

Ensembles for Predicting Structured Outputs

September 10, 2010

Dragi Kocev

DRAGI.KOCEV@IJS.SI

*Department of Knowledge Technologies, Jožef Stefan Institute
Jamova cesta 39, 1000 Ljubljana, Slovenia***Celine Vens**

CELINE.VENS@CS.KULEUVEN.BE

Jan Struyf

JAN.STRUYF@CS.KULEUVEN.BE

*Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium***Sašo Džeroski**

SASO.DZEROSKI@IJS.SI

*Department of Knowledge Technologies, Jožef Stefan Institute
Jamova cesta 39, 1000 Ljubljana, Slovenia***Editor:**

Abstract

This is not an abstract.

Keywords: ensembles, predictive clustering trees, structured outputs

1. Introduction

Supervised learning is one of the most widely researched and investigated areas of machine learning. The goal in the supervised learning is from a set of known examples to learn a function f that outputs a prediction for a previously unseen example. Most widely used tasks of supervised learning are binary classification, multi-class classification and regression. If the examples can belong to two classes (e.g., the example has some property or not) the task is then called binary classification. The task where the examples can belong to a single class from a given set of m -classes is known as multi-class classification. The case where the output is a real value, the task is then called regression.

However, many real-world problems have structured output (target or response) variables and require more complex approaches. Typically, the outputs are tuples of continuous or discrete variables, classes organized in hierarchies (trees or directed acyclic graphs), sequences etc. Here, we consider the first two types of structured outputs.

Real-world problems that have tuples of variables as target concepts include: prediction of vegetation condition (the condition is presented as an index of 7 continuous variables (Kocev et al., 2009)), prediction of forest stand height and canopy cover using GIS data ((Stojanova et al., 2010)), prediction of media behaviour (which journals/TV stations a user would prefer to read/watch or ignore (Skrjanc et al., 2001)) etc. Target concepts organized in hierarchies are encountered in the following problems (for more details see (Silla and Freitas, 2010)): assigning functions to a gene (functional genomics), text/web documents classification and categorization, genre classification, image classification/annotation etc.

There are two groups of approaches to solving the problem of predicting structured outputs (Bakır et al., 2007; Silla and Freitas, 2010): (1) algorithms that predict component(s) of the output and then combine the separate models to get the global model and (2) algorithms that predict the complete structure as a whole (also known as ‘big-bang’ approach). The advantage of the latter approach is that it can exploit and use the dependencies that exist between the components of the structured output in the model learning phase. Furthermore, they produce models that are typically smaller than the sum of the sizes of the models for the components.

In this paper, we advocate the ‘big-bang’ approach, in particular we propose to use the predictive clustering trees (PCT) framework. The PCTs are able to make predictions for several types of structured outputs: tuples of continuous/discrete variables, hierarchies of classes (organized into tree or DAG) and time series. More details about the PCT framework can be found in (Blockeel et al., 1998; Struyf and Džeroski, 2006; Kocev et al., 2007; Vens et al., 2008; Slavkov et al., 2010). Furthermore, we construct ensembles that use PCTs as base classifiers. In particular, we employ bagging (Breiman, 1996) and random forests (Breiman, 2001a) – the two most widely used ensemble learning techniques.

The ensemble methods are able to lift the predictive performance of the base classifier in the case of single continuous or discrete target variable (Breiman, 1996; Bauer and Kohavi, 1999; Breiman, 2001a). Here, we test whether the ensembles lift the predictive performance of a classifier when the target is a structure. We also compare the performance of ensembles that predict the structured output as a whole with the performance of ensembles that predict components of the structured output. The question we address is whether exploitation of the structure of the output can lift the predictive performance in the context of ensemble learning. Moreover, we compare the ensemble learning methods by their efficiency in terms of running time and size of models.

The remainder of this paper is organized as follows. In Section 2, the considered machine learning tasks are formally defined. Section 3 presents the related work and Section 4 explains the predictive clustering trees framework and the extensions for predicting multiple targets and hierarchical multi-label classification. The ensemble methods and their extension for predicting structured outputs are discussed in Section 5. The design of the experiments, the descriptions of the datasets, the evaluation measures and the parameter instantiations for the algorithms are given in Section 6. Section 7 presents and discusses the obtained results. Finally, the conclusions are stated in Section 8.

2. Machine learning tasks

Following the recommendations from Džeroski (2007), we formally describe the machine learning tasks that we consider here. In particular, this work is focused on multiple targets prediction and hierarchical classification.

