superfamily class predictions made by the Superfamily server [38]. $\mathbf{D_{16}}$ (struc) was obtained in the same way as for *S. cerevisiae*. $\mathbf{D_{17}}$ (interpro) includes features from several motif or signature finding databases, like PROSITE, PRINTS, Pfam, ProDom, SMART and TIGRFAMs, calculated using the EBI's stand-alone InterProScan package [39]. To obtain features, the relational data was mined in the same manner as the structure data. $\mathbf{D_{18}}$ (hom) was obtained in the same way as for *S. cerevisiae*, but now using SwissProt v41.

Mus musculus

We use the data that was provided for the MouseFunc challenge [21,22]. It consists of 21603 genes, of which 1718 are set aside as test genes. Each gene is annotated with GO terms from a specified subset of the Gene Ontology. The annotations are up-propagated using the Gene Ontology's "is-a" and "part-of" relation. The data is composed of several sources: gene expression data, protein sequence pattern annotations, protein-protein interactions, phenotype annotations, phylogenetic profile and disease associations. In order to construct a single dataset ($\mathbf{D_{19}}$), we joined all data tables, removed attributes with fewer than five non-zero values and computed additional attributes that indicate for each gene the classes of other genes to which it is linked through a protein-protein interaction (only considering training set genes). This yields 18746 attributes in total. The resulting representation is similar to the one used by Guan et al. [18].

Methodology

Evaluation measure

We report the performance of the different methods with precision-recall (PR) and ROC [40] based evaluation measures. This is motivated by the following two observations: (1) both measures have been used before to evaluate approaches to gene function prediction [1,8,22], and (2) they both allow to simultaneously compare classifiers for different classification thresholds. Of both measures, PR based evaluation better suits the characteristics of typical HMC datasets, in which many classes are infrequent (i.e., typically only a few genes have a particular function). Viewed as a binary classification task for each class, this implies that for most classes the number of negative instances by far exceeds the number of positive instances. In some cases, it is preferred to recognize the positive instances (i.e., that a gene has a given function), rather than correctly predict the negative ones (i.e., that a gene does not have a particular function). ROC curves are then less suited for this task, exactly because they also reward a learner if it correctly predicts negative instances (giving rise to a low false positive rate). This can present an overly optimistic view of the algorithm's performance [41]. Therefore, unless it is impossible to reconstruct the PR behaviour of the methods we compare to, we report a PR based evaluation.

We use the following definitions of precision, recall, average precision, and average recall:

$$Precision_i = \frac{TP_i}{TP_i + FP_i}, \quad \text{and} \quad Recall_i = \frac{TP_i}{TP_i + FN_i},$$

$$\overline{Precision} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i}, \quad \text{and} \quad \overline{Recall} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i},$$

where i ranges over all functions, TP_i is the number of true positives (correctly predicted positive instances) for function i, FP_i is the number of false positives (positive predictions that are incorrect) for function i, and FN_i is the number of false negatives (positive instances that are incorrectly predicted negative) for function i. Note that these measures ignore the number of correctly predicted negative examples.

A precision-recall curve (PR curve) plots the precision of a model as a function of its recall. We consider two types of PR curves: (1) a function-wise PR curve for a given function i, which plots $Precision_i$ versus $Recall_i$, and (2) an average or pooled PR curve, which plots $\overline{Precision}$ versus $\overline{Recall}$ and summarizes the performance of the model across all functions.

We construct the PR curves as follows. Remember that every leaf in the tree contains a vector $\overline{v}$ with for each function the probability that the gene is predicted to have this function. When decreasing the prediction

threshold t_i from 1 to 0, an increasing number of instances is predicted to belong to c_i, causing the recall to increase whereas precision may increase or decrease (with normally a tendency to decrease). Thus, a single tree (or an ensemble of trees) with a specific threshold has a single precision and recall, and by varying the threshold a PR curve is obtained. Such curves allow us to evaluate the predictive performance of a model regardless of t. In the end, a domain expert can choose the threshold corresponding to the point on the curve that looks most interesting to him.

Although a PR curve helps in understanding the predictive behaviour of the model, a single performance score is more useful to compare models. A score often used to this end is the area between the PR curve and the recall axis, the so-called "area under the PR curve" (AUPRC). The closer the AUPRC is to 1.0, the better the model is. We consider two measures that are based on this idea, that correspond to the two types of PR curves and that are often reported in the literature: AU($\overline{\text{PRC}}$), the area under the average PR curve, and $\overline{\text{AUPRC}}$, the average over all areas under the function-wise PR curves. Note that AU($\overline{\text{PRC}}$) gives more weight to more frequent functions, while $\overline{\text{AUPRC}}$ considers the importance of every function to be equal.

Parameter settings for CLUS-HMC and CLUS-HMC-ENS

In the experiments, w_0, which determines the weights of the different functions in the decision tree heuristic, is set to 0.75 and the number of examples in each decision tree leaf is lower bounded to 5. The parameter k, which denotes the number of trees used in the ensemble, is set to 50. Preliminary experiments show that performance does not strongly depend on the choice of w_0 and that it does not significantly increase after k = 50, so the latter value is a good trade-off between performance and runtime. The significance parameter used in the F-test stopping criterion of CLUS-HMC and CLUS-HMC-ENS is tuned on a separate validation set (1/3 of the training data) and optimized out of 6 possible values (0.001, 0.005, 0.01, 0.05, 0.1, 0.125), maximizing the AU($\overline{\text{PRC}}$). The final model is constructed on the entire training set using the selected value of the significance parameter.

Results

We will first investigate if ensembles improve the predictive performance of CLUS-HMC in gene function prediction and if so, quantify this difference. We will then compare CLUS-HMC and CLUS-HMC-ENS against several state-of-the-art systems in gene function prediction. On the one hand, we will compare CLUS-HMC to C4.5H/M [3,9], because they both build a single decision tree. On the other hand, we will compare CLUS-HMC-ENS to Bayesian-corrected SVMs [10], a statistical learning approach, on D_0, and to the methods that entered the MouseFunc challenge on D_{19}.

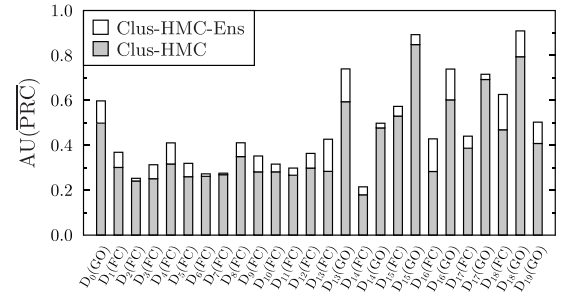


Figure 3 Comparison of AU($\overline{\text{PRC}}$) between Clus-HMC and Clus-HMC-Ens. The white surface represents the gain in AU($\overline{\text{PRC}}$) obtained by CLUS-HMC-ENS.

The datasets originating from [3,9] (i.e., datasets D_1 to D_{18}) are divided into a training set (2/3) and a test set (1/3). We use exactly the same splits. For dataset D_0, we randomly construct a training and test set with the same ratio. For dataset D_{19}, we use the same training and test sets that were used in the MouseFunc challenge.

Comparison between CLUS-HMC and CLUS-HMC-ENS

For each of the datasets, the AU($\overline{\text{PRC}}$) of CLUS-HMC and CLUS-HMC-ENS is shown in Figure 3. We see that for every dataset, there is an increase in AU($\overline{\text{PRC}}$) when using ensembles. The average gain is 0.071 (which is an improvement of 18% on average); the maximal gain is 0.157. Representative PR curves can be found in Figures 4, 5 and 6. Figure 7 shows the $\overline{\text{AUPRC}}$ of CLUS-HMC and CLUS-HMC-ENS. Again, there is an increase in $\overline{\text{AUPRC}}$ when using ensembles, with an average gain of 0.093 (which is an improvement of 108% on average) and a maximal gain of 0.337. These results show that the increase in performance obtained by CLUS-HMC-ENS is larger according to $\overline{\text{AUPRC}}$ than according to AU($\overline{\text{PRC}}$), which indicates that ensembles are performing particularly better for the less frequent classes, typically occurring at the lower levels of the hierarchy. To summarize, the improvement in predictive performance that can be obtained by using tree ensembles in more straightforward machine learning settings carries over to the HMC setting with functional genomics data.

Comparison between CLUS-HMC and C4.5H/M

We now concentrate on the comparison of the results obtained by our algorithms to those obtained by other decision tree based algorithms. For the datasets that are annotated with FunCat classes (D_1 - D_{18}), we will compare to the hierarchical extension of C4.5 [3], which we will refer to as C4.5H. For the datasets with GO annotations (D_{13} - D_{18}), we will use the non-hierarchical multi-label extension of C4.5 [9], as C4.5H cannot handle hierarchies structured as a DAG. We refer to this system as C4.5M.

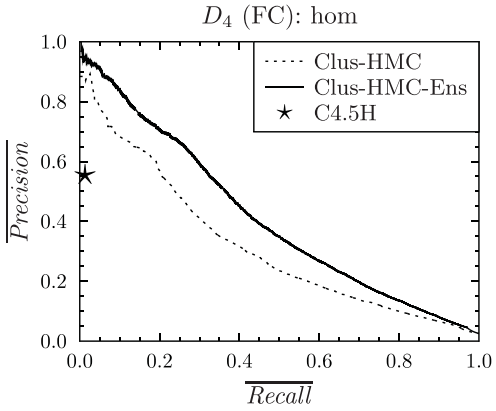


Figure 4 Precision-recall curve for all classes for C4.5H, Clus-HMC and Clus-HMC-Ens on D_4 with FunCat annotations.

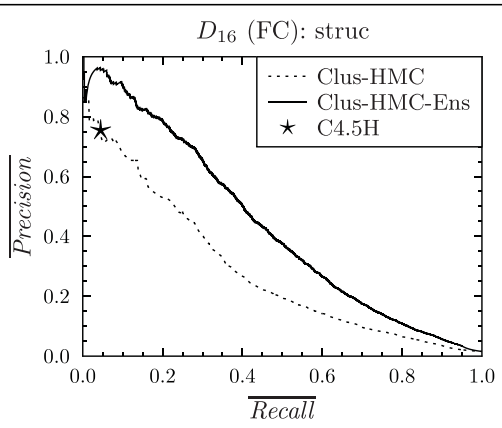


Figure 5 Precision-recall curve for all classes for C4.5H, Clus-HMC and Clus-HMC-Ens on D_{16} with FunCat annotations.

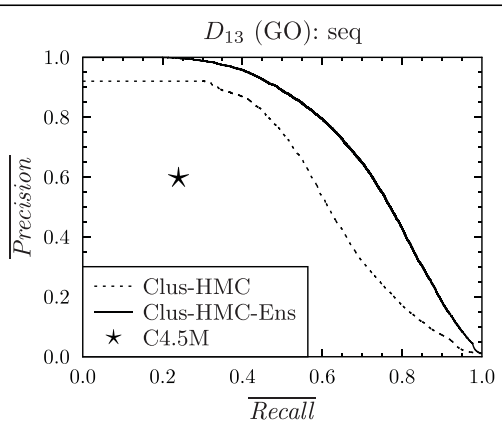


Figure 6 Precision-recall curve for all classes for C4.5M, Clus-HMC and Clus-HMC-Ens on D_{13} with GO annotations.

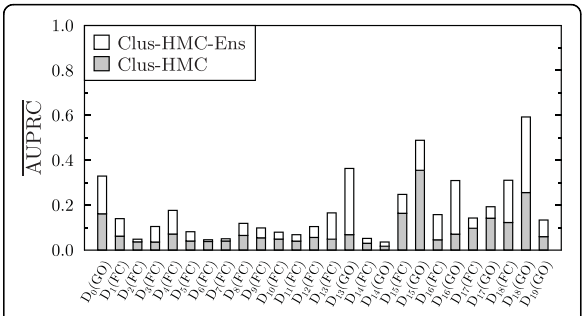


Figure 7 Comparison of $\overline{\text{AUPRC}}$ between Clus-HMC and Clus-HMC-Ens. The white surface represents the gain in $\overline{\text{AUPRC}}$ obtained by CLUS-HMC-ENS.

For their experiments on *A. thaliana*, Clare et al. [9] only report results per level of the hierarchy. In order to obtain these results, they learn a separate classifier per level, removing from their training and test set those genes that do not have annotated functions at that level. This approach may give a biased result: when annotating a new gene, it is not known in advance at which levels of the hierarchy it will have functions. Therefore, we reran C4.5M to learn one classifier that uses all training data and tested it on the complete test set.

For evaluating their systems, Clare et al. [3,9] report precision. Indeed, as the biological experiments required to validate the learned rules are costly, it is important to avoid false positives. However, precision is always traded off by recall: a classifier that predicts one example positive, but misses 1000 other positive examples may have a precision of 1, although it can hardly be called a good classifier. Therefore, we also compute the recall of the models obtained by C4.5H/M. These models were presented as rules for specific classes without any probability scores, so each model corresponds to precisely one point in PR space.

For each of the datasets D_1 - D_{18}, these PR points are plotted against the average PR curves for CLUS-HMC. As we are comparing curves with points, we speak of a "win" for CLUS-HMC when its curve is above C4.5H/M's point, and of a "loss" when it is below the point. Under the null hypothesis that both systems perform equally well, we expect as many wins as losses. We observed that only in one case out of 24, for dataset D_{16} with FunCat annotations, C4.5H/M outperforms CLUS-HMC. For all other cases there is a clear win for CLUS-HMC. Representative PR curves can be found in Figures 4, 5 and 6.

For each of these datasets, we also compared the precision of C4.5H/M, CLUS-HMC and CLUS-HMC-ENS, at the recall obtained by C4.5H/M. The results can be found in Figure 8. The average gain in precision w.r.t. C4.5H/M is 0.209 for CLUS-HMC and 0.276 for CLUS-HMC-ENS.

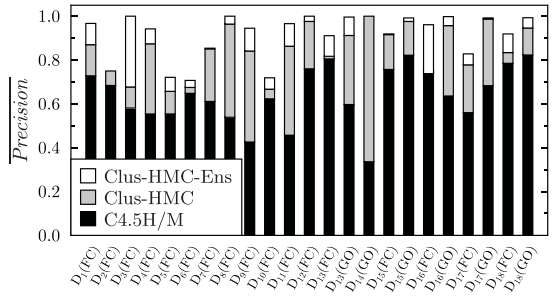


Figure 8 Comparison of precision between C4.5H/M, Clus-HMC and Clus-HMC-Ens, at the recall obtained by C4.5H/M. The gray surface represents the gain in precision obtained by CLUS-HMC, the white surface represents the gain for CLUS-HMC-ENS. D_{14}(FC) was not included, since C4.5H did not find significant rules. For D_{16}(FC), C4.5H scored a slightly better precision (see Figure 5), hence the lack of gray surface.

We can conclude that CLUS-HMC is the tree-building system that yields the best predictive performance. Compared with other existing methods, we are able to obtain the same precision with higher recall, or the same recall with higher precision. Moreover, the hierarchy constraint is always fulfilled, which is not the case for C4.5H/M.

Comparing individual rules

Every leaf of a decision tree corresponds to an *if ... then ...* rule. When comparing the complexity and precision/recall of these individual rules, CLUS-HMC also performs well. For instance, take FunCat class 29, which has a prior frequency of 3%. Figure 9 shows the PR evaluation for the algorithms for this class using homology dataset D_4. The PR point for C4.5H corresponds to one rule, shown in Figure 10. This rule has a precision/recall of 0.55/0.17.

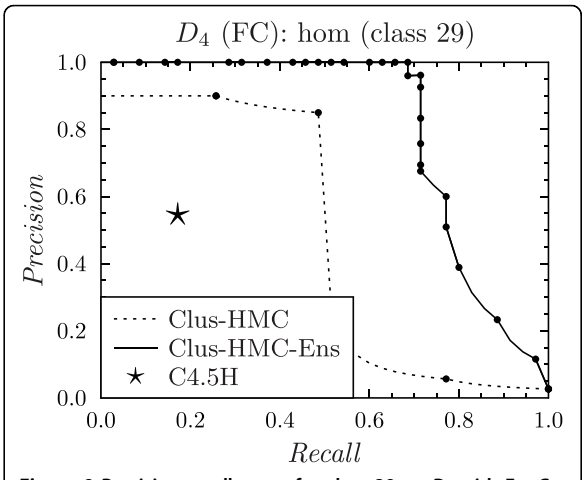


Figure 9 Precision-recall curve for class 29 on D_4 with FunCat annotations.

