Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 00 (2012) 1–26
Pattern Recognition

Tree Ensembles for Predicting Structured Outputs

Dragi Kocev[a,*], Celine Vens[b], Jan Struyf[b], Sašo Džeroski[a,c,d]

[a]*Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[b]*Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium*
[c]*International Postgraduate School Jožef Stefan, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[d]*Centre of Excellence for Integrated Approaches in Chemistry and Biology of Proteins (CIPKeBiP), Jamova cesta 39, 1000 Ljubljana, Slovenia*

Abstract

In this article, we address the task of learning models for predicting structured outputs. We consider both global and local prediction of structured outputs, the former 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. We use ensemble methods and apply them in the context of predicting structured outputs. We propose 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 two types of ensembles (bagging and random forests) of predictive clustering trees for global and local prediction of different types of structured outputs. The types of outputs considered correspond to different predictive modelling tasks: multi-target regression, multi-target classification, and hierarchical multi-label classification. 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 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). The results show that both global and local tree ensembles perform better than the single model counterparts in terms of predictive power. Global and local tree 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.

Keywords: ensemble methods, predictive clustering trees, structured outputs, multi-target regression, multi-target classification, hierarchical multi-label classification

1. Introduction

Supervised learning is one of the most widely researched and investigated areas of machine learning. The goal in supervised learning is to learn, from a set of examples with known class, a function that outputs a prediction for the class of a previously unseen example. If the examples belong to two classes (e.g., the example has some property or not) the task is called binary classification. The task where the examples can belong to a single class from a given

*Corresponding author (telephone: +386 1 477 3639)
Email addresses: Dragi.Kocev@ijs.si (Dragi Kocev), Celine.Vens@cs.kuleuven.be (Celine Vens), Jan.Struyf@cs.kuleuven.be (Jan Struyf), Saso.Dzeroski@ijs.si (Sašo Džeroski)

set of m classes ($m \geq 3$) is known as multi-class classification. The case where the output is a real value is called regression.

However, in many real life problems of predictive modelling the output (i.e., the target) 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 gene function, finding the most important genes for a given disease, predicting toxicity of 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 pages). Having in mind the needs of these application domains and the increasing quantities of structured data, Yang and Wu [1] and Kriegel et al. [2] listed the task of "mining complex knowledge from complex data" as one of the most challenging problems in machine learning.

A variety of methods, specialized in predicting a given type of structured output (e.g., a hierarchy of classes [3]), have been proposed [4]. These methods can be categorized into two groups of methods for solving the problem of predicting structured outputs [4, 3]: (1) local methods that predict component(s) of the output and then combine the individual models to get the overall model and (2) global methods that predict the complete structure as a whole (also known as 'big-bang' approaches). The global methods have several advantages over the local methods. First, they exploit and use the dependencies that exist between the components of the structured output in the model learning phase, which can result in better predictive performance. Next, they are typically more efficient: it can easily happen that the number of components in the output is very large (e.g., hierarchies in functional genomics can have several thousands of components), 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 built for each of the components.

The predictive models that we consider in this article are predictive clustering trees (PCTs). PCTs belong to the group of global methods. PCTs offer a unifying approach for dealing with different types of structured outputs and construct the predictive models very efficiently. They are able to make predictions for several types of structured outputs: tuples of continuous/discrete variables, hierarchies of classes, and time series. More details about the PCT framework can be found in [5, 6, 7, 8, 9].

PCTs can be considered as a generalization of standard decision trees towards predicting structured outputs. Although they offer easily interpretable trees with good predictive performance, they inherit the stability issues of decision trees [10]. Namely, change in just a few training examples can sometimes drastically change the structure of the tree. Breiman [11] states that un-stable predictive models (such as decision trees) could be combined into an ensemble to improve their predictive performance. An ensemble is a set of (base) predictive models, whose output is combined. For basic classification and regression tasks, it is widely accepted that an ensemble lifts the predictive performance of its base predictive models [12]. However, for the task of predicting structured outputs, this issue has not been thoroughly investigated. Moreover, in the case where the base predictive models are decision trees, Bauer and Kohavi [13] conclude that the ensemble's increase in performance is stronger if the trees are unpruned, i.e., allowed to overfit. On the other hand, Blockeel et al. [14] state that PCTs for structured outputs show less overfitting 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.

Furthermore, 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 a combination of ensembles of local predictive models. Generally, an ensemble is known to perform better than its base learner if the base learner is accurate and diverse [15]. While the superior predictive performance of global models has been shown before [8], less is known about their diversity or instability (i.e., whether they produce different errors with small changes to the training data). It is expected that a PCT for predicting structured outputs, especially in the case of hierarchical classification, is less unstable than a PCT for predicting components of the output. It is also not clear which approach will be more efficient, both in terms of running time and size of the predictive models.

In this article, we investigate the aforementioned questions. We use bagging and random forests as ensemble learning methods, since they are the two most widely used ensemble learning methods in the context of decision trees [16]. We consider two structured output machine learning tasks: predicting multiple target variables and hierarchical multi-label classification. We perform an extensive empirical evaluation of the proposed methods over a variety of benchmark datasets.

The paper is based on our previous work by Kocev et al. [7] and Schietgat et al. [17]. Kocev et al. [7] conducted an initial comparison of different ensemble schemes using predictive clustering trees in the context of predicting multiple targets. Schietgat et al. [17] introduced bagging of PCTs for hierarchical multi-label classification (HMC) in the context of functional genomics. The present paper extends the previous work [7][17] in the following directions:

- The tasks of predicting multiple target variables and hierarchical multi-label classification are discussed jointly. Formal definitions of the considered tasks are provided, as well as an extensive discussion of the related work.
- The experimental evaluation is performed on a much wider selection of datasets, from various domains. The performance of ensembles with different number of base predictive models is evaluated and saturation curves are provided.
- A better methodology for performance comparisons is used and a more detailed discussion of the results is presented. Friedman tests combined with a Nemenyi post-hoc test are used to compare different ensemble schemes, instead of performing pairwise comparisons.
- Global random forests of PCTs for HMC are introduced and evaluated. Local ensembles (both bagging and random forests) of PCTs for HMC are introduced and evaluated.
- A computational complexity analysis of the proposed methods is performed, which is consistent with the empirical evidence. Global tree ensembles are most efficient, especially random forests, and are scalable to large datasets.

The remainder of this paper is organized as follows. In Section 2, the considered machine learning tasks are formally defined. Section 3 first explains the predictive clustering trees framework and the extensions for predicting multiple targets and hierarchical multi-label classification. It then describes the ensemble methods and their extension for predicting structured outputs. The design of the experiments, the descriptions of the datasets, the evaluation measures and the parameter settings for the algorithms are given in Section 4. Section 5 presents and discusses the obtained results. The related work is presented in Section 6. Finally, the conclusions are stated in Section 7.

2. Machine learning tasks

The work presented in this article concerns the learning of ensembles for predicting structured outputs. First, we formally describe the machine learning tasks that we consider here: predicting multiple targets and hierarchical multi-label classification. We follow the suggestions by Džeroski [18], where predictive modelling is defined for arbitrary types of input and output data. In particular, we describe the tasks of predicting multiple targets and hierarchical multi-label classification.

2.1. Predicting multiple targets

The task of predicting multiple targets was previously referred to as multi-objective prediction [6, 7, 19]. However, the term 'multi-objective' is already established in the area of optimization. We will thus use the term 'predicting multiple targets' or multi-target prediction (resp. multi-target classification and regression). We define the task of predicting multiple targets as follows.

Given:

- A description space X that consists of tuples of values of primitive data types (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 consists of 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 in 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.

Here, the function f is represented with decision trees, i.e., predictive clustering trees or ensembles thereof. If the tuples from Y (the target space) consist of continuous/numeric variables then the task at hand is multi-target regression. Likewise, if the tuples from Y consist of discrete/nominal variables then the task is called multi-target classification.

The task of multi-target classification can be seen as a generalization of the multi-label classification task [20]. Namely, multi-label classification is concerned with learning from examples, where each example is associated with multiple labels. These multiple labels belong to a predefined set of labels. The goal is then to construct a predictive model that will provide a list of relevant labels for a given, previously unseen example. For addressing the task multi-label classification using multi-target classification, each of the labels can be considered as a binary (0/1) discrete target variable: The label vector is then a target tuple of binary discrete variables. The labels for which 1's are predicted are thus considered as relevant labels.

2.2. Hierarchical 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 [10, 21]. The unseen examples are classified into a single class from a set of possible classes.

Hierarchical classification differs from traditional classification that 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, if an example can belong simultaneously to multiple classes that can follow multiple paths from the root class, then the task is called hierarchical multi-label classification (HMC) [3, 8]. This is the setting we use in this article.

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 (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 (e.g., 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 in 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.

3. Tree ensembles for predicting structured outputs

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

3.1. PCTs for structured outputs

The Predictive Clustering Trees (PCTs) framework views 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 [22], which is written in Java and is open-source software licensed under the GNU General Public Licence. The CLUS system is available for download at `http://clus.sourceforge.net`.

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 article, we consider three different classes of datatypes T_Y: tuples of discrete values, tuples of real values, and hierarchies. For each type T_Y, CLUS needs two functions to be defined. The prototype function returns a representative structured value given a set of such structured values. For example, given a set of tuples of discrete variables, the prototype function computes and returns a tuple of discrete variables that is representative for the whole set. 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 [10]. The algorithm is presented in Table 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 the BestTest procedure in Table 1). By maximizing the variance reduction, the cluster homogeneity is maximized and the predictive performance is improved.