2.1 Predicting multiple targets task

We define the task of multiple targets prediction¹ as follows:

Given:

- A description space X that consists of tuples of primitives (boolean, discrete or continuous variables), i.e. $\forall X_i \in X, X_i = (x_{i_1}, x_{i_2}, \dots, x_{i_D})$, where D is the size of the tuple (or number of descriptive variables);
- a target space Y , where each tuple consists of several variables that can be either continuous or discrete, i.e., $\forall Y_i \in Y, Y_i = (y_{i_1}, y_{i_2}, \dots, y_{i_T})$, where T is the size of the tuple (or number of target variables),
- a set E , where $E = \{(X_i, Y_i) | X_i \in X, Y_i \in Y, 1 \leq i \leq N\}$ and N is the number of examples of E ($N = |E|$), and
- a quality criterion q (which rewards models with high predictive accuracy and low complexity).

Find: a function $f : X \rightarrow Y$ such that f maximizes q . Here, the function f is represented with decision trees, i.e., predictive clustering trees.

If the tuples from Y (the target space) consist of continuous/numeric variables then the task at hand is multiple targets regression. Likewise, if the tuples from Y consist of discrete/nominal variables then the task is called multiple targets classification.

2.2 Hierarchical classification task

Classification is defined as the task of learning a model using a set of classified instances and applying the obtained model to a set of previously unseen examples. The unseen examples are classified into a single class from a set of possible classes. Hierarchical classification differs from the ‘traditional’ classification in the following: the classes are organized in hierarchy, so, an example that belongs to a given class it automatically belongs to all its superclasses (this is known as the ‘hierarchy constraint’). Furthermore, if an example can belong to multiple classes simultaneously, then this task is called hierarchical multi-label classification (Silla and Freitas, 2010; Vens et al., 2008).

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

Given:

- A description space X that consists of tuples of primitives (boolean, discrete or continuous variables), i.e. $\forall X_i \in X, X_i = (x_{i_1}, x_{i_2}, \dots, x_{i_D})$, where D is the size of a tuple (or number of descriptive variables),
- a target space S , defined with a class hierarchy (C, \leq_h) , where C is a set of classes and \leq_h is a partial order (structured as a rooted tree) representing the superclass relationship (for all $c_1, c_2 \in C : c_1 \leq_h c_2$ if and only if c_1 is a superclass of c_2),

1. Multiple targets prediction is previously referred to as multi-objective prediction in the literature (Struyf and Džeroski, 2006; Kocev et al., 2007).

- a set E , where $E = \{(X_i, S_i) | X_i \in X, S_i \subseteq C, c \in S_i \Rightarrow \forall c' \leq_h c : c' \in S_i, 1 \leq i \leq N\}$ and N is the number of examples of E ($N = |E|$), and
- a quality criterion q (which rewards models with high predictive accuracy and low complexity).

Find: a function $f : X \rightarrow 2^C$ (where 2^C is the power set of C) such that f maximizes q and $c \in f(x) \Rightarrow \forall c' \leq_h c : c' \in f(x)$. The last condition is called the ‘hierarchy constraint’. Here, the function f is represented with decision trees, i.e., predictive clustering trees.

3. Related work

In this section, we shortly present some approaches that were used in the context of predicting multiple targets and hierarchical multi-label classification. To the best of the knowledge of the authors, there are no previous attempts of treating these tasks jointly.

3.1 Prediction of multiple targets

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

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

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

We present and categorize the related work along four dimensions: statistics, statistical learning theory, Bayesian theory and kernel learning. To begin with, in statistics, Brown and Zidek (1980) extend the standard ridge regression to multivariate ridge regression and Breiman and Friedman (1997) propose the curds&whey method, where the relations between the task are modeled in a post-processing phase. In statistical learning theory, for handling the multiple tasks, an extension of the VC-dimension and the basic generalization

bounds for single task learning are proposed by Baxter (2000); Ben-David and Borbely (2008).

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

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

3.2 Hierarchical multi-label classification

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

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

The network based approaches predict functions of unannotated genes based on known functions of genes that are nearby in a functional association network or protein-protein interaction network (Chen and Xu, 2004). Mostafavi et al. (2008) calculate per gene function a composite functional association network from multiple networks derived from different genomic and proteomic data sources. Since the network base approaches are based on label propagation, a number of approaches were proposed to combine predictions of functional networks with those of a predictive model. Tian et al. (2008), for instance, use logistic regression to combine predictions made by a functional association network with predictions from a random forest.