CLUS-HMC's most precise rule for class 29 is shown in Figure 11. This rule has a precision/recall of 0.90/0.26.

Note from Figure 9 that an even higher precision can be obtained with CLUS-HMC-ENS, although the rules which lead to this prediction are more complex.

Comparison between CLUS-HMC-ENS and Bayesian-corrected SVMs

In this section, we compare CLUS-HMC-ENS to the statistical learning method of Barutcuoglu et al. [10], which consists of Bayesian-corrected SVMs (see "Related work"). We will further refer to this method as BSVM. The authors have used dataset D_0 to evaluate their method and report class-wise area under the ROC convex hull (AUROC) for a small subset of 105 nodes of the Gene Ontology. As only AUROC scores are reported by Barutcuoglu et al. [10], we adopt the same evaluation metric for this comparison.

Barutcuoglu et al. [10] build a bagging procedure around their system and report out-of-bag error estimates [42] as evaluation, which removes the need for a set-aside test set. Out-of-bag error estimation proceeds as follows: for each example in the original training set, the predictions are made by aggregating only over those classifiers for which the example was not used for training. This is the out-of-bag classifier. The out-of-bag error estimate is then the error rate of the out-of-bag classifier on the training set. The number of bags used in this procedure was 10. To compare our results, we use exactly the same method.

On dataset D_0, the average of the AUROC over the 105 functions is 0.871 for CLUS-HMC-ENS and 0.854 for BSVM. Figure 12 compares the class-wise out-of-bag AUROC estimates for CLUS-HMC-ENS and BSVM outputs. CLUS-HMC-ENS scores better on 73 of the 105 functions, while BSVM scores better on the remaining 32 cases. According to the (two-sided) Wilcoxon signed rank test [43], the performance of CLUS-HMC-ENS is significantly better ($p = 4.37 \cdot 10^{-5}$).

Moreover, CLUS-HMC-ENS is faster than BSVM. Runtimes are compared for one of the datasets having annotations from Gene Ontology's complete *biological process* hierarchy (in particular, we used D_{16}, which is annotated with 629 classes). Run on a cluster of AMD Opteron processors (1.8 - 2.4 GHz, ≥ 2 GB RAM), CLUS-HMC-ENS required 15.9 hours, while SVM-light [44], which is the first step of BSVM, required 190.5 hours for learning the models (i.e., CLUS-HMC-ENS is faster by a factor 12 in this case).

Comparison between CLUS-HMC-ENS and the methods in the MouseFunc challenge

In this section we compare CLUS-HMC-ENS to the seven systems that submitted predictions to the MouseFunc challenge. These systems are the ensemble extension of BSVM [18] (which we will call BSVM$^+$), Kernel

if	the ORF is NOT homologous to another yeast protein ($e \geq 0.73$) and homologous to a protein in rhodospirillaceae ($e < 1.0 \cdot 10^{-8}$)
and	NOT homologous to another yeast protein ($5.0 \cdot 10^{-4} < e < 3.3 \cdot 10^{-2}$) and homologous to a protein in anabaena ($e \geq 1.1$)
and	homologous to another yeast protein ($2.0 \cdot 10^{-7} < e < 5.0 \cdot 10^{-4}$) and homologous to a protein in beta_subdivision ($e < 1.0 \cdot 10^{-8}$)
and	NOT homologous to a protein in sinorhizobium with keyword transmembrane ($e \geq 1.1$)
and	NOT homologous to a protein in entomopoxvirinae with dbref pir ($e \geq 1.1$)
and	NOT homologous to a protein in t4-like_phages with molecular weight between 1485 and 38502 ($4.5 \cdot 10^{-2} < e < 1.1$)
and	NOT homologous to a protein in chroococcales with dbref prints ($1.0 \cdot 10^{-8} < e < 4.0 \cdot 10^{-4}$)
and	NOT homologous to a protein with sequence length between 344 and 483 and dbref tigr ($e < 1.0 \cdot 10^{-8}$)
and	homologous to a protein in beta_subdivision with sequence length between 16 and 344 ($e < 1.0 \cdot 10^{-8}$)
then	class 29/0/0/0 "transposable elements, viral and plasmid proteins"

Figure 10 Rule found by C4.5H on the D_4 (FC) homology dataset, with a precision of 0.55 and a recall of 0.17.

if	the ORF is NOT homologous to a protein in rhizobiaceae_group with dbref interpro ($e < 1.0 \cdot 10^{-8}$)
and	NOT homologous to a protein in desulfurococcales ($e < 1.0 \cdot 10^{-8}$)
and	homologous to a protein in ascomycota with dbref transfac ($e < 1.0 \cdot 10^{-8}$)
and	homologous to a protein in viridiplantae with sequence length ≥ 970 ($e < 1.0 \cdot 10^{-8}$)
and	homologous to a protein in rhizobium with keyword plasmid ($1.0 \cdot 10^{-8} < e < 4.0 \cdot 10^{-4}$)
and	homologous to a protein in nicotiana with dbref interpro ($e < 1.0 \cdot 10^{-8}$)
then	class 29/0/0/0 "transposable elements, viral and plasmid proteins"

Figure 11 Rule found by Clus-HMC on the D_4 (FC) homology dataset, with a precision of 0.90 and a recall of 0.26.

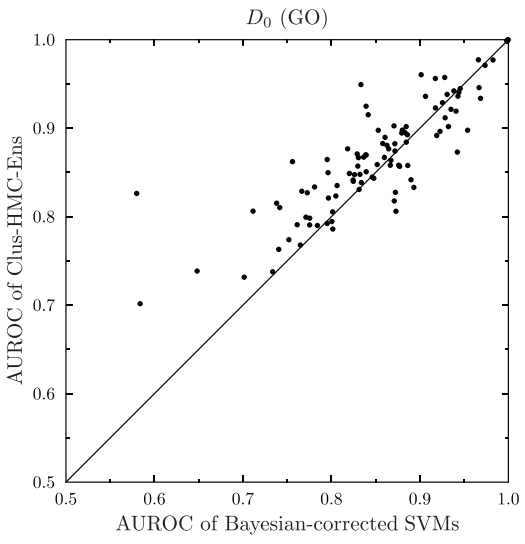


Figure 12 Class-wise out-of-bag AUROC comparison between Clus-HMC-Ens and Bayesian-corrected SVMs.

Logistic Regression [13] (which we will call KLR), calibrated SVMs [19] (which we will call CSVM), GENEFAS [4], GENEMANIA [15], the combined functional network and classifier strategy of Kim et al. [16] (which we will call KIM) and the Funckenstein system [17]. These methods were described in the "Related work" section. Note that, when comparing the results, one should keep in mind that each team independently constructed a dataset, possibly using different features. As a result, the differences in performance can be due not only to the learning methods compared, but also the different feature sets used by the methods. As mentioned in the "Datasets" section, the representation that we use is the one of the BSVM$^+$ team.

The organizers have made available a program that computes several evaluation measures and was used to compare the results by the different participating teams in the challenge. This software is available at the same URL where the data can be found, and computes AUROC scores and precision values at several levels of recall for a list of GO terms.

A close inspection of this program reveals that it exhibits some undesirable behaviour. This can easily be verified by observing the result for a classifier that always predicts the same value. The correct function-wise PR curve for any GO term would be a straight line parallel to the recall axis, with precision equal to the frequency of the term. However, the PR curve returned by the software differs from this. If the ordering in which the genes are processed happens to start with a positive gene, then the precision at zero recall equals one. Moreover, if the ordering ends with a negative gene, the precision at recall one is still higher than the class frequency. The ordering in which the examples are processed should be independent from the resulting PR curve.

For this reason, we included the computation of precision and recall in the Clus software. Because the MouseFunc website lists a prediction matrix (containing for each gene-term pair the corresponding probability that the gene is annotated with the GO term) for each of the methods we compare to, we can run our own evaluation program on these predictions, producing corrected results for these methods.

Each method gives predictions for 2815 selected GO terms. These terms are divided into 12 disjunct subsets corresponding to all combinations of the three GO branches (Biological Process, Molecular Function and Cellular Component) with four ranges of specificity, which is defined as the number of genes in the training set to which each term is annotated (3-10, 11-30, 31-100 and 101-300). We have adopted the same subsets and trained and evaluated our models on each of them. Since 1846 of the selected 2815 GO terms were used as annotation in the test set, our evaluation of all the systems is based only on those.

Table 1 shows the AU($\overline{\text{PRC}}$) results of all the methods on the 12 subsets. Looking at the wins/losses for each of the 12 subsets, according to the (two-sided) Wilcoxon signed rank test, the performance of CLUS-HMC-ENS is significantly better at the 1% level than BSVM$^+$($p = 4.88{\cdot}10^{-4}$), CSVM ($p = 1.47{\cdot}10^{-3}$), GENEFAS ($p = 4.88{\cdot}10^{-4}$), and KIM ($p = 4.88{\cdot}10^{-4}$). CLUS-HMC-ENS has more wins than KLR ($p = 1.61{\cdot}10^{-2}$) and GENEMANIA ($p = 1.61{\cdot}10^{-2}$), but is not significantly better at 1%. CLUS-HMC-ENS is performing significantly worse than Funckenstein ($p = 9.28{\cdot}10^{-3}$).

Table 2 shows the same comparison, but now for $\overline{\text{AUPRC}}$. According to the Wilcoxon signed rank test, CLUS-HMC-ENS is performing significantly better at the 1% level than KIM ($p = 4.88{\cdot}10^{-4}$), while it is not significantly different from BSVM$^+$ ($p = 4.70{\cdot}10^{-1}$), KLR ($p = 1.61{\cdot}10^{-2}$), CSVM ($p = 1.51{\cdot}10^{-1}$) and GENEFAS ($p = 2.59{\cdot}10^{-2}$). CLUS-HMC-ENS is performing significantly worse than GENEMANIA ($p = 9.28{\cdot}10^{-3}$) and Funckenstein ($p = 9.77{\cdot}10^{-4}$).

Because $\overline{\text{AUROC}}$, the average over all areas under the function-wise ROC curves, was used as evaluation measure in the MouseFunc challenge [22], we report it in Table 3. According to the Wilcoxon signed rank test, CLUS-HMC-ENS is not performing significantly different at the 1% level than KLR ($p = 9.10{\cdot}10^{-1}$), CSVM ($p = 2.20{\cdot}10^{-2}$), GENEFAS ($p = 5.69{\cdot}10^{-1}$) and KIM ($p = 3.22{\cdot}10^{-2}$). CLUS-HMC-ENS is performing significantly worse than BSVM$^+$ ($p = 4.88{\cdot}10^{-4}$), GENEMANIA ($p = 9.77{\cdot}10^{-4}$) and Funckenstein ($p = 9.77\ 10^{-4}$).

The fact that CLUS-HMC-ENS performs better according to AU($\overline{\text{PRC}}$) than to $\overline{\text{AUPRC}}$ and $\overline{\text{AUROC}}$ can be explained as follows. The variance function used to select the best tests gives a higher weight to functions at higher levels of the hierarchy (see "Methods" section), causing CLUS-HMC-ENS to perform well especially on those functions. In contrast to $\overline{\text{AUPRC}}$ and $\overline{\text{AUROC}}$, which consider each function as equal, the AU($\overline{\text{PRC}}$)

Table 1 Comparison of AU($\overline{\text{PRC}}$) between Clus-HMC-Ens and the MouseFunc systems

Subset	CLUS-HMC-ENS	BSVM$^+$	KLR	CSVM	GENEFAS	GeneMANIA	KIM	Funckenstein
BP_3-10	0.045	0.040⊖	0.028⊖	0.029⊖	0.028⊖	0.071⊕	0.029⊖	0.085⊕
BP_11-30	0.055	0.042⊖	0.053	0.017⊖	0.012⊖	0.038⊖	0.031⊖	0.083⊕
BP_31-100	0.109	0.100⊖	0.135⊕	0.077⊖	0.033⊖	0.035⊖	0.044⊖	0.190⊕
BP_101-300	0.173	0.161⊖	0.174⊕	0.146⊖	0.078⊖	0.055⊖	0.051⊖	0.225⊕
CC_3-10	0.182	0.076⊖	0.060⊖	0.046⊖	0.050⊖	0.131⊖	0.128⊖	0.202⊕
CC_11-30	0.207	0.085⊖	0.128⊖	0.094⊖	0.038⊖	0.068⊖	0.112⊖	0.167⊖
CC_31-100	0.233	0.163⊖	0.161⊖	0.074⊖	0.107⊖	0.046⊖	0.127⊖	0.226⊖
CC_101-300	0.220	0.166⊖	0.225⊕	0.157⊖	0.110⊖	0.101⊖	0.094⊖	0.248⊕
MF_3-10	0.266	0.243⊖	0.191⊖	0.205⊖	0.174⊖	0.359⊕	0.189⊖	0.368⊕
MF_11-30	0.356	0.258⊖	0.285⊖	0.275⊖	0.136⊖	0.270⊖	0.215⊖	0.384⊕
MF_31-100	0.360	0.245⊖	0.294⊖	0.231⊖	0.120⊖	0.284⊖	0.191⊖	0.482⊕
MF_101-300	0.368	0.283⊖	0.331⊖	0.386⊕	0.184⊖	0.202⊖	0.140⊖	0.485⊕

For each of the 12 subsets, the AU($\overline{\text{PRC}}$) of CLUS-HMC-ENS is compared with the MouseFunc systems. A win (⊕) means that the MouseFunc system outperforms CLUS-HMC-ENS, a loss (⊖) means that it is outperformed by CLUS-HMC-ENS.

Table 2 Comparison of $\overline{\text{AUPRC}}$ between CLUS-HMC-ENS and the MouseFunc systems

Subset	CLUS-HMC-ENS	BSVM$^+$	KLR	CSVM	GENEFAS	GENEMANIA	KIM	Funckenstein
BP_3-10	0.120	0.156⊕	0.075⊖	0.075⊖	0.108⊖	0.170⊕	0.108⊖	0.198⊕
BP_11-30	0.110	0.141⊕	0.087⊖	0.085⊖	0.074⊖	0.151⊕	0.107⊖	0.162⊕
BP_31-100	0.139	0.172⊕	0.158⊕	0.140⊕	0.094⊖	0.177⊕	0.116⊖	0.244⊕
BP_101-300	0.171	0.172⊕	0.169⊖	0.173⊕	0.104⊖	0.160⊖	0.056⊖	0.214⊕
CC_3-10	0.319	0.249⊖	0.119⊖	0.083⊖	0.233⊖	0.324⊕	0.271⊖	0.316⊖
CC_11-30	0.260	0.194⊖	0.212⊖	0.151⊖	0.131⊖	0.235⊖	0.178⊖	0.267⊕
CC_31-100	0.217	0.232⊕	0.197⊖	0.161⊖	0.191⊖	0.261⊕	0.144⊖	0.287⊕
CC_101-300	0.244	0.217⊖	0.259⊕	0.221⊖	0.177⊖	0.258⊕	0.118⊖	0.279⊕
MF_3-10	0.320	0.441⊕	0.258⊖	0.228⊖	0.427⊕	0.465⊕	0.304⊖	0.472⊕
MF_11-30	0.356	0.373⊕	0.347⊖	0.393⊕	0.350⊖	0.401⊕	0.302⊖	0.455⊕
MF_31-100	0.269	0.289⊕	0.230⊖	0.278⊕	0.242⊖	0.291⊕	0.255⊖	0.416⊕
MF_101-300	0.322	0.317⊖	0.321⊖	0.374⊕	0.295⊖	0.391⊕	0.172⊖	0.441⊕

For each of the 12 subsets, the $\overline{\text{PRC}}$ of CLUS-HMC-ENS is compared with the MouseFunc systems. A win (⊕) means that the MouseFunc system outperforms CLUS-HMC-ENS, a loss (⊖) means that it is outperformed by CLUS-HMC-ENS.