Table 1: The top-down induction algorithm for PCTs.

procedure PCT
Input: A dataset E
Output: A predictive clustering tree
1: $(t^*, h^*, \mathcal{P}^*)$ = BestTest(E)
2: **if** $t^* \neq$ *none* **then**
3: **for each** $E_i \in \mathcal{P}^*$ **do**
4: $tree_i$ = PCT(E_i)
5: **return** node(t^*, $\bigcup_i \{tree_i\}$)
6: **else**
7: **return** leaf(Prototype(E))

procedure BestTest
Input: A dataset E
Output: the best test (t^*), its heuristic score (h^*) and the partition ($\mathcal{P}^*$) it induces on the dataset (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** $(h > h^*) \wedge$ 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 (i.e., the C4.5 algorithm [23]) 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, PCTs have been instantiated for the following tasks: multi-target prediction [6, 7], hierarchical multi-label classification [8] and prediction of time-series [9]. In this article, we focus on the first two tasks.

3.1.1. PCTs for predicting 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.

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 target variables are normalized, so that each target variable contributes equally to the overall variance. This is due to the fact that the target variables can have completely different ranges. Namely, if one of the target variables is in the range (0, 1) and another in the range (10, 100) and normalization is not used, then the values of the second variable will contribute much more to the overall score than the values of the first variable. In addition, CLUS-MTDT supports weighting of the (normalized values of the) target variables so that the variance function gives more weight to some variables and

less to others. 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.

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 [24]). The CLUS system also implements other variance functions, such as reduced error, gain ratio and the *m*-estimate. 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.

3.1.2. PCTs for hierarchical multi–label classification

Hierarchical multi-label classification is a variant of classification where a single example may belong to multiple classes at the same time and the classes are organized in a form of 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 [3] give a detailed overview of the possible application areas and the available approaches to HMC.

Silla and Freitas [3] 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 single or multiple paths, i.e., predicted classes, to each instance.
- Σ = non-mandatory leaf-node prediction: an instance can be labelled with a label at any level of the taxonomy.
- Ω = tree or directed acyclic graph: the algorithm can handle both tree-shaped and DAG hierarchies of classes.
- Θ = global classifier: the algorithm constructs a single model valid for all classes.

CLUS-HMC is the instantiation (with the distances and prototypes as defined below) of the PCT algorithm for hierarchical classification implemented in the CLUS system. The variance and prototype are defined as follows [8]. 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, $w(c) = w_0 \cdot \{w(p(c))\}$, where $p(c)$ denotes the parent of class c and $0 < w_0 < 1$).

For example, consider the toy class hierarchy shown in Figure 1(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 1(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 of class labels). The distance is then calculated as follows:

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

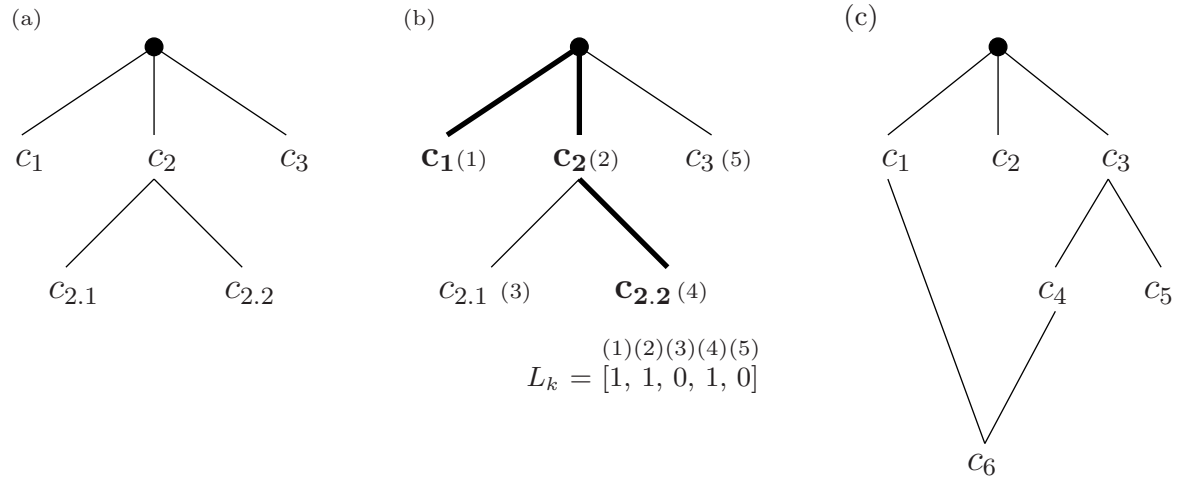


Figure 1: Toy examples of hierarchies structured as a tree and a DAG.(a) Class label names contain information about the position in the hierarchy, e.g., $c_{2.1}$ is a subclass of c_2. (b) The set of classes $S_1 = \{c_1, c_2, c_{2.2}\}$, shown in bold in the hierarchy, represented as a vector (L_k). (c) A class hierarchy structured as a DAG. The class c_6 has two parents: c_1 and c_4.

This example calculates the distance between two data instances that belong to classes organized into a tree-shaped hierarchy. As discussed at the beginning of this section, the proposed methods also support hierarchies in the form of a DAG. Figure 1(c) depicts an example of a DAG structured hierarchy. The main difference between tree-shaped and DAG hierarchies is that the classes can have multiple parent classes at different levels/depths in the hierarchy. Hence, the depth of a class is not unique: classes do not have a single path from the top-node (for example see class c_6 in Figure 1(c)). This makes it impossible to calculate the value of the weight as $w(c) = w_0 \cdot \{w(p(c))\}$. To resolve this, several approaches have been proposed to compute the weight of a class with multiple parents:

- Flattening the DAG into a tree by copying the subtrees that have multiple parents ($w(c) = w_0 \cdot \sum_j \{w(p_j(c))\}$, where $p(c)$ denotes the parent of class c). The more paths in DAG lead to a class, the larger weight is assigned to this class by this method.
- Take the weight of the parent with largest depth ($w(c) = w_0 \cdot \min_j \{w(p_j(c))\}$). The drawback of this approach is that assigns a small weight to a class with multiple parents that appear both close to the top-level and deep in the hierarchy.
- Take the weight of the parent with smallest depth ($w(c) = w_0 \cdot \max_j \{w(p_j(c))\}$). It guarantees a large weight for classes that appear near the top-level of the hierarchy, however it does not satisfy the inequality $w(c) < w(par_j(c))$.
- Take the average weight of all parents ($w(c) = w_0 \cdot \text{avg}_j \{w(p_j(c))\}$). This approach is a compromise between the previous two approaches.

Note that all these weighting schemes become equivalent for tree-shaped hierarchies. Vens et al. [8] perform an extensive experimental evaluation of the weighting schemes and recommend to use as a weight of a given class the average over the weights of all its parents (i.e., $w(c) = w_0 \cdot \text{avg}_j\{w(par_j(c))\}$). We thus use the average weighting scheme in this study.

Recall that the instantiation of PCTs for a given task requires proper instantiation of the variance and prototype functions. The variance function for the HMC task is instantiated by using the weighted Euclidean distance measure (as given above), which is further used to select the best test for a given node by calculating the heuristic score (line 4 from the algorithm in Table 1). We now discuss the instantiation of the prototype function for the HMC task.

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 at the leaf can be obtained by applying a user defined threshold τ to the probability; if the i-th component of $\bar{L}$ is above τ then the examples belong to class c_i. When a PCT is making a prediction, it preserves the hierarchy constraint (the predictions comply with 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.

3.2. Ensembles of PCTs for predicting structured outputs

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

3.2.1. Constructing ensembles of PCTs

A necessary condition for an ensemble to have better predictive performance than any of its individual members is that the base predictive models are accurate and diverse [15]. However, there is an ongoing scientific debate concerning the trade-off between the accuracy and the diversity of the base predictive models in the context of the ensembles' predictive performance [25, 26]. 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 [11, 27], by changing the attribute values of the examples [28], by manipulating the feature space [29, 30]) and by manipulating the learning algorithm itself [29, 31].

We have implemented the bagging and random forests methods within the CLUS system. These two ensemble learning techniques are most widely known and have primarily been used in the context of decision trees [16]. The algorithms of these ensemble learning methods are presented in Table 2. For the random forests method (Table 2, right), the PCT algorithm for structured prediction was changed to *PCT_rnd*: randomized version of the selection of attributes was implemented, which replaced the standard selection of attributes.

3.2.2. Bagging

Bagging [11] is an ensemble method that constructs the different classifiers by making bootstrap replicates of the training set and using each of these replicates to construct a predictive model (Table 2, 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 [11] 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.

Table 2: The ensemble learning algorithms: bagging and random forests. 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 considered at each node during tree construction for random forests.

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

3.2.3. Random forests

A random forest [29] 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 2, 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 classifiers is presented in Table 2.

3.2.4. 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 [13, 11], or by taking more complex aggregation schemes [25].

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 the prediction of the ensemble, we take the average per target of the predictions of the base classifiers. For the datasets for hierarchical classification we also use the average of the predictions and apply the thresholding described in Section 3.1.2. We obtain the ensemble predictions for the datasets with multiple discrete targets using probability distribution voting (as suggested by Bauer and Kohavi [13]) per target.

3.3. Local prediction of structured outputs with PCTs and ensembles

The presented structured output learning algorithms (CLUS-MTDT and CLUS-HMC) belong to the group of methods known as 'big-bang' or global predictive models [3, 4]. 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: flat classification, local classifiers per level, local classifiers per node, and local classifiers per parent node (see [3] for details).