Lee et al. (2006) combine Markov random fields and support vector machines which are generated for each class separately. They compute diffusion kernels and use them in kernel logistic regression. Obozinski et al. (2008) present a two-step approach in which SVMs are first learned independently for each class separately (allowing violations of the hierarchy constraint) and are then reconciliated to enforce the hierarchy constraint. Simi-

larly, Barutcuoglu et al. (2006) use unthresholded SVMs learned for each class separately and then the SVMs are combined using a Bayesian network so that the predictions are consistent with the hierarchical relationships. Guan et al. (2008) extend this method to an ensemble framework. Valentini and Re (2009) also propose a hierarchical ensemble method that uses probabilistic SVMs as base learners and combines the predictions by propagating the weighted true path rule both top-down and bottom-up through the hierarchy, which ensures consistency with the hierarchy constraint.

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

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

Clare (2003) adapts the well-known decision tree algorithm C4.5 (Quinlan, 1993) to cope with the issues introduced by the HMC task. Her 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. (2006) presented a decision tree based approach related to predictive clustering trees. They start from a different definition of variance and then kernelize this variance function. The result is a decision tree induction system that can be applied to structured output prediction using a method similar to the large margin methods mentioned above. Therefore, this system could also be used for HMC after defining a suitable kernel. To this end, an approach similar to that of Rousu et al. (2006) could be used.

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

4. Predictive clustering trees

The Predictive Clustering Trees (PCTs) framework sees a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively

partitioned into smaller clusters while moving down the tree. The PCT framework is implemented in the CLUS system (Blockeel and Struyf, 2002).¹

PCTs can be induced with a standard “top-down induction of decision trees” (TDIDT) algorithm (Breiman et al., 1984). The algorithm is presented in Table 1. The algorithm takes as input a set of examples (E) and outputs a tree. The heuristic (h) that is used for selecting the tests (t) is the reduction in variance caused by partitioning (\mathcal{P}) the instances (see line 4 of BestTest procedure in Table 1). Maximizing the variance reduction maximizes cluster homogeneity and improves predictive performance.

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

procedure PCT(E) returns tree	procedure BestTest(E)
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$	1: $(t^*, h^*, \mathcal{P}^*) = (\text{none}, 0, \emptyset)$
2: if $t^* \neq \text{none}$ then	2: for each possible test t do
3: for each $E_k \in \mathcal{P}^*$ do	3: $\mathcal{P} =$ partition induced by t on E
4: $\text{tree}_k = \text{PCT}(E_k)$	4: $h = \text{Var}(E) - \sum_{E_k \in \mathcal{P}} \frac{ E_k }{ E } \text{Var}(E_k)$
5: return $\text{node}(t^*, \bigcup_k \{\text{tree}_k\})$	5: if $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ then
6: else	6: $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $\text{leaf}(\text{Prototype}(E))$	7: return $(t^*, h^*, \mathcal{P}^*)$

The main difference between the algorithm for learning PCTs and a standard decision tree learner (for example, see the C4.5 algorithm proposed by Quinlan (1993)) is that the former considers the variance function and the prototype function, that computes a label for each leaf, as parameters that can be instantiated for a given learning task. So far, the PCTs were extended for the following tasks: multiple targets prediction (Struyf and Džeroski, 2006; Kocev et al., 2007), hierarchical-multi label classification (Vens et al., 2008) and prediction of time-series (Slavkov et al., 2010). In this paper, we focus on the first two tasks.

4.1 PCTs for multiple targets

The PCTs that are able to predict multiple targets simultaneously are called multiple targets decision trees (MTDTs). The MTDTs that predict tuple of discrete variables are called multiple targets classification trees (MTCTs), while the MTDTs that predict tuple of continuous variables (regression tasks) are called multiple targets regression trees (MTRTs). An example of predictive clustering tree for predicting multiple continuous targets is shown in Figure 1. The internal nodes of the tree contain tests on the descriptive variables (in this case, some GIS data) and the leafs store the predictions (in this case, the index of the condition of the vegetation).