Table 3 Comparison of $\overline{\text{AUROC}}$ between Clus-HMC-Ens and the MouseFunc systems

Subset	CLUS-HMC-ENS	BSVM$^+$	KLR	CSVM	GENEFAS	GENEMANIA	KIM	Funckenstein
BP_3-10	0.695	0.808⊕	0.581⊖	0.588⊖	0.715⊕	0.873⊕	0.813⊕	0.790⊕
BP_11-30	0.748	0.808⊕	0.741⊖	0.659⊖	0.767⊕	0.849⊕	0.822⊕	0.796⊕
BP_31-100	0.831	0.874⊕	0.846⊕	0.778⊖	0.780⊖	0.872⊕	0.851⊕	0.880⊕
BP_101-300	0.823	0.853⊕	0.845⊕	0.813⊖	0.733⊖	0.840⊕	0.795⊖	0.838⊕
CC_3-10	0.748	0.845⊕	0.571⊖	0.618⊖	0.782⊕	0.899⊕	0.865⊕	0.837⊕
CC_11-30	0.791	0.873⊕	0.790⊖	0.785⊖	0.834⊕	0.907⊕	0.846⊕	0.850⊕
CC_31-100	0.863	0.896⊕	0.850⊖	0.851⊖	0.783⊖	0.887⊕	0.863	0.849⊖
CC_101-300	0.845	0.873⊕	0.851⊕	0.821⊖	0.750⊖	0.842⊖	0.808⊖	0.867⊕
MF_3-10	0.818	0.887⊕	0.630⊖	0.681⊖	0.850⊕	0.951⊕	0.880⊕	0.879⊕
MF_11-30	0.842	0.903⊕	0.861⊕	0.836⊖	0.865⊕	0.936⊕	0.884⊕	0.909⊕
MF_31-100	0.838	0.888⊕	0.892⊕	0.881⊕	0.843⊕	0.887⊕	0.884⊕	0.903⊕
MF_101-300	0.874	0.904⊕	0.894⊕	0.884⊕	0.843⊖	0.909⊕	0.844⊖	0.918⊕

For each of the 12 subsets, the $\overline{\text{PRC}}$ of CLUS-HMC-ENS is compared with the MouseFunc systems. A win (⊕) means that the MouseFunc system outperforms CLUS-HMC-ENS, a loss (⊖) means that it is outperformed by CLUS-HMC-ENS.

evaluation measure shares the idea of giving a higher penalty to mistakes made for functions at higher levels of the hierarchy.

We can conclude that, in general, the performance of CLUS-HMC-ENS is not significantly different from that of BSVM$^+$, which has been evaluated on the same dataset. Moreover, also compared to the other systems, which have used other preprocessing methods, CLUS-HMC-ENS is competitive: only the Funckenstein method and GENEMANIA produce significantly better results on 3 and 2 evaluation measures, respectively. In a function-wise comparison over all 12 subsets (1846 functions in total), CLUS-HMC-ENS still performed better than Funckenstein on 607 (according to AUPRC) and 625 (according to AUROC) functions, while it had an equal score for 98 (AUPRC) and 97 (AUROC) functions. Similarly, it performed better than GENEMANIA on 645/563 functions and had an equal score for 84/88 functions, respectively. This shows that none of the methods is guaranteed to be the best choice for any given function.

This comparison to the methods in the MouseFunc competition suggests that incorporating functional linkage information in the predictions made by an ensemble method can substantially improve its performance. How this could be achieved for CLUS-HMC-ENS will be investigated in further work.

Conclusions

In this article, we have presented the use of a decision tree learner, called CLUS-HMC, in functional genomics. The learner produces a single tree that predicts, for a given gene, its biological functions from a function classification scheme, such as the Gene Ontology. The main contributions of this work are the introduction of the tree-based ensemble learner CLUS-HMC-ENS and

empirical evidence showing that this learner outperforms several state-of-the-art methods on *S. cerevisiae*, *A. thaliana* and *M. musculus* datasets.

First, we have shown that CLUS-HMC outperforms an existing decision tree learner (C4.5H/M) w.r.t. predictive performance. Second, we have shown that the predictive performance boost in regular classification tasks obtained by using ensembles, carries over to the hierarchical multi-label classification context, in which the gene function prediction task is set. Third, by constructing an ensemble of CLUS-HMC-trees, our method outperforms a statistical learner based on SVMs for *S. cerevisiae*, both in predictive performance and in efficiency. Fourth, this ensemble learner is competitive to statistical and network based methods for *M. musculus* data.

To summarize, CLUS-HMC can give additional biological insight in the predictions. Moreover, CLUS-HMC-ENS yields state-of-the-art quality for gene function prediction. The software implementing these methods is easy to use and available online as open-source software. As such, CLUS-HMC(-ENS) is competitive to the current state-of-the-art systems and therefore, we believe it should be considered for making automated predictions in functional genomics.

Acknowledgements
Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) to LS. Research Fund K.U.Leuven to CV and JS. Research Foundation - Flanders (FWO-Vlaanderen) to CV and HB. The GOA Probabilistic Logic Learning to CV. The EU funded project IQ (Inductive Queries for Mining Patterns and Models). The authors thank Amanda Clare and Zafer Barutcuoglu for their cooperation. This research was conducted utilizing high performance computational resources provided by the K.U. Leuven http://ludit.kuleuven.be/hpc.

Author details
[1]Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium. [2]Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia.

Authors' contributions
LS and CV performed the experimental analysis and drafted the manuscript. JS provided expertise about CLUS-HMC and helped revising the manuscript. HB supervised the study and helped drafting the manuscript. DK developed CLUS-HMC-ENS under the supervision of SD. SD also helped in acquiring the datasets used in the study and provided input to various parts of the manuscript. All authors read and approved the final manuscript.

Received: 8 January 2009
Accepted: 2 January 2010 Published: 2 January 2010

References

1. Deng M, Zhang K, Mehta S, Chen T, Sun F: **Prediction of protein function using protein-protein interaction data.** *Proceedings of the IEEE Computer Society Bioinformatics Conference, IEEE Computer Society* 2002, 197-206.
2. Troyanskaya O, Dolinski K, Owen A, Altman R, D B: **A Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces Cerevisiae).** *Proceedings of the National Academy of Sciences* 2003, **100(14)**:8348-8353.
3. Clare A, King RD: **Predicting gene function in Saccharomyces cerevisiae.** *Bioinformatics* 2003, **19(Suppl 2)**:ii42-49.
4. Chen Y, Xu D: **Global protein function annotation through mining genome-scale data in yeast Saccharomyces cerevisiae.** *Nucleic Acids Research* 2004, **32(21)**:6414-6424.
5. Karaoz U, Murali T, Letovsky S, Zheng Y, Ding C, Cantor C, Kasif S: **Whole-genome annotation by using evidence integration in functional-linkage networks.** *Proceedings of the National Academy of Sciences* 2004, **101(9)**:2888-2893.
6. Lanckriet GR, Deng M, Cristianini N, Jordan MI, Noble WS: **Kernel-based data fusion and its application to protein function prediction in yeast.** *Proceedings of the Pacific Symposium on Biocomputing* 2004, 300-311.
7. Hayete B, Bienkowska J: **GOTrees: Predicting GO associations from protein domain composition using decision trees.** *Pacific Symposium on Biocomputing* World ScientificAltman RB, Jung TA, Klein TE, Dunker AK, Hunter L 2005, 127-138.
8. Chua H, Sung W, Wong L: **Exploiting indirect neighbours and topological weight to predict protein function from protein-protein interactions.** *Bioinformatics* 2006, **22(13)**:1623-1630.
9. Clare A, Karwath A, Ougham H, King RD: **Functional bioinformatics for Arabidopsis thaliana.** *Bioinformatics* 2006, **22(9)**:1130-1136.
10. Barutcuoglu Z, Schapire R, Troyanskaya O: **Hierarchical multi-label prediction of gene function.** *Bioinformatics* 2006, **22(7)**:830-836.
11. Cesa-Bianchi N, Gentile C, Zaniboni L: **Incremental algorithms for hierarchical classification.** *Journal of Machine Learning Research* 2006, **7**:31-54.
12. Rousu J, Saunders C, Szedmak S, Shawe-Taylor J: **Kernel-based learning of hierarchical multilabel classification models.** *Journal of Machine Learning Research* 2006, **7**:1601-1626.
13. Lee H, Tu Z, Deng M, Sun F, Chen T: **Diffusion kernel-based logistic regression models for protein function prediction.** *OMICS* 2006, **10**:40-55.
14. Vens C, Struyf J, Schietgat L, Džeroski S, Blockeel H: **Decision trees for hierarchical multi-label classification.** *Machine Learning* 2008, **73(2)**:185-214.
15. Mostafavi S, Ray D, Warde-Farley D, Grouios C, Morris Q: **GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function.** *Genome Biology* 2008, **9(Suppl 1)**:S4.
16. Kim W, Krumpelman C, Marcotte E: **Inferring mouse gene functions from genomic-scale data using a combined functional network/classification strategy.** *Genome Biology* 2008, **9(Suppl 1)**:S5.
17. Tian W, Zhang L, Tasan M, Gibbons F, King O, Park J, Wunderlich Z, Cherry J, Roth F: **Combining guilt-by-association and guilt-by-profiling to predict Saccharomyces cerevisiae gene function.** *Genome Biology* 2008, **9(Suppl 1)**:S7.
18. Guan Y, Myers C, Hess D, Barutcuoglu Z, Caudy A, Troyanskaya O: **Predicting gene function in a hierarchical context with an ensemble of classifiers.** *Genome Biology* 2008, **9(Suppl 1)**:S3.
19. Obozinski G, Lanckriet G, Grant C, Jordan M, Noble W: **Consistent probabilistic outputs for protein function prediction.** *Genome Biology* 2008, **9(Suppl 1)**:S6.
20. Quinlan J: *C4.5: Programs for Machine Learning* Morgan Kaufmann series in Machine Learning, Morgan Kaufmann, Springer Netherlands 1993.
21. Hughes T, Roth F: **A race through the maze of genomic evidence.** *Genome Biology* 2008, **9(Suppl 1)**:S1.
22. Pena-Castillo L, Tasan M, Myers C, Lee H, Joshi T, Zhang C, Guan Y, Leone M, A P, Kim W, Krumpelman C, Tian W, Obozinski G, Qi Y, Mostafavi S, Lin G, Berriz G, Gibbons F, Lanckriet G, Qiu J, Grant C, Barutcuoglu Z, Hill D, Warde-Farley D, Grouios C, Ray D, Blake J, Deng M, Jordan M, Noble W, Morris Q, Klein-Seetharaman J, Bar-Joseph Z, Chen T, Sun F, Troyanskaya O, Marcotte E, Xu D, Hughes T, Roth F: **A critical assessment of Mus musculus gene function prediction using integrated genomic evidence.** *Genome Biology* 2008, **9(Suppl 1)**:S2.
23. Clare A: **Machine learning and data mining for yeast functional genomics.** *PhD thesis* University of Wales, Aberystwyth, Computer Science Department 2003.
24. Blockeel H, Bruynooghe M, Džeroski S, Ramon J, Struyf J: **Hierarchical multi-classification.** *Proceedings of the ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining* 2002, 21-35.
25. Blockeel H, De Raedt L, Ramon J: **Top-down induction of clustering trees.** *Proceedings of the 15th International Conference on Machine Learning* 1998, 55-63.

26. Struyf J, Džeroski S, Blockeel H, Clare A: **Hierarchical multi-classification with predictive clustering trees in functional genomics.** *Progress in Artificial Intelligence: 12th Portugese Conference on Artificial Intelligence* Lecture Notes in Computer Science, Springer 2005, **3808**:272-283.
27. Blockeel H, Schietgat L, Struyf J, Džeroski S, Clare A: **Decision trees for hierarchical multilabel classification: A case study in functional genomics.** *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases* Lecture Notes in Artificial Intelligence 2006, **4213**:18-29.
28. Mewes H, Heumann K, Kaps A, Mayer K, Pfeiffer F, Stocker S, Frishman D: **MIPS: A database for protein sequences and complete genomes.** *Nucleic Acids Research* 1999, **27**:44-48.
29. Breiman L, Friedman J, Olshen R, Stone C: *Classification and Regression Trees* Belmont: Wadsworth 1984.
30. Breiman L: **Random forests.** *Machine Learning* 2001, **45**:5-32.
31. Drucker H: **Improving regressors using boosting techniques.** *Proceedings of the 14th International Conference on Machine Learning* 1997, 107-115.
32. Breiman L: **Bagging predictors.** *Machine Learning* 1996, **24(2)**:123-140.
33. Caruana R: **Multitask Learning.** *Machine Learning* 1997, **28**:41-75.
34. Kocev D, Vens C, Struyf J, Džeroski S: **Ensembles of multi-objective decision trees.** *Proceedings of the 18th European Conference on Machine Learning* 2007, 624-631.
35. Ashburner M, Ball C, Blake J, Botstein D, Butler H, Cherry J, Davis A, Dolinski K, Dwight S, Eppig J, Harris M, Hill D, Issel-Tarver L, Kasarskis A, Lewis S, Matese J, Richardson J, Ringwald M, Rubin G, Sherlock G: **Gene Ontology: Tool for the unification of biology. The Gene Ontology Consortium.** *Nature Genetics* 2000, **25**:25-29.
36. Ouali M, King R: **Cascaded multiple classifiers for secondary structure prediction.** *Protein Science* 2000, **9(6)**:1162-76.
37. Altschul S, Madden T, Schaffer A, Zhang J, Zhang Z, Miller W, Lipman D: **Gapped BLAST and PSI-BLAST: A new generation of protein database search programs.** *Nucl Acids Res* 1997, **25**:3389-3402.
38. Gough J, Karplus K, Hughey R, Chothia C: **Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure.** *Molecular Biology* 2001, **313(4)**:903-919.
39. Zdobnov E, Apweiler R: **InterProScan - an integration platform for the signature-recognition methods in InterPro.** *Bioinformatics* 2001, **17(9)**:847-848.
40. Provost F, Fawcett T: **Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions.** *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* AAAI Press 1998, 43-48.
41. Davis J, Goadrich M: **The relationship between precision-recall and ROC curves.** *Proceedings of the 23rd International Conference on Machine Learning* 2006, 233-240.
42. Breiman L: **Out-of-bag estimation.** Technical Report, Statistics Department, University of California 1996http://ftp.stat.berkeley.edu/pub/users/breiman/OOBestimation.ps.Z.
43. Wilcoxon F: **Individual comparisons by ranking methods.** *Biometrics* 1945, **1**:80-83.
44. Joachims T: **Making large-scale SVM learning practical.** *Advances in Kernel Methods - Support Vector Learning* MIT-Press, Cambridge, MA, USAScholkopf B, Burges C, Smola A 1999.

doi:10.1186/1471-2105-11-2
Cite this article as: Schietgat *et al.*: Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics* 2010 **11**:2.

6.4 Summary of the case studies

We applied the developed ensembles of PCTs to three problems of practical relevance in three application domains. In the three 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 remotely 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 state of Victoria, 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 of hierarchical annotation of medical (X-Ray) images and an additional task of general photo annotation. The ensembles of PCTs outperformed, on all three tasks, a collection of SVMs with χ^2 kernels (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 so far. 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 predicting 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 (hierarchy shaped as a directed acyclic graph). 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 Further developments

In the previous chapters, we presented an extension for predicting structured outputs of the most widely used ensemble techniques in the 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, 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 cases, closed-form prototypes do not exist, which means medoids have to be used as prototypes and the combination of the predictions of individual PCTs in an ensemble needs 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 (Bauer and Kohavi, 1999; Craven, 1996; Domingos, 1998; Ferri *et al.*, 2002; Geurts, 2001; Kargupta *et al.*, 2006; Triviño-Rodriguez *et al.*, 2008; Van Assche, 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.

7.1 Predicting other types of structured outputs

The approaches that we described in Chapters 3 and 4 and 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 (Sections 3.1 and 7.3) will now use the new distance measure.

The construction of the ensembles will change in the part of the voting scheme (Section 4.1.5). 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 Section 4.1.5 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.

7.1.1 Distances for hierarchical classification

In Section 3.2.2, 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, Kocev and Džeroski, 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 directed acyclic graphs 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).

7.1.2 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, Squire, Kocev, Hawes, Young and Džeroski (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.

7.1.3 Prototypes, voting and variance

The ensembles of PCTs we considered in Chapter 4, 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. The medoid is calculated as $argmin_m \sum_{X \in E} d^2(X, m)$, where m is the medoid of the set of examples E and d is the distance defined for the given type of structured output.

We follow the same line of reasoning for the voting scheme used by the ensembles. The predictions of the base predictive models form a set of examples over which we calculate the medoid using the above equation. This means that the prediction of the whole ensemble will be one of the predictions of the base predictive models. The selected prediction is the prediction with the smallest average squared distance to all other predictions.

7.2 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 7.1). 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.