Vens et al. [8] 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. [8], presented in Figure 2(a), 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 trees uses few instances, as 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.

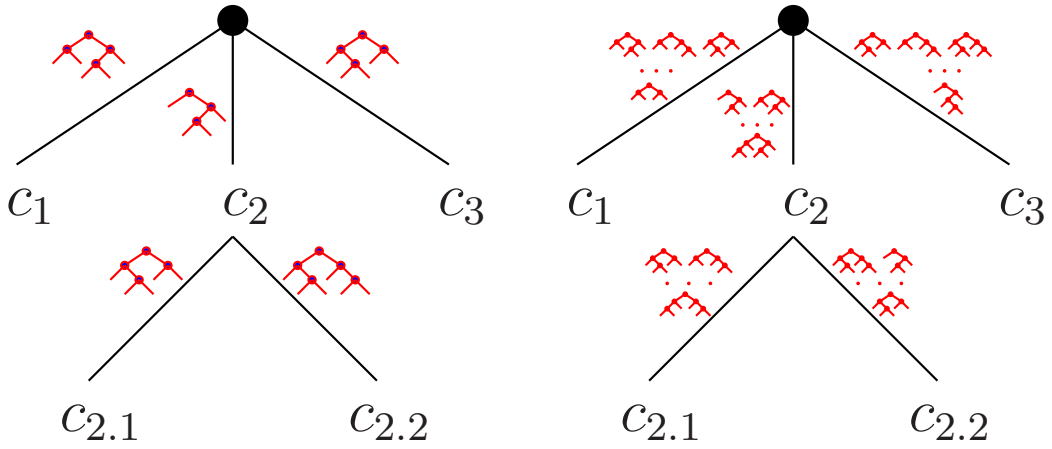


Figure 2: An illustration of the hierarchical single-label classification approach used by Vens et al. [8]. The local classifiers at each branch from the hierarchy are: (a) decision trees and (b) ensembles of decision trees.

The resulting HSC tree predicts the conditional probability $P(c|par(c))$. A new instance is predicted by recursive application of the product rule $P(c) = \min_j P(c \mid par_j(c)) \cdot P(par_j(c))$ (with $par_j(c)$ denoting the j-th parent of c in the case of a DAG), starting from the 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.

In this article, we extend the approach of Vens et al. [8] 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 (similar to the one shown in Figure 2(a)). 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 2(b)). We prefer here the second approach since it is closer to the learning of local classifiers for predicting multiple target variables.

3.4. Computational complexity

In this section, we analyze and discuss the computational complexity aspects of the proposed methods. We begin by deriving the computational complexity of a single tree. We then analyze the computational complexity of a single ensemble. Next, we discuss the computational complexity of the trees and the ensembles for local prediction of structured outputs. After that, we discuss the computational complexity of the trees and the ensembles for global prediction of structured outputs. Finally, we compare the computational complexities of the models for local and global prediction of the structured outputs.

We assume that the training set contains N instances and D descriptive attributes (of which M are continuous). Furthermore, S is the size of the output measured as number of target variables or the number of classes in the hierarchy and the hierarchy contains G edges. The ensembles contain k base predictive models.

Single tree Three procedures contribute to the computational complexity of the tree learning algorithm described in Table 1. The procedures are executed at each node of the tree and they include: sorting the values of the M numeric descriptive attributes with a cost of $O(MN \log N)$, calculating the best split for a single target variable which costs $O(DN)$, and applying the split to the training instances with a cost of $O(N)$. Furthermore, we assume, as in [32], that the tree is balanced and bushy. This means that the depth of the tree is in the order of $\log N$, i.e., $O(\log N)$.

Having this in mind and considering that $M = O(D)$, the total computational cost of constructing a single tree is $O(DN \log^2 N) + O(DN \log N) + O(N \log N)$.

Single ensemble The computational complexity of constructing an ensemble depends on the computational complexity of constructing its base predictive models. In general, the computation complexity of learning an ensemble with k base predictive models is k times higher than the computational complexity of learning a single base predictive model. Thus, the overall computational complexity of an ensemble method can be calculated as $k(O(DN \log^2 N) + O(DN \log N) + O(N \log N))$. However, ensemble methods perform sampling of the instances and/or the features, thus potentially 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 instance/feature sampling. The creation of the bootstrap replicates of the training set for bagging and random forests has a computational complexity of $O(N)$, while the number of instances used to train the base predictive models is not N, but $N' = 0.632 \cdot N$ [11]. The random forests, in addition to the bootstrap sampling of the instances, also perform random sampling of the features with a selection function $f(D)$, i.e., the number of descriptive attributes (considered at each step) is $D' = f(D)$. The random sampling of the features at each node costs $O(D' \log N')$ Considering this, the overall computational complexity of constructing a random forest is $k(O(D'N' \log^2 N') + O(D'N' \log N') + O(N' \log N') + O(N) + O(D' \log N'))$.

Local models We extend this analysis for local prediction of structured outputs. For the task of predicting multiple targets, a model (a tree or an ensemble) is constructed for each target separately, thus the computational complexity is S times higher than the computational complexity of a model for a single target attribute. For the HMC task, a model is constructed for each edge in the hierarchy (see Figure 2). Hence, the computational complexity of this architecture of models is at most G times higher than the computational complexity of a single model. However, the computational complexity of an architecture of models 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 models 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 model.

Global models The derivation of the computational complexity of constructing PCTs for global prediction of structured outputs follows the same pattern as for a single PCT 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 of $O(DN)$, for the task of predicting structured outputs has a computational complexity of $O(S\,DN)$. The computational complexity for the construction of the complete tree is then as follows: $O(DN \log^2 N) + O(S\,DN \log N) + O(N \log N)$. The construction of an ensemble that consists of k PCTs for global prediction costs k times more than the 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.

Comparison of local and global models We further compare the computational complexity of local and global models (both PCTs and ensembles) for prediction of structured outputs. The dominant terms in the computational complexity of local models (a set of PCTs) for multiple targets is $O(S\,DN \log^2 N)$ and local models (a set of PCTs) for HSC is $O(GDN \log^2 N)$. Let us assume that $S < \log N$. This means that the dominant term in the computational complexity of global models for both multiple targets and HSC is $O(DN \log^2 N)$. Considering this, global models have $O(S)$ and $O(G)$ times lower computational complexity than local models for predicting multiple targets and HMC, respectively. On the other hand, if we assume that $S > \log N$, then the dominant term in the computational complexity of global models is $O(S\,DN \log N)$. In this case, global models have $O(\log N)$ or $O(\frac{G}{S} \log N)$ times lower computational complexity than local models for predicting multiple targets or HMC, respectively. Moreover, for the task of HMC, if the target hierarchy is tree-shaped then $G = S - 1$, i.e., $G \approx S$ and the global models have $O(\log N)$ times smaller computational complexity than local models. Furthermore, when the target hierarchy is a DAG, the computational advantage of the global models depends of the average number of parents that classes have. If the classes have more parents on average, the $\frac{G}{S}$ ratio will be larger and the computational complexity of the global models will be lower.

Global models have lower computational complexity than local models mainly due to the multiple repetitions of the sorting of the numeric attributes in the latter. However, the difference in the computational complexity is further amplified by the fact that global models are smaller, on average, than the local models. Also, implementation-wise, global models are faster to construct also because of some properties of most CPU architectures. Namely, it is faster and easier to multiply/add/subtract two arrays with size S, than to multiply/add/subtract S times two arrays with size

one [33].

Random forests are very efficient ensemble methods and have very good predictive performance. Thus, we discuss in more detail the computational complexity of random forests for global prediction of structured outputs. The upper bound of the computational complexity of global random forests is $k(O(D'N' \log^2 N') + O(S\,D'N' \log N'))$. The complexity of global random forests depends linearly on the number of base predictive models ($O(k)$) and logarithmically (or via some other user defined function that determines the number of features sampled) on the number of numeric descriptive attributes ($O(D') = O(\log D)$). Furthermore, if $S > \log N$ then the complexity will depend linearly on the size of the structured output and $O(N' \log^2 N')$ on the number of training instances. If $S < \log N$, then the complexity of the random forests depends $O(N' \log N'^2)$ 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 [34], 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 on 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. To summarize, global random forests (and global ensembles, in general), according to the above analyses of their computation complexity, are very efficient methods for predicting structured outputs.

4. Experimental design

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

4.1. Experimental questions

Given the methodology from Section 3, we construct several types of trees and ensembles thereof. First, we construct PCTs that predict components of the structured output: a separate tree for each variable from the target tuple (predicting multiple targets) and a separate tree for each hierarchy edge (hierarchical classification). Second, we learn PCTs that predict the entire structured output simultaneously: a tree for the complete target tuple and a tree for the complete hierarchy, respectively. Finally, we construct the ensemble models in the same manner by using both bagging and random forests.

We consider three aspects of constructing tree ensembles for predicting structured outputs: predictive performance, convergence and efficiency. We first assess the predictive performance of global and local tree ensembles and investigate whether global and local ensembles have better predictive performance than the respective single model counterparts. Moreover, we check whether the exploitation of the structure of the output can lift the predictive performance of an ensemble (i.e., global versus local ensembles). Next, we investigate the saturation/convergence of the predictive performance of global and local ensembles with respect to the number of base predictive models they consist of. Namely, we inspect the predictive performance of the ensembles at different ensemble sizes (i.e., we construct saturation curves). The goal is to check which type of ensembles, global or local, saturates at a smaller number of trees. Finally, we assess the efficiency of both global and local single predictive models and ensembles thereof by comparing the running times for and the sizes of the models obtained by the different approaches.