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

1. CLUS system is available for download at <http://www.cs.kuleuven.be/~dtai/clus>

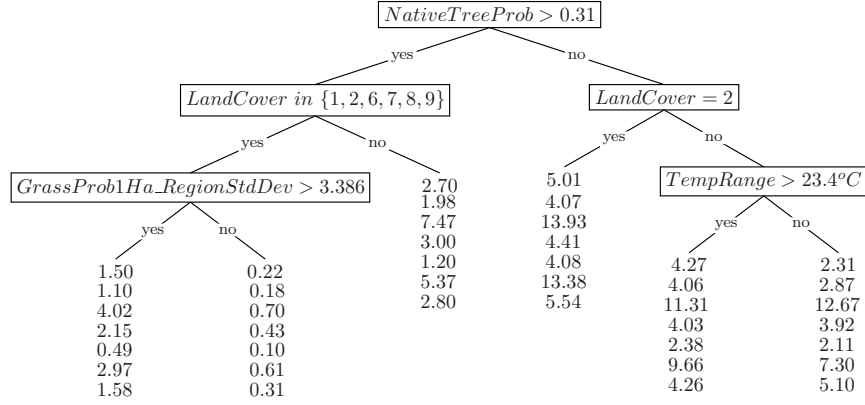


Figure 1: Example of a predictive clustering tree for multiple continuous targets taken from (Kocev et al., 2009). Each of the leafs stores the predictions for the indices of the vegetation quality.

The variance function for classification trees is computed as the sum of the gini indexes of the target variables, i.e., $Var(E) = \sum_{i=1}^T Gini(E, Y_i)$. Furthermore, one can use the sum of the entropies of class variables as variance function, i.e., $Var(E) = \sum_{i=1}^T Entropy(E, Y_i)$ (this definition has also been used in the context of multi-label prediction (Clare, 2003)). The prototype function returns a vector of probabilities that an instance belongs to a given class value for each target variable. Using this probability, the majority class for each target attribute can be calculated. In addition to the two aforementioned instantiations of the variance function for classification problems, the CLUS system also implements other variance functions, such as reduced error, information gain, gain ratio and m -estimate.

4.2 PCTs for hierarchical classification

Silla and Freitas (2010) describe the algorithms for hierarchical classification as 4-tuple $\langle \Delta, \Sigma, \Omega, \Theta \rangle$. In this 4-tuple, Δ indicates whether the algorithm makes prediction 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 CLUS-HMC algorithm can be described as follows:

- $\Delta = MPP$ (multiple path prediction): the algorithm can assign multiple paths or predicted classes to each instance.
- $\Sigma = NMLNP$ (non-mandatory leaf-node prediction): an instance can be labeled with a label at any level of the taxonomy.
- $\Omega = T \text{ OR } DAG$ (Tree or Directed Acyclic Graph): the algorithm can handle both tree-shaped or DAG hierarchies of classes.
- $\Theta = GC$ (global classifier): the algorithm constructs a single model valid for all classes.

Hence, the CLUS-HMC algorithm belongs to the group of approaches known as ‘big-bang’ or global classifiers (Silla and Freitas, 2010). The global classifier has two main advantages over the local classifiers (i.e., classifiers per node in the hierarchy, per parent node or per level): (1) the total size of the global model is considerably smaller than the total size of all local classifiers and (2) the dependencies between the classes can be taken into account while learning the classifier and these dependencies can be explicated (Blockeel et al., 2002; Vens et al., 2008).

In this work, we also consider an approach that constructs a classifier for each edge in the hierarchy (we refer to it as hierarchical single-label classification – HSLC). Here, we construct a tree for each class c using the instances that belong to the parent class of c (denoted as $par(c)$). Construction of this type of tree requires less instances: only the instances that are labeled with the $par(c)$. Thus, the instances labeled with class c become the positive instances, while the other instances (ones that are labeled with the sibling classes of c) become negative.

The resulting HSLC tree predicts the conditional probability $P(c|par(c))$. A new instance is predicted by recursive application of the product rule $P(c) = P(c|par(c)) \cdot P(par(c))$ starting from the tree for a top-level class. Additionally, the probabilities are thresholded to obtain the set of predicted classes. To satisfy the hierarchy constraint, the threshold τ should be chosen in the following way (using the annotation from above): $\tau_i \leq \tau_j$ whenever $c_i \leq_h c_j$. For a detailed description of the tasks of HMLC and HSLC see (Vens et al., 2008).

In the remaining of this section, we first describe the instantiation of CLUS-HMC that can predict tree-shaped hierarchies and afterwards the instantiation for predicting DAG hierarchies.