7.2.1 Feature ranking using random forests

The approach for feature ranking using random forests of PCTs, proposed by Kocev *et al.* (2008), is presented in Table 7.1. 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 7.1). The samples that are not selected for the bootstrap replicate are called out-of-bag 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 out-of-bag samples are randomly permuted attribute-by-attribute thus obtaining D noised/permuted out-of-bag samples (line 3 from $Update\_Imp$ procedure). The predictive performance of each tree T_i is evaluated on the original out-of-bag data ($Err_{OOB_i} = Evaluate(T_i, E_{OOB_i})$) and the permuted versions of the out-of-bag data ($Err_{ji} = Evaluate(T_i, Randomize(E_{OOB_i}, j))$). The 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 7.1). The importance is averaged over all trees in the forest. The variable importance is calculated using the following equation:

$$I_j = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{Err_{ji} - Err_{OOB_i}}{Err(OOB_i)} \tag{7.1}$$

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 ($0 < j \leq D$).

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)).

Table 7.1: 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**

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).

7.2.2 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 7.1 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

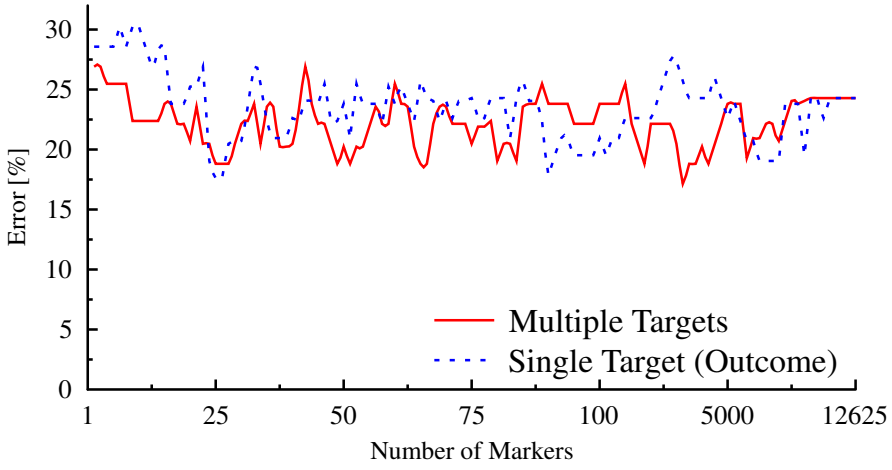


Figure 7.1: Two testing error curves for feature ranking: one for all clinical parameters simultaneously and one for disease outcome. The error rate is measured on the target variable 'disease outcome'.

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 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.

7.3 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 TDIDT 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).

[1]Note that this is to be expected having in mind the algorithm presented below in Table 7.2 and the heuristic score from Equation 7.2. If a given tree has good predictive performance, then its refinements will most probably also have good predictive performance.

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 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.

7.3.1 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 7.2. 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. 7.2). 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.

Table 7.2: 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: $\quad\quad\quad h_{\text{worst}} = \max_{\text{T} \in \text{beam}_\text{i}} \text{Heuristic(T, E)}$
13: $\quad\quad\quad T_{\text{worst}} = \text{argmax}_{\text{T} \in \text{beam}_\text{i}} \text{Heuristic(T, E)}$
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

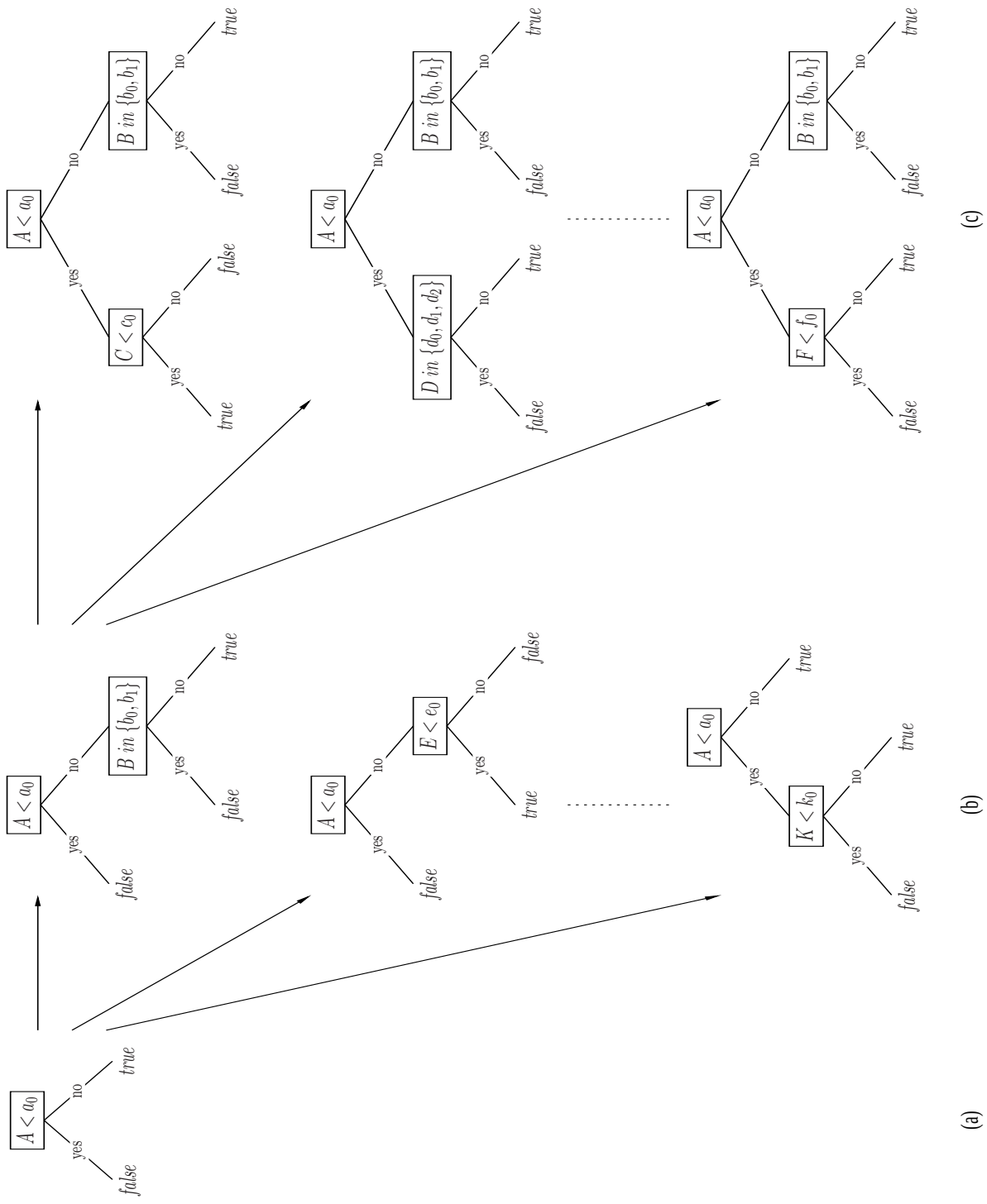


Figure 7.2: Refining the trees in the beam. (a) A tree in the beam; (b) The refinements of tree (a); (c) The refinements of the top-most tree in (b). Note that the refinements (c) are only computed in a subsequent iteration of the search after the top-most tree of (b) has entered the beam.

For each generated refinement, CLUS-BS computes its heuristic value. The heuristic function differs from the heuristic used in the TDIDT 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 TDIDT only needs to compare trees that differ in one node, i.e., to rank different tests for the same tree node.

The heuristic that we propose to use is:

$$h(T, E) = \left(\sum_{\text{leaf} \in T} \frac{|E_{\text{leaf}}|}{|E|} Var(I_{\text{leaf}}) \right) + \alpha \cdot \text{size}(T) \,, \tag{7.2}$$

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 and α is a regularization parameter (Hastie *et al.*, 2003). 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 7.2 is identical to the heuristic used in the TDIDT 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 TDIDT. The only difference with TDIDT is the order in which the leaves are refined: TDIDT 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 7.2) to include also a similarity term. We discuss this similarity term in the next section.

7.3.2 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:

$$d(T_1, T_2, E) = \frac{1}{\eta} \cdot \sqrt{\frac{\sum_{t \in E} d_p(p(T_1, t), p(T_2, t))^2}{|E|}}\,, \tag{7.3}$$

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 7.3, 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. Alternatively, one can also consider the Hellinger distance (Beran, 1977) as a distance measure between the predictions.

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

$$\delta(a, b) = \begin{cases} 1 \text{ if } a \neq b \\ 0 \text{ if } a = b \end{cases} \tag{7.4}$$

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 7.2) can be calculated as follows:

$$h_s(T, \text{beam}, E) = \left(\sum_{\text{leaf} \in T} \frac{|E_{\text{leaf}}|}{|E|} Var(E_{\text{leaf}}) \right) + \alpha \cdot \text{size}(T) + \beta \cdot \text{sim}(T, \text{beam}, E) \quad (7.5)$$

where the first two terms are the same as in the Equation 7.2, 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:

$$\text{sim}(T, \text{beam}, E) = 1 - \frac{d(T, T_{cand}, E) + \sum_{T_i \in \text{beam}} d(T, T_i, E)}{|\text{beam}|} \quad (7.6)$$

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 7.3.

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 7.5. 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 $d(T_a, T_a, E) = 0$.

The similarity constraint can be used in a knowledge discovery scenario where the task is to learn trees that are more similar or different than a given tree (called syntactic constraint). For this purpose, the heuristic score was adapted as follows. The adapted heuristic score will consider only the distance to the tree provided by the syntactic constraint and not the distance to the other trees in the beam (it will use slightly modified version of the equation for the similarity score). The user can require, by changing the β parameter from the heuristic score, from the beam search algorithm, that the resulting trees are more similar or more different than the provided tree. This enables the beam

search induction to include some background knowledge, given in the form of partial tree, into the tree induction process.

7.3.3 Computational complexity

We present the computational costs of both CLUS-BS and CLUS-BS-S. The computational cost of CLUS-BS is as follows. Computing the best test for one attribute for the instances in a given leaf costs $\mathcal{O}(|E_{\text{leaf}}| \log |E_{\text{leaf}}|)$ (to find the best split point for a numeric attribute, the instances must be sorted; after sorting, finding the best split can be done in $\mathcal{O}(|E_{\text{leaf}}|)$ time Quinlan (1993)). If the score of the best test is better than that of the worst tree in the beam, then the refined tree must be constructed $\mathcal{O}(\text{size}(T))$ and inserted into the beam $\mathcal{O}(\log k)$ (if the beam is implemented as a balanced binary search tree).

Repeating this for all attributes (A) and all leaves yields $\mathcal{O}(|A| \cdot |E| \log |E| + |E| \cdot |\text{leaves}(T)| \cdot (\text{size}(T) + \log k))$ because each instance occurs in at most one leaf. If s upper-bounds the size of the trees in the beam, then the cost of refining the entire beam is $\mathcal{O}(k \cdot |A| \cdot |E| \log |E| + s^2 k \cdot |A| + s \cdot |A| \cdot k \log k)$. Finally, the cost of running n iterations of CLUS-BS is $\mathcal{O}(nk \cdot |A| \cdot |E| \log |E| + ns^2 k \cdot |A| + ns \cdot |A| \cdot k \log k)$. For comparison, the computational cost of TDI is $\mathcal{O}(D \cdot |A| \cdot |E| \log |E|)$, with D the depth of the tree. Assuming that the first term dominates the complexity of CLUS-BS, it follows that CLUS-BS is $\mathcal{O}(kn/D)$ times slower. Note that n is in the best case equal to the number of leaves in the largest tree because each iteration can add at most one leaf.

The CLUS-BS-S has bigger computational complexity than CLUS-BS. First, in CLUS-BS-S, evaluating a single model takes a factor $\mathcal{O}(k^2 \cdot |E|)$ longer than in CLUS-BS, with k the beam width and $|E|$ the number of instances. We exploit properties of the distance measure ($d(T_a, T_b, I) = d(T_b, T_a, I)$ and $d(T_a, T_a, I) = 0$) to make the evaluation of the similarity component efficient. Second, CLUS-BS-S evaluates more PCTs than CLUS-BS because of the similarity measure that is included in the heuristic score. In CLUS-BS, the score of the *worst* tree in the beam monotonically improves with the iteration number. In CLUS-BS-S, this is no longer the case because the score of the trees in the beam needs to be recomputed when a new tree enters the beam (because of the similarity component). As a result, it becomes harder for CLUS-BS-S to satisfy the stopping criterion (the beam no longer changes).

7.3.4 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 (practically turning off the soft size constraint) and the influence of the diversity β to 1. This value for β was chosen because the values for the similarity and the average variance are in the same range and this gives equal importance to the accuracy and the diversity of the beam. 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 TDIDT algorithm. CLUS-BS wins[1] on 5 classification and 3 regression tasks[2]. TDIDT 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 TDIDT 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 when size constraint is set to 31 and 51 and without size constraint), because the relative 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

[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.

[2]In this context, we consider a task the combination of a dataset with a size constraint.

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.

8 Conclusions and further work

In this thesis, we propose methods for learning ensembles of models for predicting structured outputs. Each of the models in our ensembles makes a prediction for the whole output data 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 (shaped as a tree and a directed acyclic graph). Additionally, the methods are scalable to a wide range of datasets with different number of examples and descriptive variables, and different types and sizes of structured outputs.

We evaluate the proposed methods on a wide range of datasets. Approximately ten benchmark datasets are used for each type of structured output to evaluate the strengths and weaknesses of our approaches and compare them to other methods. Our methods have been used in three case studies that concern problems of practical importance in three different domains: vegetation condition assessment, image annotation and functional genomics.

In the remainder of this chapter, we first summarize the scientific contributions of the thesis. We then discuss how the proposed methods can be further improved and applied.

8.1 Conclusions

The methods we propose in this thesis extend the predictive clustering framework in the context of ensemble learning. They also contribute to the areas of ensemble learning, predicting structured outputs, as well as the respective application domains of the case studies. We summarize the contributions as follows:

- We developed ensemble learning methods for predicting structured outputs that are based on PCTs. The proposed methods are general in terms of the type of the structured output. Currently, they can handle three types of structured outputs: multiple continuous targets, multiple discrete targets and classes organized in the form of a hierarchy (tree-shaped or directed acyclic graph). However, they can be easily adapted for other types of structured outputs. With this, we extend the predictive clustering framework in the context of ensemble learning.

- We have performed extensive empirical evaluation of the proposed methods over a variety of domains.
  - The experimental results show that both ensembles of global and local predictive models statistically significantly improve the predictive performance of their single model counterpart.
  - We have constructed ensembles of up to 1000 predictive models and selected ensembles of 50 global predictive models as optimal in terms of the trade-off between predictive performance and efficiency (running time and model complexity).
  - Ensembles of global and local predictive models perform equally well. However, ensembles of global classifiers need fewer trees in the ensemble to reach the saturation point. Furthermore, the ensembles of global predictive models are more efficient in terms of training time and size of the trees.
- We applied the developed ensembles of PCTs to three application domains. In the case studies, the ensembles of PCTs were compared to state-of-the-art approaches used in the respective domains. We summarize the conclusions from the case studies as follows:
  - Predicting vegetation condition: The obtained PCT models contributed to a 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, ensembles of PCTs were used to generate maps of the condition of the indigenous vegetation across Victoria state, Australia. These maps can be used to support biodiversity planning, management and investment decisions.
  - Hierarchical classification of medical images: The ensembles of PCTs for HMC outperformed 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 so far 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.
  - Predicting gene function: We applied bagging of PCTs to predict gene function in three model organisms: *Saccharomyces cerevisiae*, *Arabidopsis thaliana* and *Mus musculus*. The extensive experimental evaluation showed that bagging of PCTs is competitive to state-of-the-art approaches in the area of functional

genomics for all three organisms. It was only outperformed by approaches that use gene interaction networks information, which was not used in our approach.

- We have proposed a method, based on random forests, that performs feature ranking for structured outputs. The method produces single feature ranking valid for the complete structured output and takes into account the dependencies and the relations that exist between the components of the structured output. The ranking produced in 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 ranking for each individual target.

- We have suggested a novel ensemble learning method that is based on the beam search strategy and uses decision trees as base classifiers. This method enables direct control over the diversity in the ensemble and allows us to investigate the optimal trade-off between the ensemble's diversity and its predictive performance. Furthermore, the ensemble produced with beam search can be interpreted by selecting the top-ranked tree from the ensemble as a representative for the whole ensemble.

8.2 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 7.1 and tuples of mixed primitive data types, both continuous and discrete). 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 7.1) 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 has the form of multiple continuous variables, 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.