4.2. Descriptions of the datasets

In this section, we present the datasets that were used to evaluate the 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 3, 4, and 5, respectively.

Table 3: 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	$\|D\|/\|C\|$	T
Collembola [35]	393	8/39	3
EDM [36]	154	0/16	2
Forestry-Kras [37]	60607	0/160	11
Forestry-Slivnica-LandSat [38]	6218	0/150	2
Forestry-Slivnica-IRS [38]	2731	0/29	2
Forestry-Slivnica-SPOT [38]	2731	0/49	2
Sigmea real [39]	817	0/4	2
Soil quality [19]	1944	0/142	3
Solar-flare 1 [40]	323	10/0	3
Solar-flare 2 [40]	1066	10/0	3
Vegetation Clustering [41]	29679	0/65	11
Vegetation Condition [42]	16967	1/39	7
Water quality [43, 44]	1060	0/16	14

Table 4: 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	$\|D\|/\|C\|$	T
EDM [36]	154	0/16	2
Emotions [45]	593	0/72	6
Mediana [46]	7953	21/58	5
Scene [47]	2407	0/294	6
Sigmea real [39]	817	0/4	2
Solar-flare 1 [40]	323	10/0	3
Thyroid [40]	9172	22/7	7
Water quality [43, 44]	1060	0/16	14
Yeast [48]	2417	0/103	14

The datasets with multiple continuous targets (13 in total, see Table 3) are mainly from the domain of ecological modelling. The datasets with multiple discrete targets (9 in total, see Table 4) are from various domains: ecological modelling (*Sigmea Real* and *Water Quality*), biology (*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). Note that two datasets from the biological domain have a hierarchy organized as a DAG (they have GO in the dataset name), while the remaining datasets have tree-shaped hierarchies. For more details on the datasets, we refer the reader to the referenced literature.

4.3. 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, 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 (CC), root mean squared error ($RMSE$) and relative root mean squared error ($RRMSE$). For

Table 5: 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. Note that the values for $\mathcal{H}_d$ are not always a natural number because the hierarchy has a form of a DAG and the maximal depth of a node is calculated as the average of the depths of its parents.

Domain	N_{tr}/N_{te}	$\|D\|/\|C\|$	$\|\mathcal{H}\|$	$\mathcal{H}_d$	$\overline{\mathcal{L}}$	$\overline{\mathcal{L}_L}$
ImCLEF07D[49]	10000/1006	0/80	46	3.0	3.0	1.0
ImCLEF07A[49]	10000/1006	0/80	96	3.0	3.0	1.0
Diatoms [50]	2065/1054	0/371	377	3.0	1.95	0.94
Enron [51]	988/660	0/1001	54	3.0	5.30	2.84
Reuters [52]	3000/3000	0/47236	100	4.0	3.20	1.20
WIPO [53]	1352/358	0/74435	183	4.0	4.0	1.0
Expression–FunCat [24]	2494/1291	4/547	475	4.0	8.87	2.29
SCOP-GO [24]	6507/3336	0/2003	523	5.5	6.26	0.95
Sequence-FunCat [24]	2455/1264	2/4448	244	4.0	3.35	0.94
Yeast-GO [54]	2310/1155	5588/342	133	6.3	5.74	0.66

each of these measures, we performed tests for statistical significance and constructed saturation curves. We present here 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 [55] conducted a systematic analysis of twenty four performance measures that can be used in a classification context. They conclude that evaluation measures for classification algorithms should be chosen based on the application domain.

In our study, we used seven evaluation measures for classification: accuracy, precision, recall, F-score, the Matthews correlation coefficient, balanced accuracy (also known as Area Under the Curve) and discriminant power. We used two averaging approaches to adapt these measures for multi-class problems: micro and macro averaging (note that averaging is not needed for accuracy). The formulas for calculating the evaluation measures are given in Appendix A. Since the goal of this study is not to assess the evaluation measures themselves, we present here only the results in terms of the micro average F-score ($F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$). However, the conclusions drawn from the evaluation of the performance of the algorithms using the other measures concur with the ones presented here.

In the case of hierarchical classification, we evaluate the algorithms using the Area Under the Precision-Recall Curve ($AUPRC$), and in particular, the Area Under the Average Precision-Recall Curve ($AU\overline{PRC}$) as suggested by Vens et al. [8]. A Precision-Recall curve plots the precision of a classifier as a function of its recall. The points in the PR space are obtained by varying the value for the threshold τ from 0 to 1 with step 0.02. The precision and recall are micro averaged for all classes from the hierarchy.

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 running at 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, ≥ 2GB RAM).

4.4. 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, i.e., 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 [13], 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 number of descriptive attributes $\lfloor \log_2 |D| \rfloor + 1$, which is recommended by Breiman [29]. 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, see Table 5) and the logarithmic function is under-sampling the feature space (e.g., it will select 14 attributes from 10000 descriptive attributes).

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 [56] for the statistical evaluation of the results. We use the Friedman test [57] for statistical significance with the correction from Iman and Davenport [58]. Afterwards, to check where the statistically significant differences appear (between which algorithms), we use the Nemenyi post-hoc test [59]. We present the results from the statistical analysis with *average ranks diagrams* [56]. 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. Results and discussion

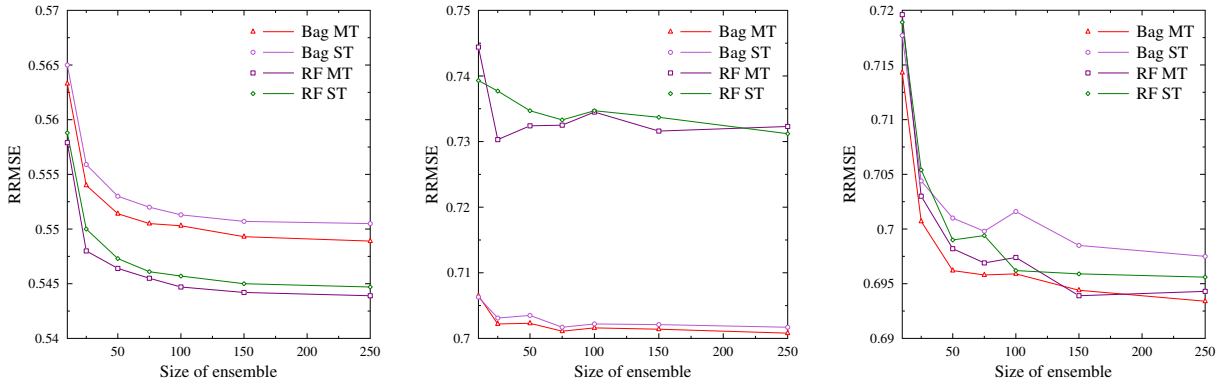
The results from the experiments can be analyzed along several dimensions. First, we present the saturation curves of the ensemble methods (both for predicting the structured outputs and the components of the outputs). We also compare single trees vs. ensembles of trees. Next, we compare models that predict the complete structured output vs. models that predict components of the structured output. Finally, we evaluate the algorithms by their efficiency in terms of running time and model size. We perform these comparisons for each task separately: multi-target regression, multi-target classification and hierarchical multi–label classification. We conclude the section with a general discussion of the experimental results.

5.1. Multi-target regression

In Figure 3, we present the saturation curves for the ensemble methods for multi-target regression. Although these curves are averaged across all target variables for a given dataset, they still provide useful insight into the performance of the algorithms. Figure 3 (a) and (b) present the saturation curves for two specific datasets, while Figure 3 (c) presents curves averaged across all datasets. We have checked at which ensemble size (saturation point) the RRMSE no longer statistically significantly changes, i.e., when adding trees to the ensemble does not increase the predictive performance significantly. For all algorithms, the differences are not statistically significant after 50 trees are added. Thus, in the remainder of the analysis, we use ensembles of 50 trees are added to the ensembles.

The statistical test in Figure 4 shows that the differences in predictive performance among the different ensemble methods are not statistically significant at the level of 0.05. For most of the datasets, the best performing method is random forests for predicting multiple targets. However, if we look at the saturation curves in Figure 3 (c), we note that, on average, multiple target bagging is best. A closer look at the results shows that, especially for the larger datasets (e.g., *Forestry-Kras* from Figure 3(a)), random forests tend to outperform bagging, both for the local and global algorithms. The difference in performance between ensembles and single PCTs is statistically significant. The PCTs for predicting multiple targets simultaneously are better (though not significantly) than the trees for predicting multiple targets separately.

Finally, we compare the algorithms by their running time and the size of the models for ensembles of 50 trees (see Figure 5). The statistical tests show that, in terms of the time efficiency, random forests for multi-target regression



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

Figure 3: Saturation curves for the different ensemble approaches to the prediction of multiple continuous targets. The curves (a) and (b) are obtained by averaging the $RRMSE$ values over all of the target variables in a dataset, while the curve (c) by averaging the $RRMSE$ values over all of the target variables in all datasets. Smaller $RRMSE$ values mean better predictive performance. Note that the scale 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*.