4.2.1 TREE-SHAPED HIERARCHIES

In CLUS-HMC, a hierarchy is represented as a vector with binary components. If an example belongs to class c_i then the i ’th component of the vector is set to 1 and to 0 otherwise (see Fig. 2(b)). The arithmetic mean of a set of such vectors contains as i ’th component the proportion of examples of the set belonging to class c_i . The variance of a set of examples E is defined as the average squared distance between each example’s class vector (L_k) and the set’s mean class vector (\bar{L}), i.e.,

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

In the HMLC context, the similarity at higher levels of the hierarchy is more important than the similarity at lower levels. To that aim, the distance measure used in the above formula is a weighted Euclidean distance:

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

where $L_{k,i}$ is the i ’th component of the class vector L_k of an instance X_k , and the class weights $w(c)$ decrease with the depth of the class in the hierarchy (e.g., $w(c) = w_0^{depth(c)}$, $0 < w_0 < 1$). For example, let’s consider the toy class hierarchy shown in Fig.2(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 Fig.2(b)) and $S_2 = \{c_2\}$, respectively. Using a vector representation with consecutive components representing membership of class $c_1, c_2, c_{2.1}, c_{2.2}$ and c_3 , in that order (preorder traversal of the tree), the distance is calculated as follows:

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

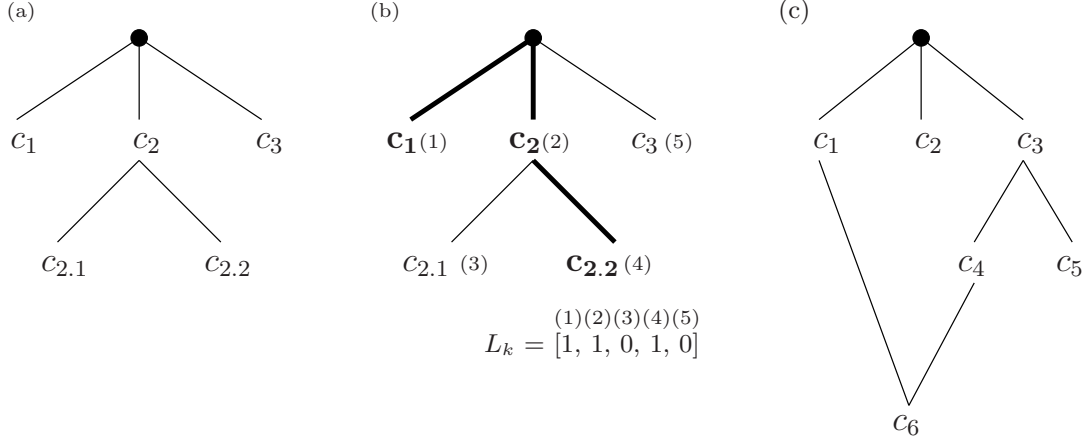


Figure 2: Toy examples of hierarchies structured as tree and DAG. (a) Class label names contain information about the position in the hierarchy, e.g., $c_{2.1}$ is a subclass of c_2 . (b) The set of classes $\{c_1, c_2, c_{2.2}\}$, indicated in bold in the hierarchy, and represented as a vector. (c) A class hierarchy structured as a DAG. The class c_6 has two parents: c_1 and c_4 .

A classification tree stores in a leaf the majority class, which will be the tree’s prediction for all examples that will arrive in the leaf. In the case of HMLC, 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 prediction. Note that the value for the i -th component of \bar{L} can be interpreted as the probability that an example arriving at the given leaf belongs to class c_i .

The prediction for an example that arrives in the leaf can be obtained by applying a user defined threshold τ on the probability; if i -th component of \bar{L} is above τ then the examples belong to the class c_i . When a PCT is making a prediction it preserves the hierarchy constraint (the predictions comply to the parent child relationships from the hierarchy) by choosing the value for the threshold τ as follows: $\tau_i \leq \tau_j$ whenever $c_i \leq_h c_j$. The threshold is selected depending on the context. The user may set the threshold such that the resulting classifier has high precision at the cost of lower recall or vice versa, to maximize F1-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 models.

4.2.2 DAG HIERARCHIES

In the previous subsection, we assumed that the class hierarchy is structured as a rooted tree. However, in many real life domains (especially in biology) the hierarchies are structured as directed acyclic graphs (DAGs). Most famous example of DAG hierarchy is the Gene Ontology (Ashburner et al., 2000), a biological classification hierarchy for genes. In general, a DAG hierarchy can have two interpretations: if an example belongs to a given class c , then it also belongs to all superclasses of c , or it belongs to at least one superclass of c . Here, we focus on the first case: the multiple inheritance interpretation.