A fourth line of further work is to conduct a theoretical analysis of the predictive power of the ensembles for predicting structured outputs. This analysis should be performed by using artificial datasets, bias-variance decomposition of the error and a model of the diversity of the ensemble. The first approach would answer how the *relatedness* between the components of the structured outputs affects the predictive performance of a single global predictive model. The results of such a study could be used to further investigate this effect in the context of ensemble learning. The second approach of bias-variance decomposition of the error is a known and established method for the analysis of ensemble methods. The work done in this area should be transferred and translated in the context of predicting structured outputs. The last approach models the diversity in the ensemble and searches for a trade-off between the diversity of the models in an ensemble and the ensembles' predictive performance. By the introduction of the beam search algorithm we have made initial steps in this direction, but much remains to be done.

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 algorithms (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.

9 Acknowledgments

I wish to thank my supervisor prof. dr. Sašo Džeroski for giving me the opportunity to join his research group and introducing me in the area of machine learning. His ideas and suggestions were the basis for the work presented in this thesis.

I would also like to thank the members of my PhD committee for carefully reading the text of the thesis and providing useful feedback. These comments improved the quality of the thesis and taught me how to better present my work. They also helped me to place my work in a broader context.

Several funding bodies financially helped the research presented here. I wish to thank all of them for their support through the years. In chronological order, these bodies are: the Ad Futura Scientific and Educational Foundation, the Department of Knowledge Technologies (headed by prof. dr. Nada Lavrač) and the Department of Environmental Sciences (headed by prof. dr. Milena Horvat) at the Jožef Stefan Institute, the European Commission (through the research projects IQ - Inductive Queries for Mining Patterns and Models, and PHAGOSYS - Systems biology of phagosome formation and maturation - modulation by intracellular pathogens) and the Slovenian Research Agency (through the research project Data Mining for Integrative Data Analysis in Systems Biology).

Many thanks to the DTAI group from Katholieke Universiteit Leuven, especially Celine Vens, Jan Struyf, Hendrik Blockeel and Leander Schietgat. Thank you for the numerous discussions, comments and insightful ideas. Jan and Celine, thank you for your patience and help regarding the questions I had about Clus and the collaboration on writing the manuscripts that are the basis for the work presented in this thesis.

The co-authors of my papers contributed many hours of hard work while envisioning and writing the papers. Specially, I would like to thank my co-authors of the papers included in the thesis: Celine Vens, Graeme Newell, Hendrik Blockeel, Ivica Dimitrovski, Jan Struyf, Leander Schietgat, Matt White, Peter Griffionen, Sašo Džeroski and Suzana Loskovska. It was a great experience to work with you and I learned a lot in the process.

The colleagues at the Department of Knowledge Technologies are responsible for the pleasant work environment. I will not name each one of you, but simply say: Thank you all.

I wish to thank Ivica Dimitrovski for the great team spirit while preparing a manuscript or entering an image annotation competition. The research in this thesis connected to image annotation would have not been possible without his help. Besides work, during his visits in Ljubljana we had many interesting 'beer inspired' discussions.

Special thanks goes to Ivica and Panče, who are not just my colleagues, but also my friends. We moved together to Ljubljana and we went together through the ups and downs in the lives of graduate students. Ivica was my office-mate during the whole time. Thank you for creating relaxed and enjoyable atmosphere in the office. I will not forget the great time that we had during the conferences we attended. Panče was my flat-mate during the last three years. Thank you for all the cooking advices - they most certainly made me a better cook. I am very grateful for carefully proof-reading the text of the thesis and providing insightful comments.

I would like to express my deepest gratitude to my parents, Ljupčo and Spomenka, and my sister Ratka, for their love and support. Without the encouragement from you, none of this would have been possible.Thanks you for making my each visit home special and unforgettable. I would also like to express my gratitude to my aunt Blagica who was my first mentor and guided me through the first steps towards scientific work and writing. Without her support I would not have never enrolled in graduate school. Last, but not least, I wish to thank my dearest Aleksandra for all her love and support, especially during the writing of this thesis. Nothing else is able to cheer me up more than a smile on your face.

10 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.

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.

Beran, R. (1977). Minimum Hellinger distance estimates for parametric models. *Annals of statistics*, **5**, 445–463.

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.

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., 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., 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.

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.

Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2011). Hierchical annotation of medical images. *Pattern Recognition*.

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., 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, J. and Popescu, B. (2003). Importance sampled learning ensembles. Technical report, Dept. of Statistics, Stanford University.

Friedman, J. and Popescu, B. (2008). Predictive learning via rule ensembles. *Annals of Applied Statistics*, **2**(3), 916–954.

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.

Gärtner, T. and Vembu, S. (2009). On structured output training: hard cases and an efficient alternative. *Machine Learning*, **76**, 227–242.

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.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2003). *The Elements of Statistical Learning*. Springer.

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.

Kim, S.-J. and Lee, K. B. (1998). Constructing decision trees with multiple response variables. *International Journal of Management and Decision Making*, **4**(4), 337–353.

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.

Larsen, D. and Speckman, P. (2004). Multivariate regression trees for analysis of abundance data. *Biometrics*, **60**(2), 543–549.

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.

Mehta, M., Agrawal, R., and Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. In *Extending Database Technology*, pages 18–32.

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.

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.

Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., and Džeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, **11**(2), 1–14.

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.

Segal, M. R. (1992). Tree-Structured methods for longitudinal data. *Journal of the American Statistical Association*, **87**(418), 407–418.

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

Shafer, J. C., Agrawal, R., and Mehta, M. (1996). Sprint: A scalable parallel classifier for data mining. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proc. 22nd Int. Conf. Very Large Databases, VLDB*, pages 544–555. Morgan Kaufmann.

Silla, C. and Freitas, A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, **22**(1-2), 31–72.

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.

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

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.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

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

Xiao, Y. and Segal, M. R. (2009). Identification of yeast transcriptional regulation networks using multivariate random forests. *PLoS computational biology*, **5**(6), e1000414+.

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

Zhang, H. (1998). Classification trees for multiple binary responses. *Journal of the American Statistical Association*, **93**(441), 180–193.

Appendix A: Complete results of the experimental evaluation

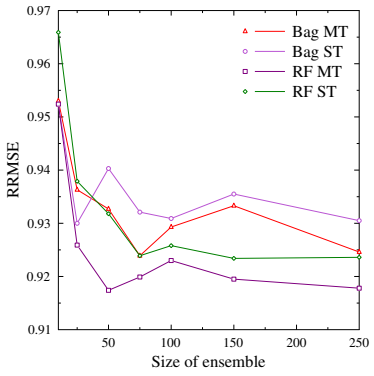
The results of applying the different methods for predicting structured outputs to the benchmark datasets were outlined in Chapter 5. Here, we give the complete results, organized by the three considered tasks: predicting of multiple continuous targets, predicting of multiple discrete targets and hierarchical multi-label classification. For each task, we give the saturation curves, statistical tests comparing the predictive performance of the different approaches and the efficiency of the approaches over all ensemble sizes.

The statistical test results are from the Friedman test for multiple hypothesis testing and the 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.

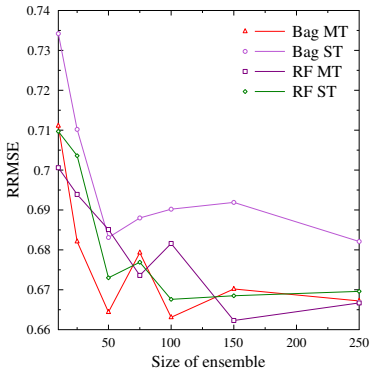
A.1 Prediction of multiple continuous targets