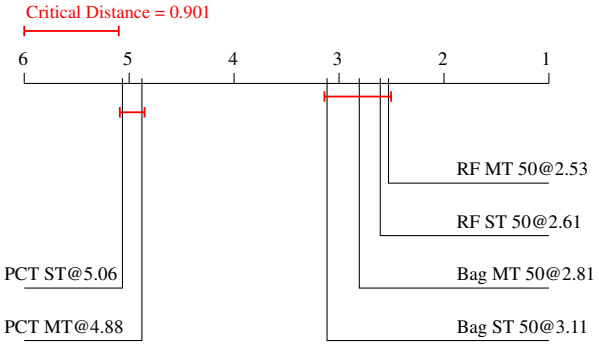
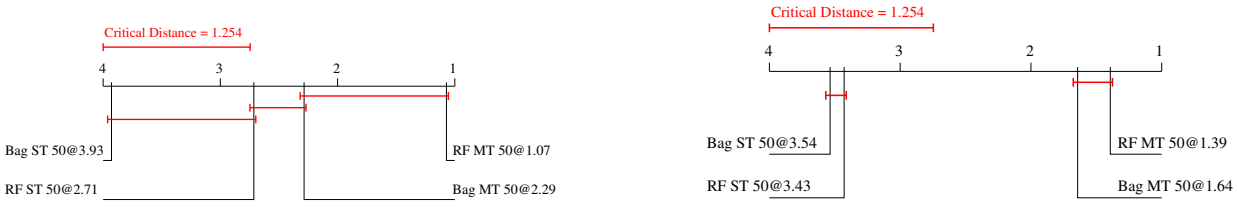


Figure 4: Average rank diagrams (with the critical distance at a significance level of 0.05) for the prediction of multiple continuous targets. The differences in performance of the algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 3, with the addition of single predictive clustering tree - *PCT*.

significantly outperform ensemble methods predicting the targets separately. Also, bagging for multiple targets is significantly faster than bagging for separate prediction of the targets. In terms of model size, both random forests and bagging for predicting multiple targets simultaneously outperform significantly the ensembles that predict multiple targets separately.

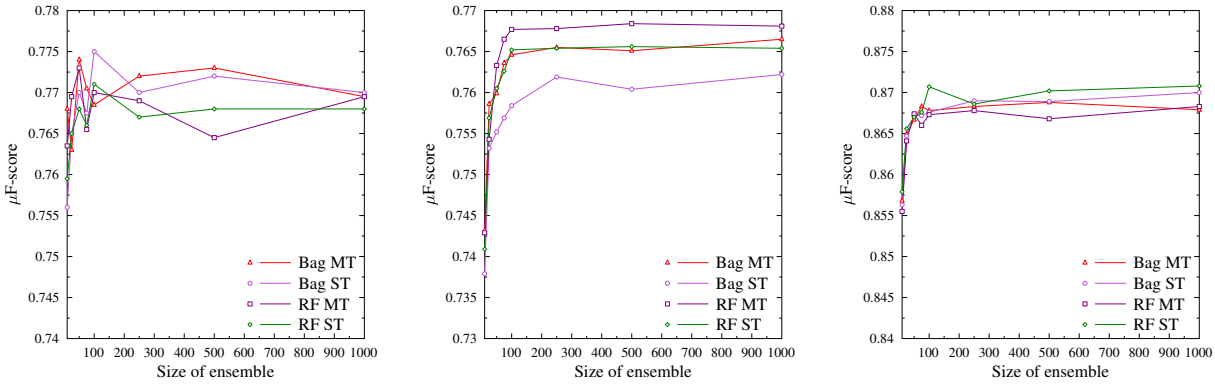


(a) *Time efficiency* (b) *Size of the models*

Figure 5: 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. Multi-target classification

In Figure 6, we present three saturation curves for the ensemble methods for multi-target classification. As for multi-target regression, the values depicted in the curves are the averages over all target variables for a given dataset (and in Figure 6(c) averaged across all datasets). As for multi-target regression, we determined the ensemble size after which the changes in predictive performance are no longer significant. The ensembles for predicting the multiple targets simultaneously saturate with 50 trees added, while the ensembles for separate prediction of the targets require more trees: 75 for random forests and 250 for bagging. After this, we select the ensembles size of 50 (Figure 7) to compare the algorithms. This is in line with the results from the saturation curves which show that ensembles for multi-target classification perform better than the ensembles for single-target classification at smaller ensemble sizes (this can be also noticed in the overall saturation curve shown in Figure 6(c)).



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

Figure 6: Saturation curves for the prediction of multiple discrete targets. The curves (a) and (b) are obtained by averaging the $\mu F - score$ values over all of the target variables in a dataset, while the curve (c) by averaging the $\mu F - score$ values over all of the target variables in all datasets. Higher $\mu F - score$ values mean better predictive performance. Note that the scale 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*.

The statistical tests reveal that there is no statistically significant difference (at the level of 0.05) in the performance of the ensemble methods (Figure 7). Bagging for predicting the multiple targets simultaneously is the best performing method (average rank 2.59) and the remaining methods have larger average ranks very close to each other (ranging from 3.0 to 3.11) with random forest for separate prediction of the targets having the largest average rank (worst performance). Similar conclusions can be made if instead of ensembles with 50 trees, we select ensembles with 75 or 250 trees. The only difference is that in these cases random forests for multiple targets have larger average ranks (i.e., the difference in performance between random forests for multiple targets and the other methods is smaller).

Both types of ensembles (multi-target and single-target classification) perform statistically significantly better than single PCTs. Furthermore, single PCTs for multi-target classification perform better (although not statistically significantly) than the PCTs for single-target classification.

Finally, we compare the ensembles by their efficiency: running times (Figure 8(a)) and size of models (Figure 8(b)). Concerning the running time, we can state that random forests for predicting multiple targets simultaneously significantly outperform bagging for predicting 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.

5.3. Hierarchical multi-label classification

In this subsection, we compare the performance of ensembles and PCTs for the tasks of HMC and HSC. We begin by presenting the saturation curves of the ensemble methods in Figure 9. Figures 9(a) and 9(b) show the saturation curves for the *SCOP-GO* and *ImCLEF07D* domains, respectively, while Figure 9(c) shows the saturation curve averaged across all domains. We determine the ensemble size after which adding trees in the ensemble does

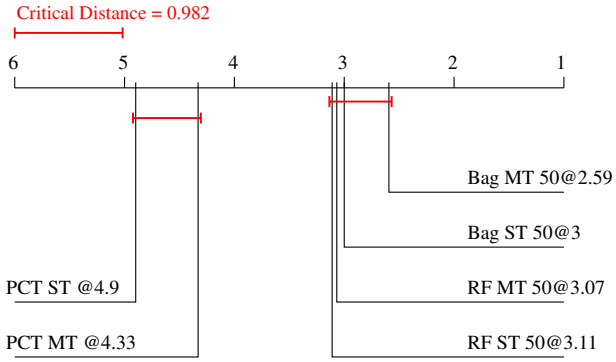
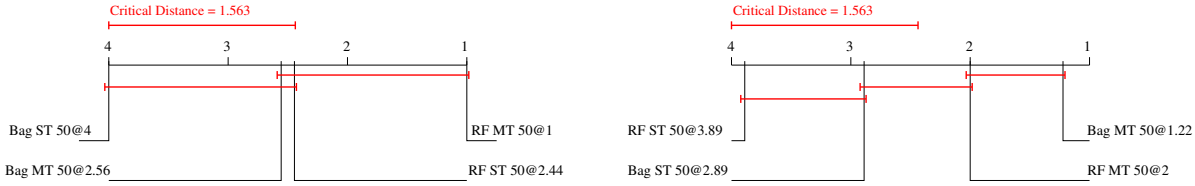


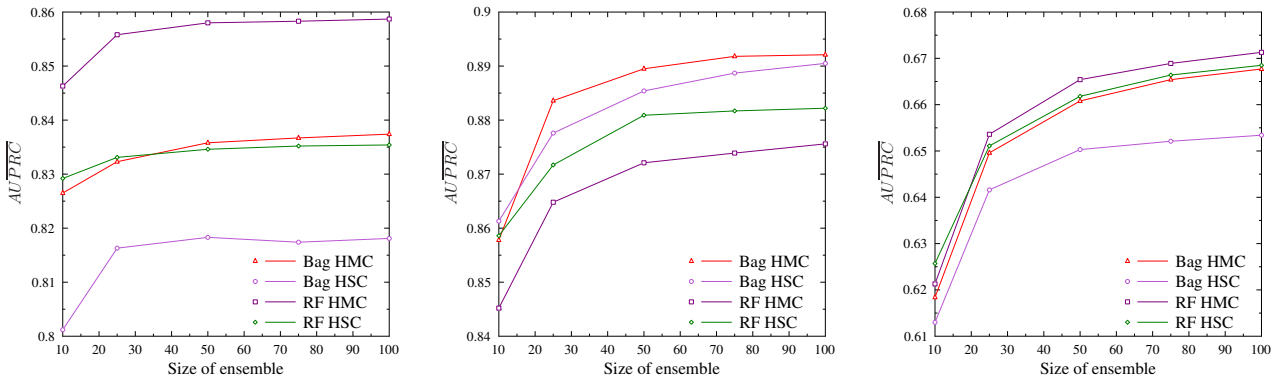
Figure 7: Average ranks diagrams (with the critical distance at significance level of 0.05) for prediction of multiple discrete targets. The differences in performance of the algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 6, with the addition of single predictive clustering tree - *PCT*.



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

Figure 8: Efficiency of the ensembles for predicting multiple discrete targets. The size of the ensembles is 50 trees.

not statistically significantly improve the ensemble's performance. Both types of ensemble methods for HMC and random forests for HSC saturate after 50 trees are added, while bagging for HSC saturates after only 25 trees. We further compare the performance of the ensembles at 50 trees (Figure 10).