The variance function used for tree-shaped hierarchies uses the weighted Euclidean distance between the class vectors, where the weight of a class depends on its depth in the hierarchy. When the hierarchy is a DAG, then the depth of a class is not unique: classes do not have single path from the top-node (for example see class c_6 in Fig. 2(c)). To resolve this issue, Vens et al. (2008) suggest four aggregation schemes of the possible paths from the top-node to a given class: average, maximum, minimum and sum. The aggregation schemes use the observation that $w(c) = w_0^{depth(c)}$ can be rewritten as the recursive relation $w(c) = w_0 \cdot w(par(c))$, with $par(c)$ as the parent class of c , and the weights of the top-level classes equal to w_0 . After an extensive experimental evaluation, Vens et al. (2008) recommend to use the average as aggregation function ($w(c) = w_0 \cdot \text{avg}_j\{w(par_j(c))\}$).

5. Ensemble methods for predicting structured outputs

An ensemble is a set of classifiers (called base classifiers) constructed with a given algorithm. Prediction for a new example is obtained by combining the predictions of all classifiers from the ensemble. The predictions from the classifiers can be combined by taking the average (for regression tasks) and the majority or probability distribution vote (for classification tasks), as described in (Bauer and Kohavi, 1999; Breiman, 1996), or by taking more complex aggregation schemes (Kuncheva, 2004).

To obtain a prediction from the ensemble for predicting structured outputs, we accordingly extend the voting schemes. For the datasets with multiple continuous targets, as prediction of the ensemble, we take average of the predictions of the base classifiers. Also, for the datasets for hierarchical classification we use the average of the predictions but additionally, we apply the thresholding described in Section 4.2. We obtain the ensemble predictions for the datasets with multiple discrete targets using probability distribution voting (as suggested by Bauer and Kohavi (1999)). We use predictive clustering trees as base classifiers for the ensembles for structured outputs (see line 4 from the *Induce_Forest* procedure in Table 2).

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

Table 2: Random forest induction algorithm, where E is the set of the training examples, k is the number of trees in the forest, and $f(D)$ is the size of the feature subset that is considered at each node during tree construction.

```

procedure Induce.Forest( $E, k, f(D)$ ) returns Forest
1:  $F = \emptyset$ 
2: for  $i = 1$  to  $k$  do
3:    $E_i = \text{sample\_with\_replacement}(E)$ 
4:    $T_i = PCT(E_i, f(D))$ 
5:    $F = F \cup T_i$ 
6: return  $F$ 

```

In this paper, we consider two ensemble learning techniques that have primarily been used in the context of decision trees: bagging and random forests.

5.1 Bagging

Bagging (Breiman, 1996) 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 classifier. 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 (1996) has shown that bagging can give substantial gains in predictive performance, when applied to an unstable learner (i.e., a learner for which small changes in the training set result in large changes in the predictions), such as classification and regression tree learners.

5.2 Random forests

A random forest (Breiman, 2001a) is an ensemble of trees, where diversity among the predictors is obtained by using bootstrap replicates as in bagging, and additionally by changing the feature set during learning. More precisely, at each node in the decision trees, a random subset of the input attributes is taken, and the best feature is selected from this subset. The number of attributes that are retained is given by a function f of the total number of input attributes D (e.g., $f(D) = 1$, $f(D) = \lfloor \sqrt{D} + 1 \rfloor$, $f(D) = \lfloor \log_2(D) + 1 \rfloor \dots$). 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.

6. 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 instantiations for the algorithms and the statistical tests that we used.

6.1 Experimental questions

Given the methodology from Sections 4 and 5, we construct several types of trees and ensembles. First, we construct PCTs that predict sub-components of the structured output: a separate tree for each variable from the target tuple (CLUS-ST) and a separate tree for each hierarchy edge (CLUS-HSLC). Then, we learn PCTs that predict the structured output simultaneously: a tree for the whole target tuple (CLUS-MT) and a tree for the whole hierarchy (CLUS-HMLC). Similarly, we are constructing the ensemble classifiers (CLUS-ENS-ST, CLUS-ENS-HSLC, CLUS-ENS-MT, CLUS-ENS-HMLC) for both bagging and random forests.

We consider the following questions:

- *Predictive performance*: Can exploitation of the structure of the output lift the predictive performance of an ensemble?
- *Convergence*: Does the performance of the ensembles for structured outputs converge/saturate faster than ensembles that predict sub-components of the output?
- *Efficiency*: How much can the learning process benefit, in terms of time and memory consumption, from the ensembles for structured outputs as compared to the basic ensembles?