A.1.1 Saturation curves

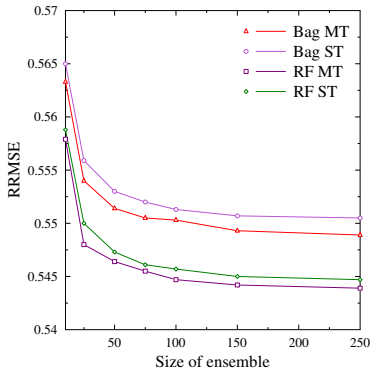
Collembola



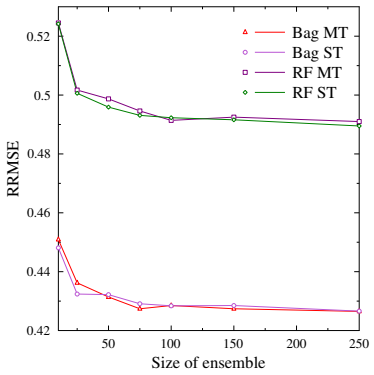
EDM



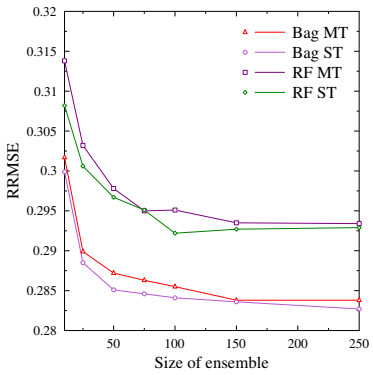
Forestry-Kras



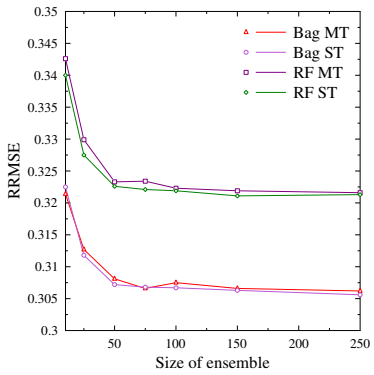
Forestry-Slivnica-LandSat



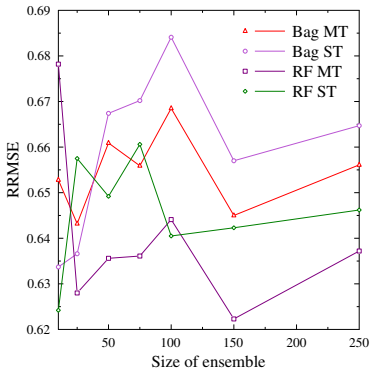
Forestry-Slivnica-IRS



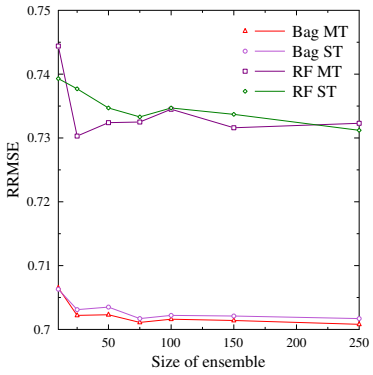
Forestry-Slivnica-SPOT



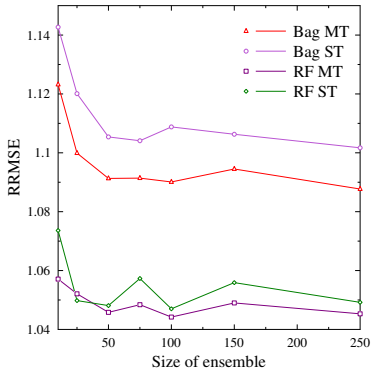
Sigmea real



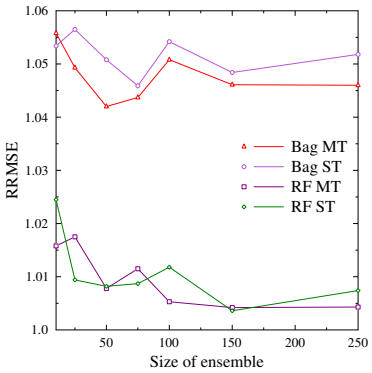
Soil quality



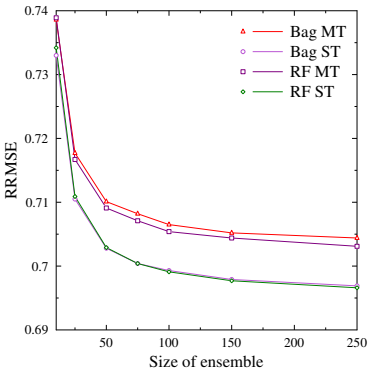
Solar-flare 1



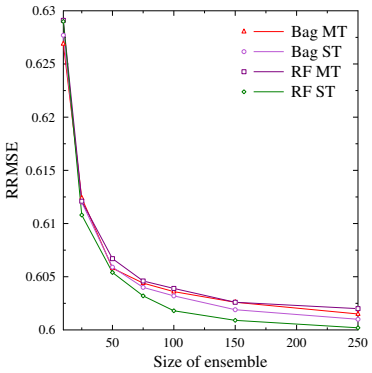
Solar-flare 2



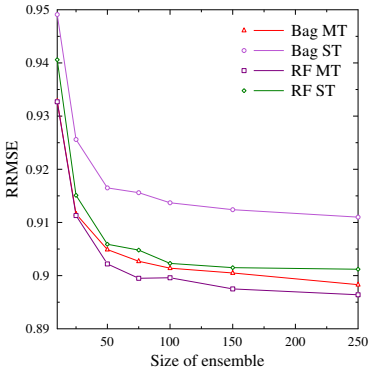
Vegetation clustering



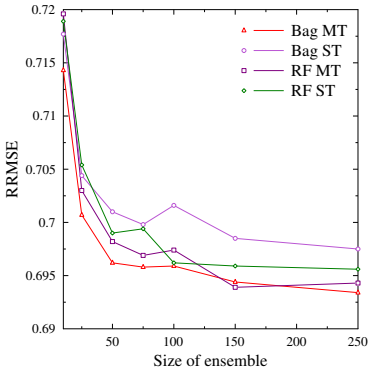
Vegetation condition



Water quality

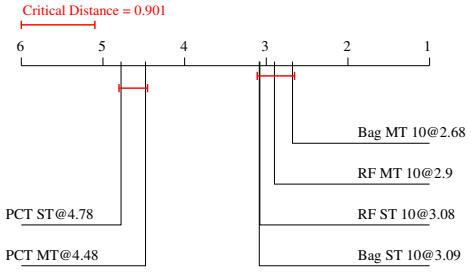


Overall

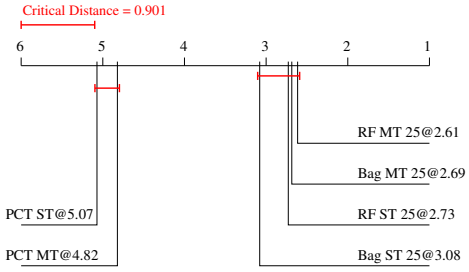


A.1.2 Statistical tests for predictive performance

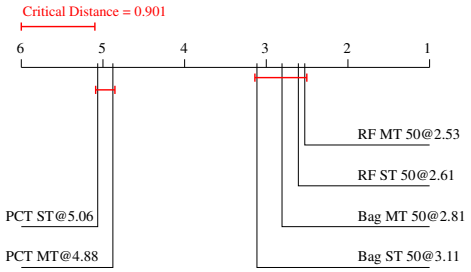
Ensemble size: 10



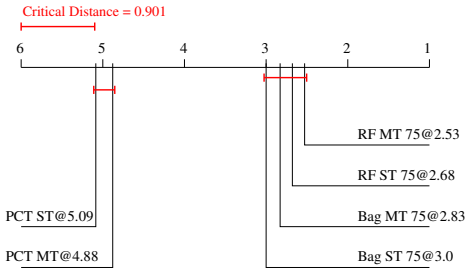
Ensemble size: 25



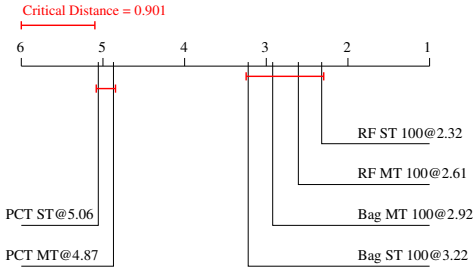
Ensemble size: 50



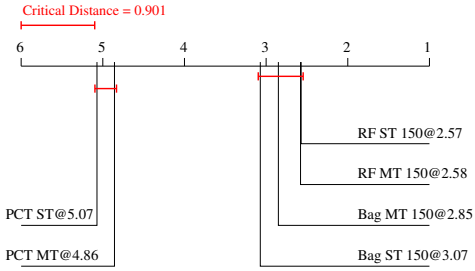
Ensemble size: 75



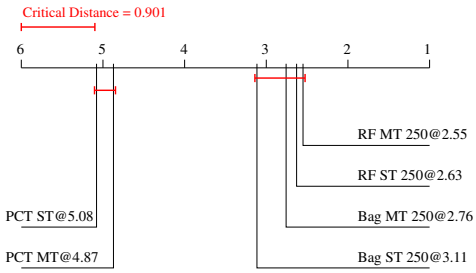
Ensemble size: 100



Ensemble size: 150



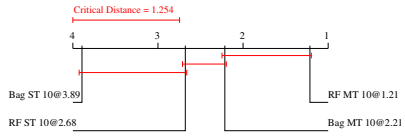
Ensemble size: 250



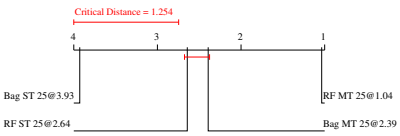
A.1.3 Statistical tests for efficiency

Time consumption

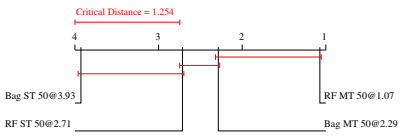
Ensemble size: 10



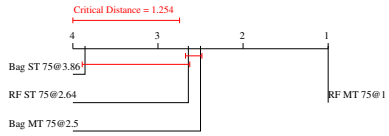
Ensemble size: 25



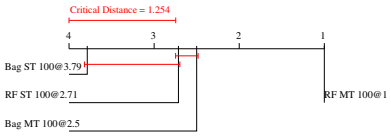
Ensemble size: 50



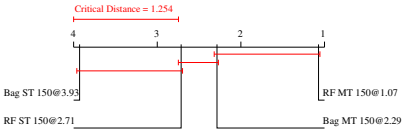
Ensemble size: 75



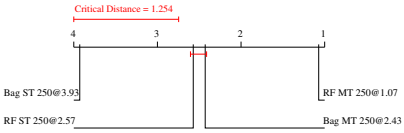
Ensemble size: 100



Ensemble size: 150

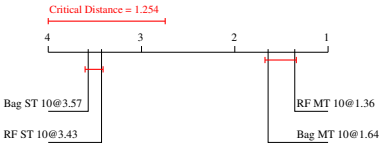


Ensemble size: 250

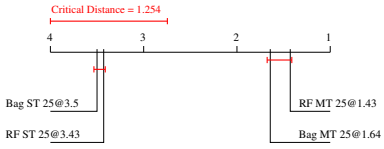


Model size

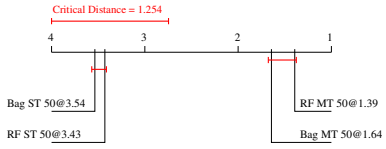
Ensemble size: 10



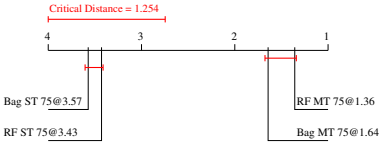
Ensemble size: 25



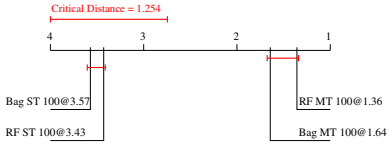
Ensemble size: 50



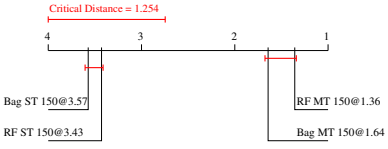
Ensemble size: 75



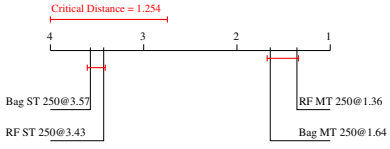
Ensemble size: 100



Ensemble size: 150

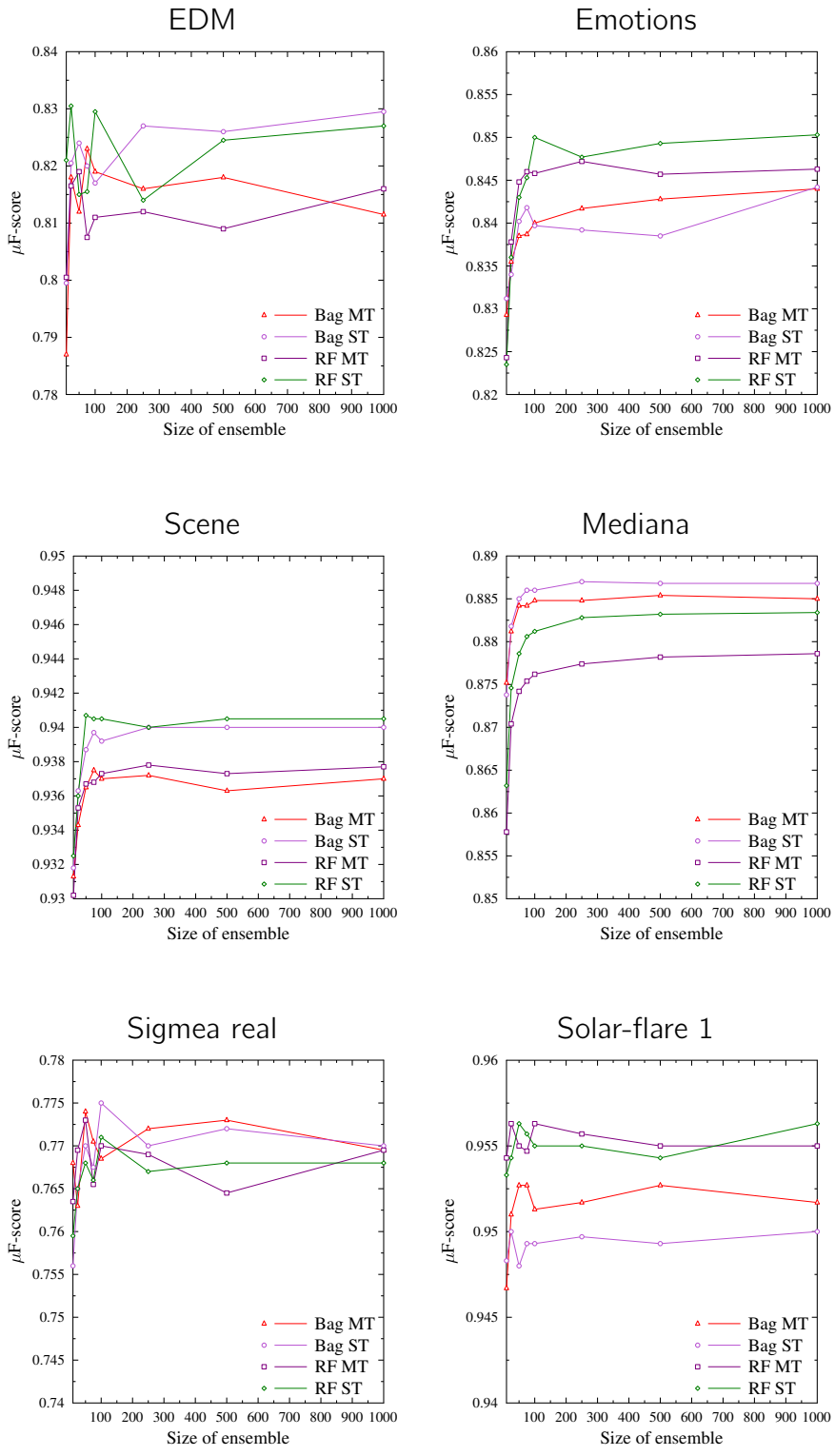


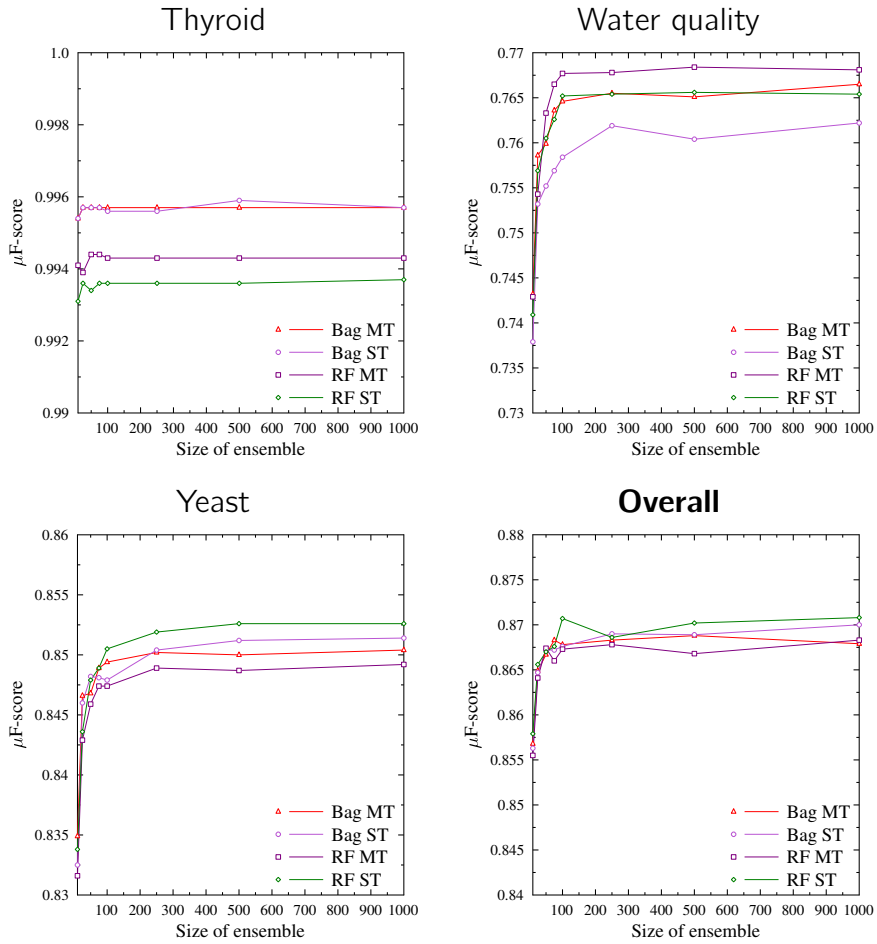
Ensemble size: 250



A.2 Prediction of multiple discrete targets

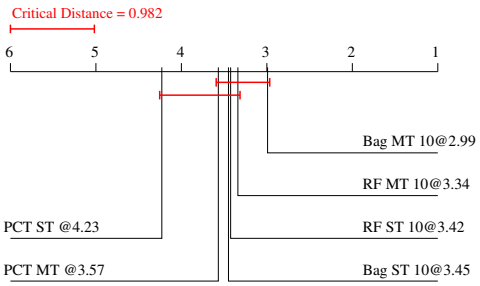
A.2.1 Saturation curves



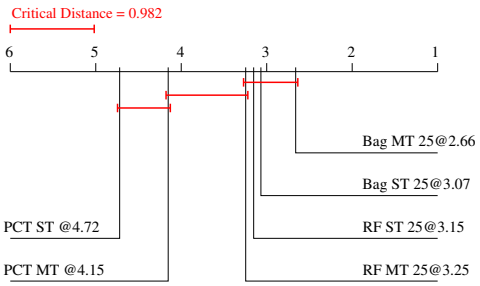

Thyroid
Water quality
Yeast
Overall
μF-score
Size of ensemble
Bag MT
Bag ST
RF MT
RF ST