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

Figure 9: Saturation curves for hierarchical multi–label classification. The curves (a) and (b) are obtained by giving the $AU\overline{PRC}$ value for a dataset, while the curve (c) by averaging the $AU\overline{PRC}$ values over all of the datasets. Higher $AU\overline{PRC}$ values mean better predictive performance. Note that the scale 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 average ranks diagram for the ensembles with 50 trees (Figure 10) shows that the performance of the different ensembles is not statistically significantly different. Note that the best performing method is random forests for HSC

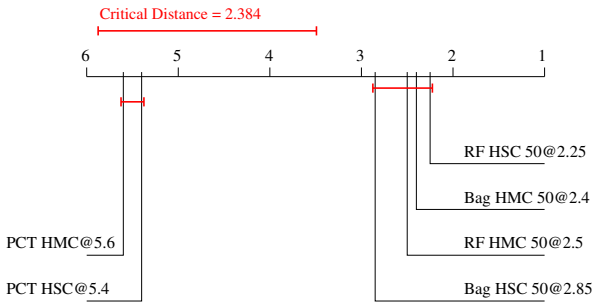
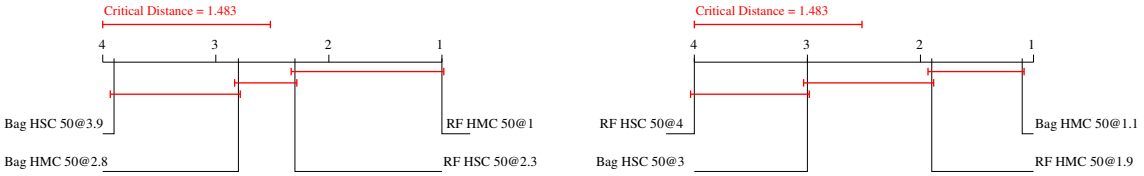


Figure 10: Average ranks diagrams (with the critical distance at a significance level of 0.05) for hierarchical multi–label classification. The differences in performance of algorithms connected with a red line are not statistically significant. The number after the name of an algorithm indicates its average rank. The abbreviations are the same as in Figure 9 with the addition of single predictive clustering tree - *PCT*.

(average rank 2.25) and the worst performing method is bagging for HSC (average rank 2.85). Ensembles for HMC and HSC are statistically significantly better than single PCT for HMC and HSC. However, the overall saturation curve from Figure 9(c), indicates that random forests for HMC is the best performing method. Moreover, ensembles for HMC perform better than ensembles for HSC on the datasets with larger hierarchies (i.e., datasets with ($|\mathcal{H}| > 300$) and in our case the datasets from functional genomics).

Finally, we compare the algorithms by their efficiency when they contain 50 trees (running times in Figure 11(a) and model size in Figure 11(b)). The random forests for HMC are statistically significantly faster than bagging for both HMC and HSC, while random forests for HSC are significantly faster than bagging for HSC. The models of bagging of HMC are statistically significantly smaller than the models from both types of ensembles for HSC, while the random forests for HMC are statistically significantly smaller than the random forests for HSC.



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

Figure 11: Efficiency of the ensembles for hierarchical multi–label classifications. The size of the ensembles is 50 trees.

5.4. Summary of the results

To summarize the results from the experiments, we discuss some general conclusions by formulating an answer to each of the questions from Section 4.1.

5.4.1. Predictive performance

The ensembles of PCTs for predicting structured outputs lift the predictive performance of a single PCT for predicting structured outputs. The difference in performance is statistically significant at the significance level of 0.05. This was previously shown only on applications where the target is a single continuous or discrete variable [12]. This finding is valid for all three machine learning tasks that we consider in this article.

The differences in predictive performance between ensembles of PCTs and ensembles of trees predicting components of the output are not statistically significant (at the 0.05 level) in any task. Also, none of the four considered ensemble algorithms is consistently performing best. 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.

5.4.2. Convergence

We looked at the saturation point of the ensembles' performances with respect to the number of base classifiers. In the majority of the cases, the predictive performance of the ensembles saturates after the 50th tree is added in the ensemble. Exceptions to this are bagging for HSC that saturates when 25 trees are added and random forests and bagging for predicting multiple discrete targets separately that saturate after 75 and 250 trees, respectively. Furthermore, the saturation curves offer some insight about the application of a given method on a group of datasets (summarized by their number of examples and number of descriptive variables). Also, the curves show that on the majority of the datasets, the ensembles of PCTs for predicting the structured outputs as a whole have better performance than the ensembles that predict the components. This is especially the case when the ensembles contain fewer trees.

5.4.3. Efficiency

With respect to model size and induction times, the tree ensembles that exploit the structure do have a significant advantage. The advantage is more pronounced when the datasets have large numbers of instances and/or descriptive attributes. Moreover, because of the feature sampling, the random forests for predicting structured outputs benefit even more, in terms of induction time, when the datasets have many descriptive attributes.

By averaging over all datasets considered in this study, we find that random forests for predicting the complete structured outputs are 4 times faster to construct and the models are 3.4 times smaller than a collection of random forests for predicting single targets or labels. In addition, they are 5.5 times faster to construct and have models with similar size as bagging for predicting the complete structured output. Furthermore, the latter is 3.9 times faster and yields models that are 2.9 times smaller than a collection of bagged trees for predicting single targets or labels.

6. Related work

The task of predicting structured outputs is gaining more and more attention within the machine learning research community [4, 3]. The community has proposed a number of different methods for addressing this task. However, they are typically "computationally demanding and ill-suited for dealing with large datasets" [4]. In this article, we have proposed a global method for predicting structured outputs that has good predictive performance and is very efficient: it scales linearly with the size of the output. We used the predictive clustering framework both for predicting multiple targets and for hierarchical multi-label classification. In the literature, individual methods typically solve only one task: only predicting multiple discrete variables (multi-target classification), only predicting multiple continuous variables (multi-target regression), or hierarchical multi-label classification only. In the remainder of this section, we first present the methods that predict multiple targets and then the methods for hierarchical multi-label classification.

6.1. Methods for multi-target prediction

The task of predicting multiple targets is connected with *multi-task learning* [60] and *learning to learn* [61] paradigms. These paradigms include the task of predicting a variable (continuous or discrete) using multiple input spaces (i.e., biological data for a disease obtained using different technologies); predicting multiple variables from multiple input spaces, and predicting multiple variables from a single input space. Here, we consider the last task. 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 [62, 63, 64, 65] 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. [66] assume that all related tasks have some common hidden structure. Greene [67] models the relatedness under the assumption of correlation between the noise for the different regression estimates. Baxter [62] 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. Furthermore, Chang et al. [68] adaptively learn the non-parametric common structure of the multiple tasks. Namely, the proposed algorithm iteratively discovers super-features effective for all the tasks. The estimation of the regression function for each task is then learned as a linear combination of these super-features.

Similarly, Hernández-Lobato et al. [69] use (efficiently parametrized) expectation propagation to approximate the posterior distribution of the model to identify relevant features for predicting all the tasks.

We present and categorize the related work in four groups: statistics, statistical learning theory, Bayesian theory and kernel learning. In statistics, Brown and Zidek [70] extend the standard ridge regression to multivariate ridge regression, while Breiman and Friedman [71] propose the CURDS&WHEY method, where the relations between the task are modelled 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 [62] and Ben-David and Borbely [65].

Most of the work in multi-task learning is done using Bayesian theory [61, 72, 73]. In this case, simultaneously with the parameters of the models for each of the tasks, a probabilistic model that captures the relations between the various tasks is being calculated. Most of these approaches use hierarchical Bayesian models.

Finally, there are many approaches for multi-task learning using kernel methods. For example, Evgeniou et al. [63] extend the kernel methods to the case of multi-task learning by using a particular type of kernel (multi-task kernel). The regularized multi-task learning then becomes equivalent to single-task learning when such a kernel is used. They show experimentally that the support vector machines (SVMs) with multi-task kernels have significantly better performance than the ones with single-task kernels. Liu et al. [74] proposed an approach to define the loss functions on the output manifold by considering it as a Riemannian submanifold in order to include its geometric structure in the learning (regression) process. The proposed approach can be used in the context of any regression algorithm and the experimental evaluation using regression SVMs provided satisfactory results. For more details on kernel methods and SVMs for multi-task learning, we refer the reader to [64, 75, 76, 77] and the references therein.

6.2. Methods for hierarchical multi-label classification

A number of approaches have been proposed for the task of hierarchical multi-label classification [4]. Silla and Freitas [3] survey and categorize the HMC methods based on some of their characteristics and their 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 and text categorization.

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 based methods and decision tree based methods.

Network based methods. The network based approaches predict functions of unannotated genes based on known functions of genes that are nearby in a functional association network or protein-protein interaction network [78]. Since the network based approaches are based on label propagation, a number of approaches were proposed to combine predictions of functional networks with those of a predictive model. Tian et al. [79], for instance, use logistic regression to combine predictions made by a functional association network with predictions from a random forest.

Kernel based methods. Obozinski et al. [80] present a two-step approach in which SVMs are first learned independently for each class separately (allowing violations of the hierarchy constraint) and are then reconciliated to enforce the hierarchy constraint. Similarly, Barutcuoglu et al. [54] 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. [81] extend the method by Barutcuoglu et al.[54] to an ensemble framework. Valentini and Re [82] 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. Next, Díez et al. [83] proposed a semi-dependent decomposition approach in which the node classifiers (binary SVMs) are constructed considering the other classifiers, their descendants and the loss function used to estimate the performance of the hierarchical classifiers. This approach follows a bottom-up strategy with the aim to optimize the loss function at every subtree assuming that all classifiers are known, except the one at the root of the hierarchy.