We compare the algorithms that predict the complete structured output (CLUS-MT, CLUS-HMLC, CLUS-ENS-MT, CLUS-ENS-HMLC) to the algorithms that predict the components of the structured outputs separately (CLUS-ST, CLUS-HSLC, CLUS-ENS-ST, CLUS-ENS-HSLC). First, we inspect the predictive performance of the algorithms. Then, we focus only on the ensembles and examine the predictive performance at different ensemble sizes (i.e., we construct ‘saturation curves’). Our intention is to check if the performance of the ensembles for structured outputs saturates with smaller number of trees as compared to the saturation of the ensembles that predict the sub-components of the structured outputs. At the end, we compare the running times and the sizes of the obtained models.

6.2 Data description

In this subsection, we present the datasets that were used to evaluate the predictive performance of the ensembles. The datasets are divided in three groups of datasets based on the type of their target concepts: multiple continuous targets datasets (regression), multiple discrete targets datasets (classification) and hierarchical multi-label classification datasets (HMLC). Statistics of the used datasets are presented in Tables 3, 4 and 5, respectively.

The datasets with multiple continuous targets (14 in total, see Table 3) are mainly from the domain of ecological modelling. While the datasets with multiple discrete targets (9 in total, see Table 4) are from various domains: ecological modelling (*Sigma Real* and *Water Quality*), biological (*Yeast*), multimedia (*Scene* and *Emotions*), media space (*Mediana*) etc. Datasets that have classes organized in a hierarchy come from various domains, such as: biology (*Expression-GO*, *SCOP-GO*, *Yeast-GO* and *Sequence-FunCat*), text classification (*Enron*, *Reuters* and *WIPO*) and image annotation/classification (*ImageCLEF2007-D*, *ImageCLEF2007-A* and *Diatoms*). Hence, we use 10 datasets from 3 domains (see Table 5). Note that only the first three datasets from the biological domain have a hierarchy organized

Table 3: Properties of the datasets with multiple continuous targets (regression datasets); N is number of instances, $\overline{D/C}$ number of descriptive attributes (discrete/continuous), and T number of target attributes.

Name of dataset	N	$\overline{D/C}$	T
Collembola (Kampichler et al., 2000)	393	8/39	3
EDM – 1 (Karalič, 1995)	154	0/16	2
Forestry-Kras (Stojanova et al., 2010)	60607	0/160	11
Forestry-Slivnica-LandSat (Stojanova, 2009)	6218	0/150	2
Forestry-Slivnica-IRS (Stojanova, 2009)	2731	0/29	2
Forestry-Slivnica-SPOT (Stojanova, 2009)	2731	0/49	2
Sigma real (Demšar et al., 2005)	817	0/4	2
Soil quality 1 (Demšar et al., 2006)	1944	0/142	3
Solar-flare 1 (Asuncion and Newman, 2007)	323	10/0	3
Solar-flare 2 (Asuncion and Newman, 2007)	1066	10/0	3
Vegetation Clustering (Gjorgjioski et al., 2003)	29679	0/65	11
Vegetation Condition (Kocev et al., 2009)	16967	1/39	7
Water quality (Blockeel et al., 1999; Džeroski et al., 2000)	1060	0/16	14

Table 4: Properties of the datasets with multiple discrete targets (classification datasets); N is number of instances, $\overline{D/C}$ number of descriptive attributes (discrete/continuous), and T number of target attributes.

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

as a DAG (they have GO in the dataset name), and the remaining datasets have tree-shaped hierarchies. For more details on the datasets, we refer the reader to the referenced literature.

6.3 Evaluation measures

Empirical evaluation is the most widely used approach for assessment of the performance of machine learning algorithms. A performance of a machine learning algorithm is computed

Table 5: Properties of the datasets with hierarchical targets; N_{train} is number of instances in the training dataset, N_{test} is number of instances in the testing dataset, D/C is number of descriptive attributes (discrete/continuous), HS is number of classes in the hierarchy, and \bar{L} is average number of labels per example.

Domain	N_{tr}	N_{te}	D/C	HS	\bar{l}	\bar{L}
ImageCLEF2007-D(Dimitrovski et al., 2008)	10000	1006	0/80	XX	1.0	XX
ImageCLEF2007-A(Dimitrovski et al., 2008)	10000	1006	0/80	XX	1.0	XX
Diatoms (ADIAC, 2008)	2065	1054	0/371	XX	1.0	XX
Enron ()	988	660	0/1001	XX	YY	XX
Reuters ()	3000	3000	0/47236	XX	YY	XX
WIPO ()	1352	358	0/74435	XX	YY	XX
Expression-GO (Clare, 2003)	2485	1288	0/551	XX	YY	XX
SCOP-GO (Clare, 2003)	6507	3336	0/2003	XX	YY	XX
Sequence-FunCat (Clare et al., 2006)	2455	1264	2/4448	XX	YY	XX
Yeast-GO (Barutcuoglu et al., 2006)	2310	1155	5588/342	XX	YY	XX

using some evaluation measure. The different machine learning tasks, we previously described, 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 assessment of the algorithm’s performance on the task of predicting multiple continuous targets (regression), we employed three well known measures: correlation coefficient (CC), root mean squared error ($RMSE$) and relative root mean squared error ($RRMSE$). For each of this measure we performed statistical analysis and constructed saturation curves. We present only the results using $RRMSE$, but same conclusions hold if the other two measures are used.