A.2.2 Statistical tests for predictive performance

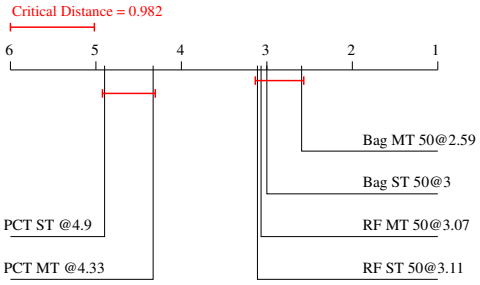
Ensemble size: 10



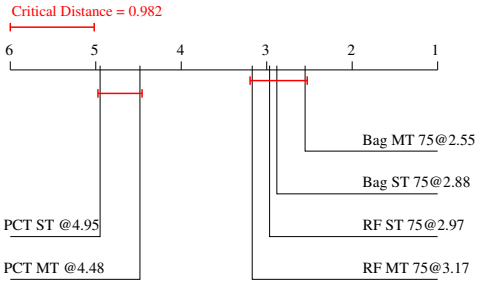
Ensemble size: 25



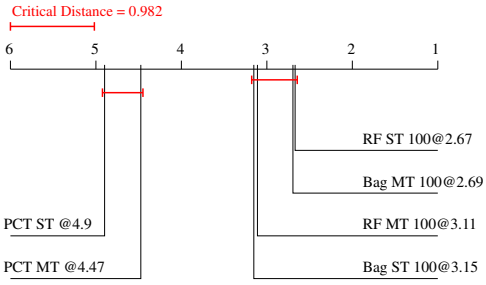
Ensemble size: 50



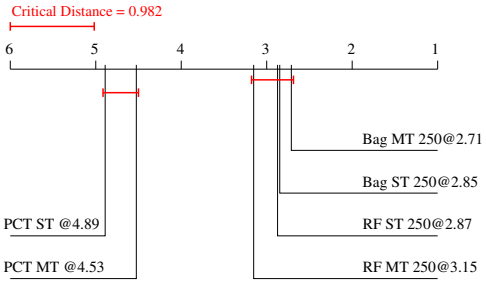
Ensemble size: 75



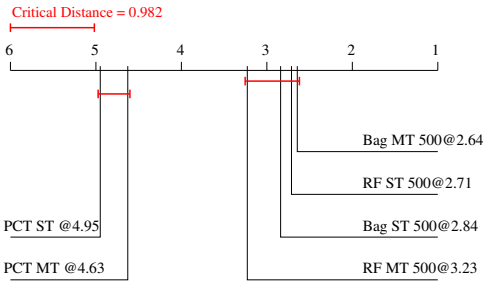
Ensemble size: 100



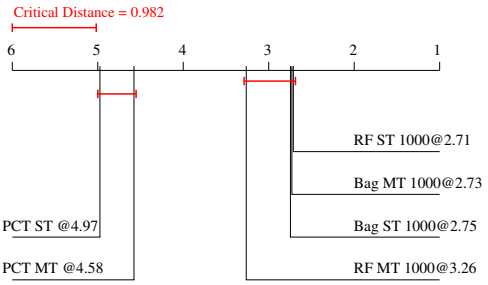
Ensemble size: 250



Ensemble size: 500



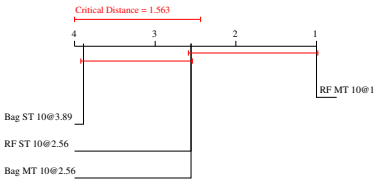
Ensemble size: 1000



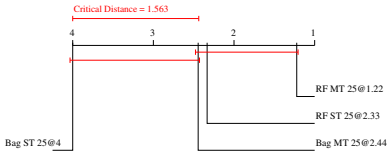
A.2.3 Statistical tests for efficiency

Time consumption

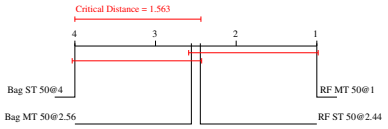
Ensemble size: 10



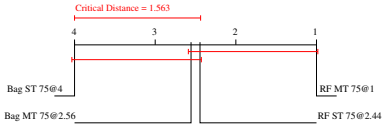
Ensemble size: 25



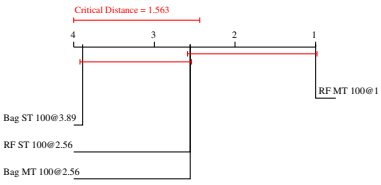
Ensemble size: 50



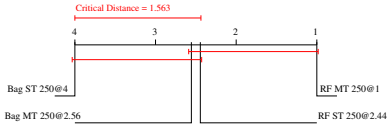
Ensemble size: 75



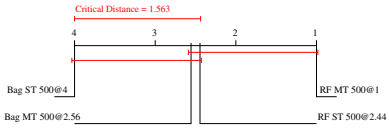
Ensemble size: 100



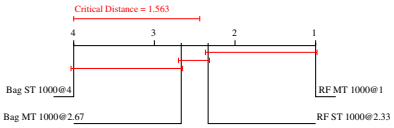
Ensemble size: 250



Ensemble size: 500

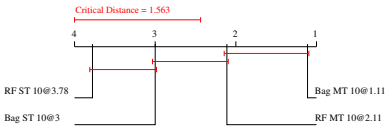


Ensemble size: 1000

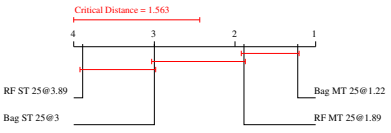


Model size

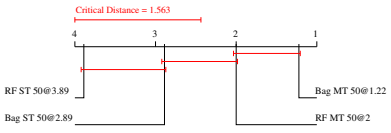
Ensemble size: 10



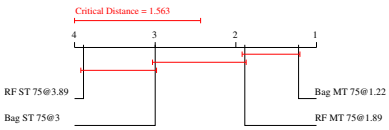
Ensemble size: 25



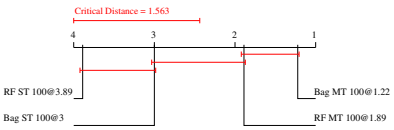
Ensemble size: 50



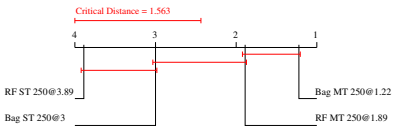
Ensemble size: 75



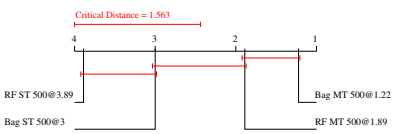
Ensemble size: 100



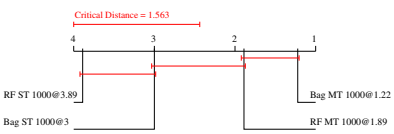
Ensemble size: 250



Ensemble size: 500

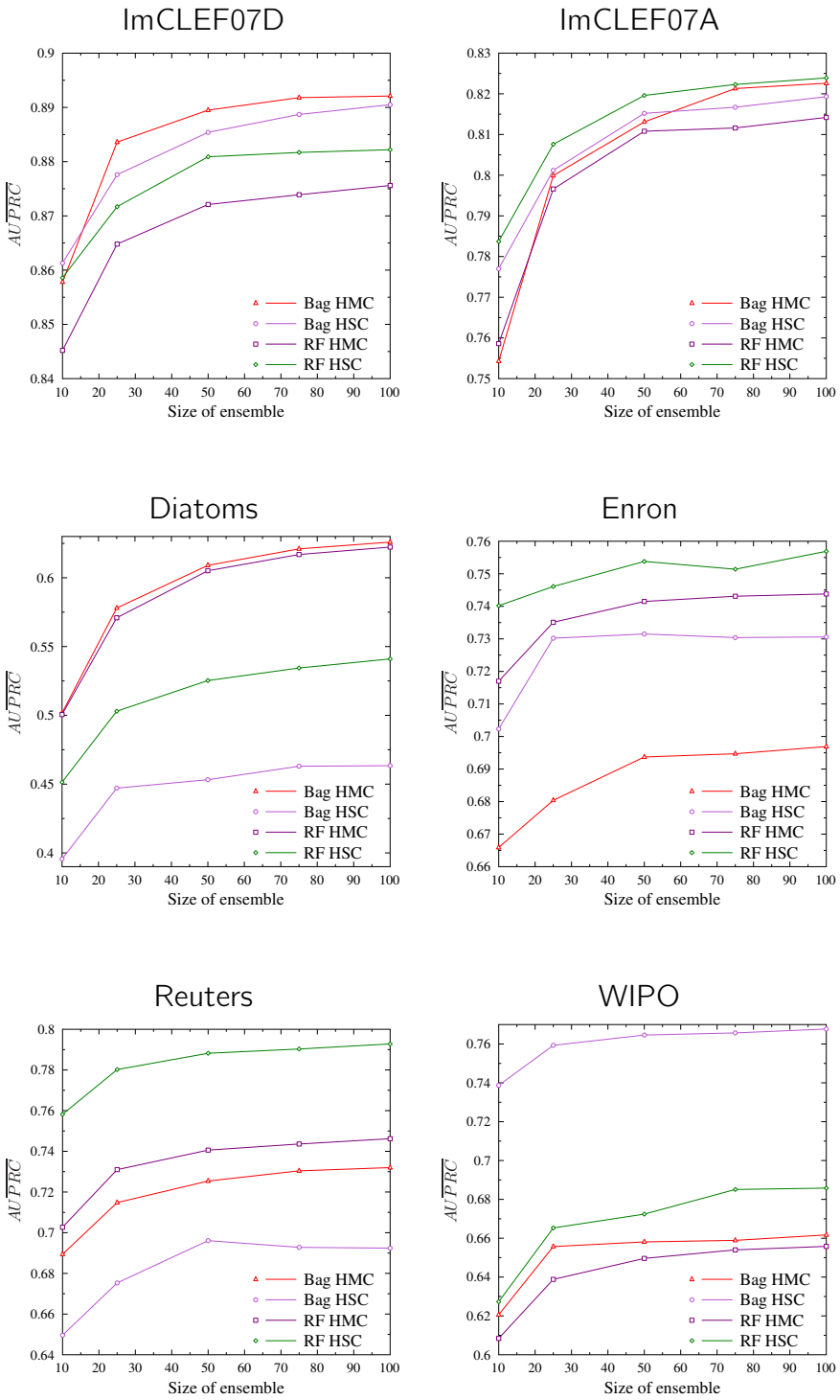


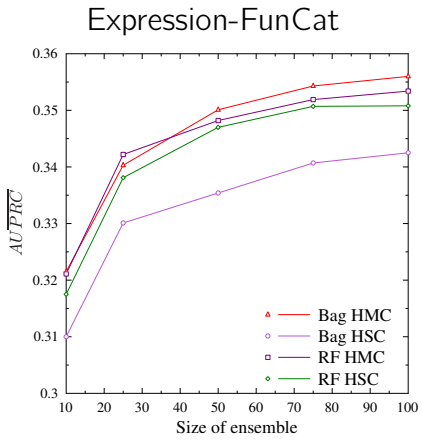
Ensemble size: 1000

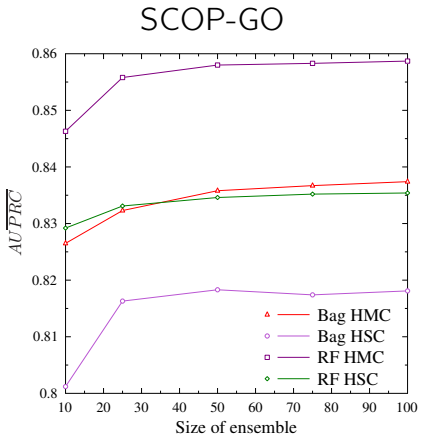


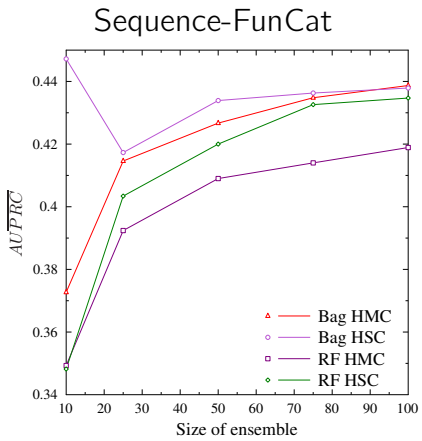
A.3 Hierarchical multi-label classification

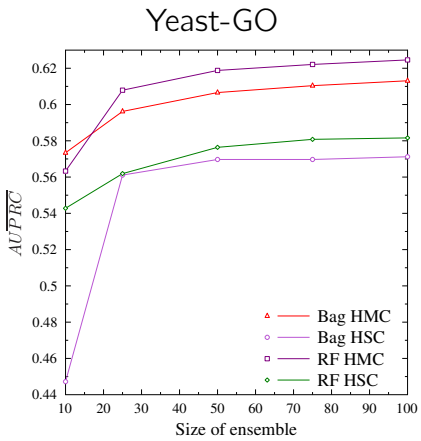
A.3.1 Saturation curves

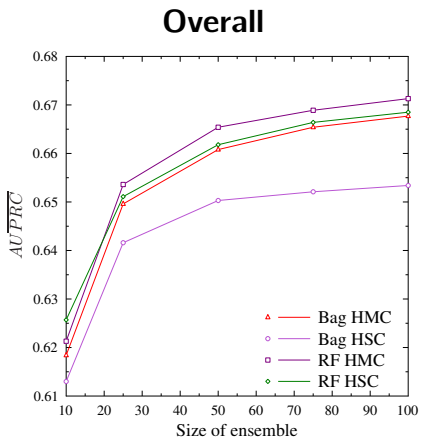



Expression-FunCat
AUPRC
Size of ensemble
Bag HMC
Bag HSC
RF HMC
RF HSC


SCOP-GO
AUPRC
Size of ensemble
Bag HMC
Bag HSC
RF HMC
RF HSC

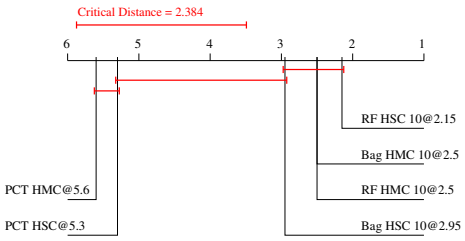

Sequence-FunCat
AUPRC
Size of ensemble
Bag HMC
Bag HSC
RF HMC
RF HSC


Yeast-GO
AUPRC
Size of ensemble
Bag HMC
Bag HSC
RF HMC
RF HSC

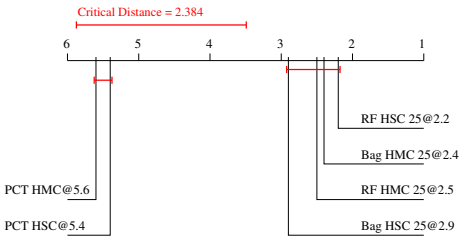

Overall
AUPRC
Size of ensemble
Bag HMC
Bag HSC
RF HMC
RF HSC

A.3.2 Statistical tests for predictive performance

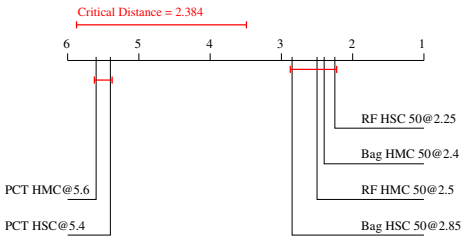
Ensemble size: 10



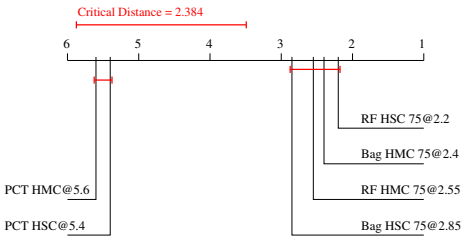
Ensemble size: 25



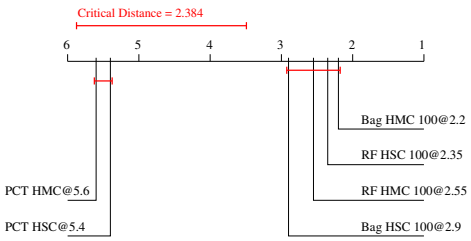
Ensemble size: 50



Ensemble size: 75



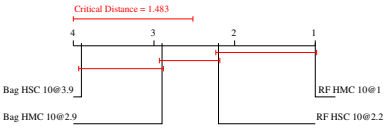
Ensemble size: 100



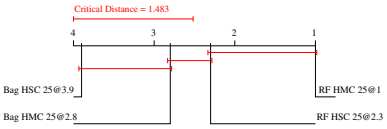
A.3.3 Statistical tests for efficiency

Time consumption

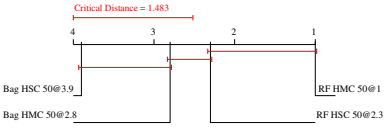
Ensemble size: 10



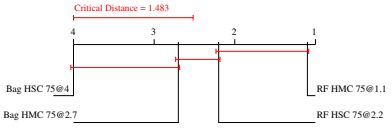
Ensemble size: 25



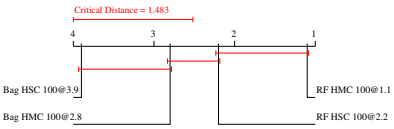
Ensemble size: 50



Ensemble size: 75

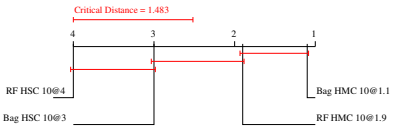


Ensemble size: 100

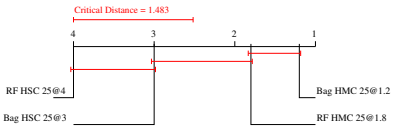


Model size

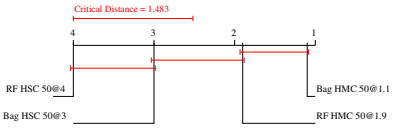
Ensemble size: 10



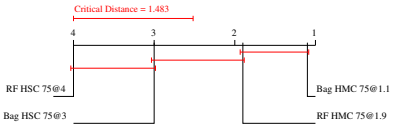
Ensemble size: 25



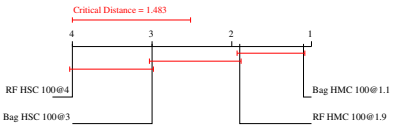
Ensemble size: 50



Ensemble size: 75



Ensemble size: 100



Appendix B: CLUS user manual

The methods presented in this thesis are implemented in the CLUS system. Here, we provide a short user manual for CLUS. We begin by stating some general information about the system and some general settings. Next, we describe the settings that are specific to the ensemble methods. Finally, we give the settings used for beam search induction of predictive clustering trees.

B.1 General information

CLUS is an open source machine learning system that implements the predictive clustering framework. It supports learning of predictive clustering trees and predictive clustering rules. The predictive clustering framework unifies unsupervised clustering and predictive modelling and allows for a natural extension to more complex prediction settings, such as predicting multiple target variables, hierarchical multi-label classification and prediction of time series.

CLUS is co-developed by the *Declarative Languages and Artificial Intelligence* group of the *Katholieke Universiteit Leuven*, Belgium, and the *Department of Knowledge Technologies* at the *Jožef Stefan Institute*, Ljubljana, Slovenia. CLUS is a free software (licensed under the *GPL*) and can be downloaded from `http://dtai.cs.kuleuven.be/clus/`.

CLUS uses (at least) two input files and these are named `filename.s` and `filename.arff`, with `filename` a name chosen by the user. The file `filename.s` contains the parameter settings for CLUS. The file `filename.arff` contains the training data, which need to be in ARRF data format for the tasks of predicting a single or multiple target variables (for HMC and clustering short time series, an extension of the ARFF format is used). The results of a CLUS run go to an output file `filename.out`. Figure 10.1 gives an overview of the input and output files of CLUS.

The algorithms implemented in the CLUS system are controlled by a number of parameters. These parameters are specified in a settings file. For most of the parameters there are default values that are used, unless a different value is specified by the user. The

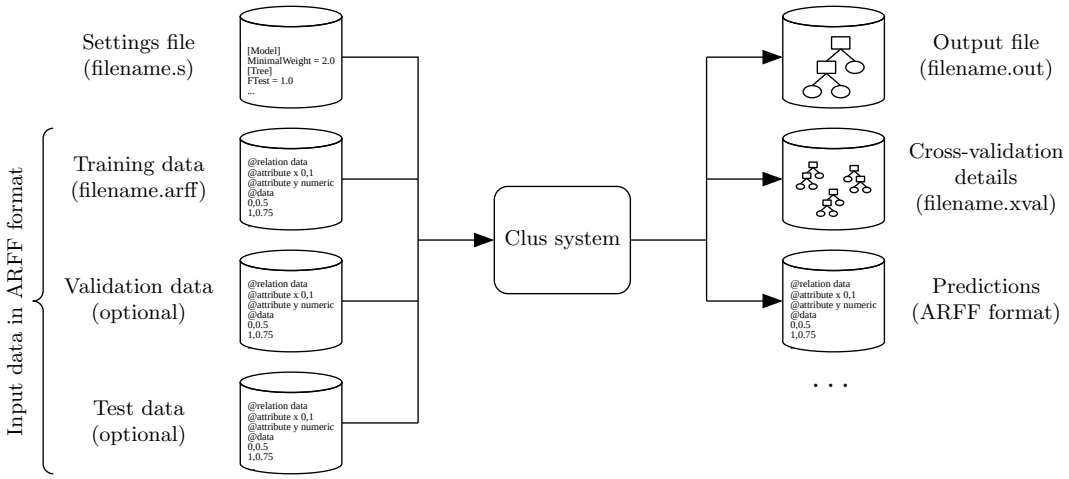


Figure 10.1: Input and output files of CLUS.

parameters in the settings file are grouped in the following sections:

- `General`: specifying the training, testing and validation datasets and number of folds for cross-validation,
- `Attributes`: lists the attributes that should be used as descriptive, target etc.,
- `Tree`: type of heuristic score used for tree learning, tree pruning algorithm, pruning parameters etc.,
- `Rules`: type of heuristic score used for rule learning, coverage, optimization parameters etc.,
- `Output`: which errors and models should be output by CLUS, should the predictions of the model also be provided, etc.,
- `Constraints`: size, depth and error of the tree, syntactic constraints etc.,
- `TimeSeries`: the distance used for clustering time series and the type of sampling when calculating the ICV heuristic,
- `Hierarchical`: parameters for hierarchical multi-label classification, such as type of hierarchy, weighting, threshold for making a prediction, etc.,
- `Beam`: parameters for beam search, such as number of trees in the beam, soft and hard size constraints, similarity constraint, etc.,
- `Ensembles`: parameters for constructing the ensembles, such as number of base predictive models, type of ensemble learning method, voting scheme, etc.