Rousu et al. [53] present a more direct method that does not require a second step to make sure that the hierarchy constraint is satisfied. Their approach is based on a large margin method for structured output prediction which defines a joint feature map over the input and the output space. Next, it applies 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. [53] 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 [34] 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. Clare [24] adapts the well-known decision tree algorithm C4.5 [23] 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. [84] present a decision tree based approach related to predictive clustering trees. They start from a different definition of variance and then kernelize this variance function. The result is a decision tree induction system that can be applied to structured output prediction using a method similar to the large margin methods mentioned above. Therefore, this system could also be used for HMC after defining a suitable kernel. To this end, an approach similar to that of Rousu et al. [53] could be used.

The present work follows Blockeel et al. [85, 14], who proposed the idea of using predictive clustering trees [5] for HMC tasks (PCTs for HMC). Their work [14] presents the first thorough empirical comparison between an HMC decision tree method in the context of tree shaped class hierarchies. Vens et al. [8] extend the algorithm towards hierarchies structured as directed acyclic graphs (DAGs) and show that learning one decision tree for predicting all classes simultaneously outperforms learning one tree per class (even if those trees are built by taking into account the hierarchy, via so-called hierarchical single-label classification - HSC).

7. Conclusions

In this article, we address the task of learning predictive models for structured output learning, which takes as input a tuple of attribute values and produces a structured object. In contrast to standard classification and regression, where the output is a single scalar value, in structured output learning 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 former 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 article to build ensemble models consisting of predictive clustering trees, which generalize classification trees. We use them for predicting different types of structured outputs, both locally and globally.

More specifically, we develop methods for learning 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 modelling tasks, i.e., multi-target regression, multi-target classification, and hierarchical multi-label classification. The different types of ensembles include bagging and random forests. 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 across a range of benchmark datasets for each of the three types of structured outputs. We compare tree 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 tree ensembles perform better than the single model counterparts in terms of predictive power. Global and local tree 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 analyse the computational complexity of the methods theoretically. The theoretical analyses are consistent with the empirical evidence, showing that the global tree ensembles are most efficient, especially random forests. The analyses also indicate that the proposed approaches are scalable to large datasets, which can be large along any of the following dimensions: number of attributes, number of examples, and size of the target.

Several directions for further work deserve a mention. The presented methods for learning ensembles can be extended to other types of structured outputs (e.g., time series or 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 use global random forests for obtaining feature ranking for structured outputs. To produce a feature ranking for a structured output, a feature ranking is typically performed first for each component of the output separately: these rankings are then merged using some aggregation function. The feature ranking based on global random forests can exploit the underlying dependencies and relations that may exist between the components of the outputs. Thus, it will provide feature rankings that are more relevant for the complete output rather than a component of it. Furthermore, it will be easily extended to a generic type of structured output.

Acknowledgment

We would like to thank Valentin Gjorgjioski for the discussions concerning the computational complexity of the proposed methods. Celine Vens is a postdoctoral fellow of the Research Fund – Flanders (FWO–Vlaanderen). The work of Dragi Kocev and Sašo Džeroski was supported by the Slovenian Research Agency (Grants P2-0103 and J2-2285), the European Commission (Grants ICT-2010-266722 and ICT-2011-287713), and Operation no. OP13.1.1.2.02.0005 nanced by the European Regional Development Fund (85%) and the Ministry of Education, Science, Culture and Sport of Slovenia (15%).

References

[1] Q. Yang, X. Wu, 10 challenging problems in data mining research, International Journal of Information Technology & Decision Making 5 (4) (2006) 597–604.
[2] H.-P. Kriegel, K. Borgwardt, P. Kröger, A. Pryakhin, M. Schubert, A. Zimek, Future trends in data mining, Data Mining and Knowledge Discovery 15 (2007) 87–97.
[3] C. Silla, A. Freitas, A survey of hierarchical classification across different application domains, Data Mining and Knowledge Discovery 22 (1-2) (2011) 31–72.
[4] G. H. Bakır, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, S. V. N. Vishwanathan, Predicting structured data, Neural Information Processing, The MIT Press, 2007.
[5] H. Blockeel, L. D. Raedt, J. Ramon, Top-down induction of clustering trees, in: Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann, 1998, pp. 55–63.
[6] J. Struyf, S. Džeroski, Constraint based induction of multi-objective regression trees, in: Proc. of the 4th International Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 3933, Springer, 2006, pp. 222–233.
[7] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Ensembles of multi–objective decision trees, in: ECML '07: Proceedings of the 18th European Conference on Machine Learning – LNCS 4701, Springer, 2007, pp. 624–631.
[8] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Machine Learning 73 (2) (2008) 185–214.
[9] 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.
[10] L. Breiman, J. Friedman, R. Olshen, C. J. Stone, Classification and Regression Trees, Chapman & Hall/CRC, 1984.
[11] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[12] G. Seni, J. F. Elder, Ensemble methods in data mining: Improving accuracy through combining predictions, Morgan & Claypool Publishers, 2010.
[13] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine Learning 36 (1) (1999) 105–139.
[14] 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, 2006, pp. 18–29.
[15] L. K. Hansen, P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (10) (1990) 993–1001.
[16] A. Verikas, A. Gelzinis, M. Bacauskiene, Mining data with random forests: A survey and results of new tests, Pattern Recognition 44 (2) (2011) 330–349.
[17] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, S. Džeroski, Predicting gene function using hierarchical multi-label decision tree ensembles, BMC Bioinformatics 11 (2) (2010) 1–14.
[18] S. Džeroski, V. Gjorgjioski, I. Slavkov, J. Struyf, Analysis of time series data with predictive clustering trees, in: Knowledge Discovery in Inductive Databases, 5th International Workshop, KDID 2006, Revised Selected and Invited Papers - LNCS 4747, Springer, 2007, pp. 63–80.
[19] D. Demšar, S. Džeroski, T. Larsen, J. Struyf, J. Axelsen, M. Bruns-Pedersen, P. H. Krogh, Using multi-objective classification to model communities of soil, Ecological Modelling 191 (1) (2006) 131–143.
[20] G. Tsoumakas, I. Katakis, Multi label classification: An overview, International Journal of Data Warehouse and Mining 3 (3) (2007) 1–13.

[21] P. Langley, Elements of machine learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
[22] H. Blockeel, J. Struyf, Efficient algorithms for decision tree cross-validation, Journal of Machine Learning Research 3 (2002) 621–650.
[23] R. J. Quinlan, C4.5: Programs for Machine Learning, 1st Edition, Morgan Kaufmann, 1993.
[24] A. Clare, Machine learning and data mining for yeast functional genomics, Ph.D. thesis, University of Wales Aberystwyth, Aberystwyth, Wales, UK (2003).
[25] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.
[26] H. Zouari, L. Heutte, Y. Lecourtier, Controlling the diversity in classifier ensembles through a measure of agreement, Pattern Recognition 38 (11) (2005) 2195–2199.
[27] Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm, in: Proc. of the Thirteenth International Conference on Machine Learning - ICML, Morgan Kaufman, 1996, pp. 148–156.
[28] L. Breiman, Using iterated bagging to debias regressions, Machine Learning 45 (3) (2001) 261–277.
[29] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[30] T. K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[31] T. G. Dietterich, Ensemble methods in machine learning, in: Proc. of the 1st International Workshop on Multiple Classifier Systems - LNCS 1857, Springer, 2000, pp. 1–15.
[32] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2005.
[33] Intel, Intel® SSE4 Programming Reference, D91561-003 Edition (2007).
[34] T. Gärtner, S. Vembu, On structured output training: hard cases and an efficient alternative, Machine Learning 76 (2009) 227–242.
[35] C. Kampichler, S. Džeroski, R. Wieland, 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) (2000) 197–209.
[36] A. Karalič, First order regression, Ph.D. thesis, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia (1995).
[37] D. Stojanova, P. Panov, V. Gjorgjioski, A. Kobler, S. Džeroski, Estimating vegetation height and canopy cover from remotely sensed data with machine learning, Ecological Informatics 5 (4) (2010) 256–266.
[38] D. Stojanova, Estimating forest properties from remotely sensed data by using machine learning, Master's thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia (2009).
[39] D. Demšar, M. Debeljak, S. Džeroski, C. Lavigne, Modelling pollen dispersal of genetically modified oilseed rape within the field, in: The Annual Meeting of the Ecological Society of America, 2005.
[40] A. Asuncion, D. Newman, UCI - Machine Learning Repository, http://www.ics.uci.edu/ mlearn/MLRepository.html (2007).
URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`
[41] V. Gjorgjioski, S. Džeroski, M. White, Clustering analysis of vegetation data, Tech. Rep. 10065, Jožef Stefan Institute (2008).
[42] D. Kocev, S. Džeroski, M. White, G. Newell, P. Griffioen, Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition, Ecological Modelling 220 (8) (2009) 1159–1168.
[43] H. Blockeel, S. Džeroski, J. Grbović, Simultaneous prediction of multiple chemical parameters of river water quality with TILDE, in: Proceedings of the 3rd European Conference on PKDD - LNAI 1704, Springer, 1999, pp. 32–40.
[44] S. Džeroski, D. Demšar, J. Grbović, Predicting chemical parameters of river water quality from bioindicator data, Applied Intelligence 13 (1) (2000) 7–17.
[45] K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas, Multilabel classification of music into emotions, in: Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), 2008, pp. 325–330.
[46] M. Skrjanc, M. Grobelnik, D. Zupanic, Insights offered by data-mining when analyzing media space data, Informatica (Slovenia) 25 (3) (2001) 357–363.
[47] M. Boutell, J. Luo, X. Shen, C. Brown, Learning multi-label scene classification, Pattern Recognition 37 (9) (2004) 1757–1771.
[48] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, in: In Advances in Neural Information Processing Systems 14, MIT Press, 2001, pp. 681–687.
[49] I. Dimitrovski, D. Kocev, S. Loskovska, S. Džeroski, Hierchical annotation of medical images, in: Proceedings of the 11th International Multiconference - Information Society IS 2008, IJS, Ljubljana, 2008, pp. 174–181.
[50] ADIAC, Automatic diatom identification and classification, http://rbg-web2.rbge.org.uk/ADIAC/ (2008).
[51] B. Klimt, Y. Yang, The enron corpus: A new dataset for email classification research, in: ECML '04: Proceedings of the 18th European Conference on Machine Learning – LNCS 3201, Springer, 2004, pp. 217–226.
[52] D. D. Lewis, Y. Yang, T. G. Rose, F. Li, RCV1: A new benchmark collection for text categorization research, Journal of Machine Learning Research 5 (2004) 361–397.
[53] J. Rousu, C. Saunders, S. Szedmak, J. Shawe-Taylor, Kernel–based learning of hierarchical multilabel classification models, Journal of Machine Learning Research 7 (2006) 1601–1626.
[54] Z. Barutcuoglu, R. E. Schapire, O. G. Troyanskaya, Hierarchical multi-label prediction of gene function, Bioinformatics 22 (7) (2006) 830–836.
[55] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Information Processing & Management 45 (4) (2009) 427–437.
[56] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[57] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Annals of Mathematical Statistics 11 (1940) 86–92.
[58] R. L. Iman, J. M. Davenport, Approximations of the critical region of the Friedman statistic, Communications in Statistics - Theory and Methods 9 (6) (1980) 571–595.
[59] P. B. Nemenyi, Distribution-free multiple comparisons, Ph.D. thesis, Princeton University, Princeton, NY, USA (1963).
[60] R. Caruana, Multitask learning, Machine Learning 28 (1997) 41–75.
[61] S. Thrun, L. Pratt, Learning to learn, Kluwer Academic Publishers, 1998.