The sections in a settings file are described in detail in the document CLUS*: User's manual* which is available within the CLUS project documentation files. In the remainder of this section, we shortly present the `Attributes` and `Hierarchical` Sections. In the next section, we describe the `Ensembles` and `Beam` sections in more detail.

In the following example, we present a scenario for learning a PCT for predicting multiple targets. First, the attributes from the 25-th position in the dataset until the 28-th position are used as target attributes in the PCT construction. Second, the attributes from the 2-nd until 24-th position are used as descriptive attributes (the splits in the internal nodes of the tree will use these attributes). Next, the construction algorithm will ignore (i.e., disable) the attribute in the first position. Finally, the algorithm will perform normalization of the numeric attributes thus they equally contribute to the overall heuristic score.

```
[Attributes]
Descriptive = 2-24          % index of descriptive attributes
Target = 25-28              % index of target attributes
Disable = 1                 % Disables some attributes (e.g., "5,7-8")
Key = None                  % Sets the index of the key attribute
Weights = Normalize         % Normalize numeric attributes
```

The next example considers construction of a PCT for hierarchical multi-label classification. The classes are organized into a directed acyclic graph (DAG) hierarchy and the weighting factor (w_0) is set to 0.75. The weights for the classes with multiple parents are calculated by using average value over the weights of all parents. The Ftest pruning of the PCT should be done to optimize the pooled *AUPRC* performance measure.

```
[Hierarchical]
Type = DAG                            % Tree or DAG hierarchy?
WType = ExpAvgParentWeight             % aggregation of class weights
WParam = 0.75                          % parameter w_0
OptimizeErrorMeasure = PooledAUPRC  % FTest optimization strategy
ClassificationThreshold = 40       % threshold for "positive"
```

B.2 Settings for learning an ensemble

The classes that implement the ensembles in the CLUS project are located in the `clus.ext` package. Here, we will not discuss technical details concerning the implementation, instead

we present the `Ensembles` section from the settings file and small example for ensemble construction.

The `Ensembles` section from the settings file contains the following options:

- `Iterations =` n : n is the number of base predictive models (trees) in the ensemble, by default it is set to 10.
- `EnsembleMethod =` o : o is an element of {`Bagging, RForest, RSubspaces, BagSubspaces`} and defines the ensemble method, by default it is set to `Bagging`.
  - `Bagging`: Bagging (Breiman, 1996a).
  - `RForest`: Random forest (Breiman, 2001a).
  - `RSubspaces`: Random Subspaces (Ho, 1998).
  - `BagSubspaces`: Bagging of subspaces (Panov and Džeroski, 2007).
- `VotingType =` o : o is an element of {`Majority, ProbabilityDistribution`} and selects the voting scheme for combining predictions of base predictive models when the task is classification or predicting multiple discrete variables. This option has no effect on the voting type for regression and HMC tasks: for those tasks, averaging is used as the voting scheme. The default value is `Majority`.
  - `Majority`: each base predictive model returns one vote.
  - `ProbabilityDistribution`: each base predictive model casts probability distributions for each target attribute; the votes cast are proportional to the predicted class probabilities.
- `SelectRandomSubspaces =` n : n defines size of feature subset for random forests, random subspaces and bagging of subspaces. The default setting is 0, meaning that the feature subset size will be calculated as $\lfloor \log_2 DescriptiveAttributes \rfloor + 1$ - a recommendation by Breiman (2001a). One can also use $f(D) = 1$, $f(D) = \lfloor \sqrt{D} + 1 \rfloor$, $f(D) = \lfloor 0.1 \cdot D + 1 \rfloor$, or arbitrary integer smaller than the number of descriptive attributes.
- `PrintAllModels = Yes/No` : If `Yes`, CLUS will print all base predictive models of an ensemble in the output file. Since the user does not usually inspect the models in an ensemble, the default setting is `No`.
- `PrintAllModelFiles = Yes/No`: If `Yes`, CLUS will save all base predictive models of an ensemble in the model file. The default setting is `No`, which avoids creating very large model files.

- `Optimize = Yes/No` : If `Yes`, CLUS will optimize memory usage during learning. The default setting is `No`.
- `OOBestimate = Yes/No` : If `Yes`, out-of-bag estimate of the error/performance of the ensemble will be performed. The default setting is `No`.
- `FeatureRanking = Yes/No` : If `Yes`, feature ranking via random forests will be performed. The default setting is `No`. Note that this setting requires that `OOBestimate` is set to `Yes`.

We present below an excerpt of a settings file that is used to construct random forests with 100 base predictive models. The selected voting scheme is probability distribution and the feature subset size is logarithmic. The memory optimization is turned on and the error is estimated using OOB.

```
[Ensembles]
Iterations = 100
EnsembleMethod = RForest
VotingType = ProbabilityDistribution
SelectRandomSubspaces = 0
Optimize = Yes
OOBestimate = Yes
FeatureRanking = No
```

B.3 Settings for beam search induction of PCTs

In the CLUS system, the predictive clustering trees can be constructed by using the standard top-down induction of decision trees strategy (a default setting in CLUS) or by using the beam search strategy. Here, we shortly describe the parameters (discussed in Section 7.3) that control the beam search induction of PCTs and give an excerpt of a settings file. Note that the beam search strategy was introduced in CLUS to easier pushing of user constraints in the decision trees induction process.

The `Beam` section from the settings file contains the following options:

- `SizePenalty` = r : sets the size penalty parameter used in the beam heuristic (the α parameter). It is a soft size constraint, with a default value of 0.1.
- `BeamWidth` = n : sets the width of the beam, i.e., the number of trees that are kept in the beam. Its default value is 10.

- `MaxSize` = n : sets the maximal size of the trees in the beam. This is a hard size constraint. By default, max size is set to -1, meaning `Infinity`.
- `BeamSimilarity` = r : sets the similarity influence on the overall beam heuristic (the β parameter). The default value is 0.0, meaning that no similarity constraint is used.
- `DistSyntacticConstr = None` : gives a subtree as a syntactic constraint for the trees in the beam. It can be used to guide the beam towards trees that are more different or more similar to the provided one, by manipulating the β parameter. Its default value is `None`.
- `BeamToForest = Yes/No` : If `Yes`, then the trees in the beam are combined into an ensemble. The default value is `No`.

Let us assume that we want to get 20 trees, each smaller than 25 nodes in total (this means we need to make the contribution of the soft size constraint to the overall heuristic score very small and use a hard size constraint instead). We also require that the trees are not similar to each other too much and we are not interested in combining them in an ensemble. This will be done using the following excerpt from the settings file:

```
[Beam]
SizePenalty = 0.00001
BeamWidth = 20
MaxSize = 25
BeamSimilarity = 0.9
BeamToForest = No
```

Appendix C: Bibliography

C.1 Publications related to this thesis

C.1.1 Journal papers:

- [Dimitrovski *et al.*(2011)] Dimitrovski, I., Kocev, C., Loskovska, S., and Džeroski, S. (2011). Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Pattern Recognition*, to appear.

- [Kocev *et al.*(2009)] Kocev, D., Džeroski, S., White, M., Newell, G., and Griffionen, 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 *et al.*(2010)a] Kocev, D., Naumovski, 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.

- [Kocev *et al.*(2011)] Kocev, D., Vens, C., Struyf, J., and Džeroski, S. (2011). Ensembles for predicting structured outputs. *Data Mining and Knowledge Discovery*, submission in preparation.

- [Schietgat *et al.*(2010)] Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., and Džeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, **11**(2), 1–14.

C.1.2 Book chapters

- [Vens *et al.*(2010)] Vens, C., Schietgat, L., Struyf, J., Blockeel, H., Kocev, D., and Džeroski, S. (2010). Predicting gene function using predictive clustering trees. *Inductive databases and constraint-based data mining*, Springer, p.365–387.

C.1.3 Conference and workshop papers:

- [Aleksovski *et al.*(2009)] Aleksovski, D., Kocev, C., and Džeroski, S. (2009). Evaluation of distance measures for hierarchical multi-label classification in functional genomics. In *Proceedings of the 1st International Workshop on Learning from Multi-Label data at ECML/PKDD 2009 (MLD 09)* , p.5–15.

- [Dimitrovski *et al.*(2008)] Dimitrovski, I., Kocev, C., Loskovska, S., and Džeroski, S. (2008). Hierchical annotation of medical images. In *Proceedings of the 11th International Multiconference Information Society (IS 2008)* , vol. A, p.174–181.

- [Dimitrovski *et al.*(2010)a] Dimitrovski, I., Kocev, C., Loskovska, S., and Džeroski, S. (2010). ImageCLEF 2009 medical image annotation task : PCTs for hierarchical multi-label classification. In *Multilingual information access evaluation II : multimedia experiments : revised selected papers* , LNCS 6242, p.231–238.

- [Dimitrovski *et al.*(2010)b] Dimitrovski, I., Kocev, C., Loskovska, S., and Džeroski, S. (2010). Detection of visual concepts and annotation of images using ensembles of trees for hierarchical multi-label classification. In *Recognizing patterns in signals, speech, images and videos : ICPR 2010 Contests* , LNCS 6388, p.152–162.

- [Dimitrovski *et al.*(2010)c] Dimitrovski, I., Kocev, C., Loskovska, S., and Džeroski, S. (2010). Detection of visual concepts and annotation of images using predictive clustering trees. In *CLEF 2010 Labs and Workshops* , p.50.

- [Kocev *et al.*(2008)] Kocev, D., Slavkov, I., and Džeroski, S. (2008). More is better: ranking with multiple targets for biomarker discovery. In *Proceedings of the 2nd International Workshop on Machine Learning in Systems Biology (MLSB 2008)* , vol. A, p.133.

- [Kocev *et al.*(2006)] Kocev, C., Struyf, J., and Džeroski, S. (2006). Similarity constraints in beam-search induction of predictive clustering trees. In *Proceedings of the 9th International Multiconference Information Society (IS 2006)* , vol. A, p.267–270.

- [Kocev *et al.*(2007)a] Kocev, C., Struyf, J., and Džeroski, S. (2007). Beam search induction and similarity constraints for predictive clustering trees. In *Proceedings of the 5th International Workshop on Knowledge Discovery in Inductive Databases (KDID 2006)* , LNCS 4747, p.134–151.

- [Kocev *et al.*(2007)b] Kocev, D., Vens, C., Struyf, J., and Džeroski, S. (2007). Ensembles of multi-objective decision trees. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)* , LNCS 4701, p.624–631.

- [Kocev *et al.*(2007)c] Kocev, D., Džeroski, S., White, M., Newell, G., and Griffionen, P. (2007). Ensembles of multi-objective regression trees : a case study for predicting the condition of remnant indigenous vegetation. In *Proceedings of the 10th International Multiconference Information Society (IS 2007)* , vol. A, p.210–213.

C.2 Other publications

C.2.1 Journal papers:

- [Debeljak *et al.*(2009)a] Debeljak, M., Kocev, D., Towers, W., Jones, M., Griffiths, B., Hallett, P. (2009). Potential of multi-objective models for risk-based mapping of the resilience characteristics of soils : demonstration at a national level. *Soil use and management* , **25**(1), 66–77.

- [Debeljak *et al.*(2010)] Debeljak, M., Squire, G., Kocev, D., Hawes, C., Young, M., and Džeroski, S. (2010). Analysis of time series data on agroecosystem vegetation using predictive clustering trees. *Ecological Modelling*, `http://dx.doi.org/10.1016/j.ecolmodel.2010.10.021`, in press.

- [Keller *et al.*(2011)] Keller, R., Kocev, D., and Džeroski, S. (2011). Trait-based risk assessment for invasive species: high performance across diverse taxonomic groups, geographic ranges and machine learning/statistical tools. *Diversity and Distributions*, `http://dx.doi.org/10.1111/j.1472-4642.2011.00748.x`, in press.

C.2.2 Conference and workshop papers:

- [Kocev *et al.*(2005)] Kocev, C., Loskovska, S., Struyf, J., and Džeroski, S. (2005). (Inductive) querying environment for predictive clustering trees. In *Proceedings of the 2nd Balkan Conference in Informatics (BCI 2005)* , p.193–199.

- [Naumoski *et al.*(2009)] Naumoski, A., Kocev, D., Atanasova, N., Mitreski, K., Krstić, S., and Džeroski, S. (2009). Predicting chemical parameters of the water from diatom abudance in lake Prespa and its tributaries. In *Information technologies in environmental engineering: Proceedings of the 4th International ICSC Symposium* , Springer, p.264–277.

- [Kocev *et al.*(2010)b] Kocev, D., Ženko, B., Paul, P., Kuijl, C., Neefjes, J., and Džeroski, S. (2010). Predictive clustering relates gene annotations to phenotype properties extracted from images. In *Proceedings of the 4th International Workshop on Machine Learning in Systems Biology (MLSB 2010)* , p.137–139.

- [Gibičar *et al.*(2006)] Gibičar, D., Kocev, D., Ženko, B., Horvat, M., Džeroski, S., Fajon, V., and Mazzola, B. (2006). Evaluation of biomarkers of exposure and effects of mercury using machine-learning methods. In *Toxicology letters: EUROTOX 2006*, **164S**, p.S13-S14.

- [Debeljak *et al.*(2006)] Debeljak, M., Griffiths, B., Hallett, P., Kocev, D., Towers, W., Džeroski, S. (2006). Mapping the resilience characteristics of soil by using multi-objective regression models. In *Proceedings of the 6th European conference on Ecological Modelling (ECEM 07)* , p.133-134.

- [Atanasova *et al.*(2009)] Atanasova, N., Gal, G., Škerjanec, M., Kocev, D., Džeroski, S., and Kompare, B. (2009). Investigation of the control mechanisms of the food web in lake Kinneret using multi-target regression trees. In *Proceedings of the ISEM 2009 conference: Ecological modelling for enhanced sustainability in management*, p.45.

- [Debeljak *et al.*(2009)b] Debeljak, M., Squire, G., Kocev, D., Hawes, C., Young, M., and Džeroski, S. (2009). Analysis of time series data on the economic and ecological components of vegetation by using predictive clustering trees. *Proceedings of the ISEM 2009 conference: Ecological modelling for enhanced sustainability in management*, p.75.

Appendix D: Biography

Dragi Kocev was born on 23.05.1982 in Strumica, Macedonia. He attended secondary school in Strumica and finished gymnasium majoring in natural sciences and mathematics. In 2000 he started his studies at the Faculty of Electrical Engineering, University Ss Cyrill and Methodius in Skopje, Macedonia. He was enrolled in a 9 semester BSc program in the area of Computer Engineering, Information Science and Automatics. He finished his undergraduate study *summa cum laude* in 2005. He defended his BSc thesis titled Inductive querying environment for predictive clustering trees in September 2005, under the supervision of professor Suzana Loškovska and co-supervision of professor Sašo Džeroski. During the secondary and undergraduate study he held a state scholarship for talented students (awarded by the Ministry of Education of Macedonia).

In the fall of 2005, he started his graduate studies at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia. He is enrolled in the PhD program titled "New Media and e-Science" under the supervision of professor Sašo Džeroski. He holds a scholarship for doctoral studies awarded by the Slovene Human Resources and Scholarship Fund Ad futura since the fall of 2005. From January 2007 until September 2009, he also held a scholarship of the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia. From October 2009, he is employed at the same department. He has collaborated on EU funded projects, such as IQ (Inductive Queries for Mining Patterns and Models) and PHAGOSYS (Systems biology of phagosome formation and maturation – modulation by intracellular pathogens), and on bilateral projects between Slovenia, on one hand, and Macedonia, Croatia and France, on the other.

His research is in the field of data mining and includes the study, development and application of different data mining algorithms. His current research is concerned with developing ensemble methods for the prediction of structured outputs (e.g., predicting multiple targets, hierarchical classification...). He has applied the developed algorithms to problems in ecological modelling and bioinformatics. He has published his work in several journal papers and has presented it at several international conferences and workshops, both in the areas of data mining and the respective domains of the case studies.