[62] J. Baxter, A model of inductive bias learning, Journal of Artificial Intelligence Research 12 (2000) 149–198.
[63] T. Evgeniou, C. A. Micchelli, M. Pontil, Learning multiple tasks with kernel methods, Journal of Machine Learning Research 6 (2005) 615–637.
[64] A. Caponnetto, C. A. Micchelli, M. Pontil, Y. Ying, Universal multi–task kernels, Journal of Machine Learning Research 9 (2008) 1615–1646.
[65] S. Ben-David, R. S. Borbely, A notion of task relatedness yielding provable multiple-task learning guarantees, Machine Learning 73 (3) (2008) 273–287.
[66] R. K. Ando, T. Zhang, P. Bartlett, A framework for learning predictive structures from multiple tasks and unlabeled data, Journal of Machine Learning Research 6 (2005) 1817–1853.
[67] W. H. Greene, Econometric analysis, 6th Edition, Prentice Hall, 2007.
[68] Y. Chang, J. Bai, K. Zhou, G.-R. Xue, H. Zha, Z. Zheng, Multi-task learning to rank for web search, Pattern Recognition Letters 33 (2) (2012) 173–181.
[69] D. Hernández-Lobato, J. Hernández-Lobato, T. Helleputte, P. Dupont, Expectation propagation for bayesian multi-task feature selection, in: ECML '10: Proceedings of the 21st European Conference on Machine Learning – LNCS 6321, Springer, 2010, pp. 522–537.
[70] P. J. Brown, J. V. Zidek, Adaptive multivariate ridge regression, The Annals of Statistics 8 (1) (1980) 64–74.
[71] L. Breiman, J. Friedman, Predicting multivariate responses in multiple linear regression, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 59 (1) (1997) 3–54.
[72] B. Bakker, T. Heskes, Task clustering and gating for bayesian multitask learning, Journal of Machine Learning Research 4 (2003) 83–99.
[73] A. Wilson, A. Fern, S. Ray, P. Tadepalli, Multi-task reinforcement learning: a hierarchical Bayesian approach, in: ICML '07: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 1015–1022.
[74] G. Liu, Z. Lin, Y. Yu, Multi-output regression on the output manifold, Pattern Recognition 42 (11) (2009) 2737–2743.
[75] A. Argyriou, T. Evgeniou, M. Pontil, Convex multi-task feature learning, Machine Learning 73 (2008) 243–272.
[76] C. A. Micchelli, M. Pontil, Kernels for multi–task learning, in: Advances in Neural Information Processing Systems 17 - Proceedings of the 2004 Conference, 2004, pp. 921–928.
[77] F. Cai, V. Cherkassky, SVM+ regression and multi-task learning, in: International Joint Conference on Neural Networks (IJCNN), 2009, pp. 418–424.
[78] Y. Chen, D. Xu, Global protein function annotation through mining genome-scale data in yeast Saccharomyces cerevisiae, Nucleic Acids Research 32 (21) (2004) 6414–6424.
[79] W. Tian, L. V. Zhang, M. Taşan, F. D. Gibbons, O. D. King, J. Park, Z. Wunderlich, J. M. Cherry, F. P. Roth, Combining guilt–by–association and guilt–by–profiling to predict Saccharomyces cerevisiae gene function, Genome biology 9 (S1) (2008) S7+.
[80] G. Obozinski, G. Lanckriet, C. Grant, M. I. Jordan, W. S. Noble, Consistent probabilistic outputs for protein function prediction, Genome Biology 9 (S1) (2008) S6+.
[81] Y. Guan, C. L. Myers, D. C. Hess, Z. Barutcuoglu, A. A. Caudy, O. G. Troyanskaya, Predicting gene function in a hierarchical context with an ensemble of classifiers, Genome biology 9 (S1) (2008) S3+.
[82] G. Valentini, M. Re, 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, 2009, pp. 133–146.
[83] J. Díez, J. J. del Coz, A. Bahamonde, A semi-dependent decomposition approach to learn hierarchical classifiers, Pattern Recognition 43 (11) (2010) 3795–3804.
[84] P. Geurts, L. Wehenkel, F. D'Alché-Buc, Kernelizing the output of tree–based methods, in: ICML '06: Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006, pp. 345–352.
[85] H. Blockeel, M. Bruynooghe, S. Džeroski, J. Ramon, J. Struyf, Hierarchical multi–classification, in: KDD-2002 Workshop Notes: MRDM 2002, Workshop on Multi-Relational Data Mining, 2002, pp. 21–35.

Appendix A. Performance measures for classification

Nomenclature:

- c_i - class value
- T - number of classes
- TP_i - true positives for class c_i
- FP_i - false positives for class c_i
- FN_i - false negatives for class c_i
- TN_i - true negatives for class c_i
- P_i - precision for class c_i
- R_i - recall for class c_i
- F - F-score
- MCC - Matthews Correlation Coefficient
- $BACC$ - Balanced Accuracy
- DP - Discriminant Power

Table A.6: Evaluation measures - general definitions.

Measure	Formula
Precision	$P = \frac{TP}{TP+FP}$
Recall	$R = \frac{TP}{TP+FN}$
F-score	$F = 2 \cdot \frac{P \cdot R}{P+R}$
MCC	$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}}$
BACC	$BACC = \frac{sensitivity + specificity}{2} = \frac{1}{2} \cdot \left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP} \right)$
DP	$DP = \frac{\sqrt{3}}{\pi} \cdot \ln \left(\frac{TP}{FP} \cdot \frac{TN}{FN} \right)$

Note: The formula for *discriminant power* is $DP = \frac{\sqrt{3}}{\pi} \cdot (\ln X + \ln Y)$, where $X = \frac{sensitivity}{1-sensitivity}$, $Y = \frac{specificity}{1-specificity}$, $sensitivity = \frac{TP}{TP+FN}$ and $specificity = \frac{TN}{TN+FP}$.

Table A.7: Micro averaged evaluation measures.

Measure	Formula
μPrecision	$\frac{\sum_i^T TP_i}{\sum_i^T TP_i+\sum_i^T FP_i}$
μRecall	$\frac{\sum_i^T TP_i}{\sum_i^T TP_i+\sum_i^T FN_i}$
μF-score	$2 \cdot \frac{P_i^\mu \cdot R_i^\mu}{P_i^\mu+R_i^\mu}$
μMCC	$\frac{\sum_i^T TP_i \cdot \sum_i^T TN_i - \sum_i^T FP_i \cdot \sum_i^T FN_i}{\sqrt{(\sum_i^T TP_i+\sum_i^T FP_i)\cdot(\sum_i^T TP_i+\sum_i^T FN_i)\cdot(\sum_i^T TN_i+\sum_i^T FP_i)\cdot(\sum_i^T TN_i+\sum_i^T FN_i)}}$
μBACC	$\frac{1}{2} \cdot \left(\frac{\sum_i^T TP_i}{\sum_i^T TP_i+\sum_i^T FN_i} + \frac{\sum_i^T TN_i}{\sum_i^T TN_i+\sum_i^T FP_i} \right)$
μDP	$\frac{\sqrt{3}}{\pi} \cdot \ln\left(\frac{\sum_i^T TP_i}{\sum_i^T FP_i} \cdot \frac{\sum_i^T TN_i}{\sum_i^T FN_i} \right)$

Table A.8: Macro averaged evaluation measures.

Measure	Formula
MPrecision	$\sum_i^T P_i$
MRecall	$\sum_i^T R_i$
MF-score	$\sum_i^T F_i$
MMCC	$\sum_i^T MCC_i$
MBACC	$\sum_i^T BACC_i$
MDP	$\sum_i^T DP_i$