The appropriate usage of evaluation measures in the case of classification algorithms is not as clear as in the case of regression. Sokolova and Lapalme (2009) performed a systematic analysis of twenty four performance measures that can be used in a classification context. They conclude that evaluation measure 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, Matthews correlation coefficient, balanced accuracy (also known as Area Under the Curve) and discriminant power. We used two averaging approaches to adapt these measures for multi-class problems: micro and macro averaging (note that averaging is not needed for accuracy). More about these measures can be found in Sokolova et al. (2006). Since the goal of this study is not to assess the evaluation measures themselves, we present here only micro average F-score ($F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$). However, the conclusions of 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 ($AUP\overline{PRC}$) as suggested by Vens et al. (2008). A Precision-Recall curve plots the precision of a classifier as a function of its recall. The points in the PR space are obtained by varying the value for the threshold τ from 0 to 1 with step 0.02. The precision and recall are micro averaged for all classes from the hierarchy.

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

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

6.4 Experimental setup

Here, we first state the parameter instantiation of the algorithms for constructing the single trees and the ensembles for all types of targets. Then, we describe how we assessed the statistical significance of the differences in the performances of the algorithms.

The single trees for all types of targets are obtained using ‘F-test pruning’. This pruning procedure uses exact Fisher’s test to check whether a given test from an internal node in the tree produces a statistically significant reduction in variance at a given significance level. If there is no test that can satisfy this, then the node is converted to a leaf. For this, we selected an optimal significance level 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 the ensembles requires a size of the ensemble as an input parameter (i.e., number of base classifiers to be constructed). We constructed ensembles with 10, 25, 50, 75 and 100 base classifiers for both multiple targets and hierarchical classification datasets. Additionally, for the datasets with multiple continuous targets we constructed ensembles with 150 and 250 base classifiers, and for the datasets with multiple discrete targets, ensembles with 250, 500 and 1000 base classifiers.

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

The predictive performance of the algorithms on the datasets with multiple targets is estimated by 10-fold cross-validation. The hierarchical datasets were previously divided (by

the data providers) on a train and a test set. Thus, we estimate the predictive performance of the algorithms on the test set.

We adopt the recommendations by Demšar (2006) for the statistical evaluation of the obtained results. We use Friedman test (Friedman, 1940) for statistical significance with the correction from Iman and Davenport (1980). Afterwards, to check where the statistically significant differences appear (between which algorithms), we use Nemenyi post-hoc test (Nemenyi, 1963). We present the results from the statistical analysis with ‘average ranks diagrams’ (see Figures 4, 5, 7, 8, 10 and 11).

7. Results and discussion

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

7.1 Multiple continuous targets

The results from the experiments for evaluation of the algorithms for the task of prediction of multiple continuous targets are presented in Figures 3, 4 and 5. First, we discuss the results with respect to the saturation curves (Figure 3). Next, we discuss the statistical evaluation of the performances (Figure 4). At the end, we compare the efficiency of the algorithms (Figure 5).

In Figure 3, we present the saturation curves for the ensemble methods. Although these curves are averaged accross all target variables for a given dataset, they still provide usefull insight on the performance of the algorithms. The random forests perform better than bagging, both when predicting the multiple targets simultaneously or separately, on the ‘larger’ datasets (the ones with more than 10000 examples), such as *Forestry-Kras* from Figure 3(a). On the other hand, the bagging outperforms the random forests, in both scenarios, on the ‘medium’ datasets (that contain between 1000 and 10000 examples), such as *Soil quality* from Figure 3(b). For the ‘small’ datasets (the ones with less than 1000 examples and less than 10 descriptive attributes), the curves are variable and it is not conclusive which algorithm should be preferred. Also, there is no clear connection between the performance of the algorithms and the number of target variables (i.e., the size of the target tuple). However, on majority of all datasets the ensembles for prediction of multiple targets simultaneously perform better than the ensembles that predict the targets separately.

The averaged saturation curve for all datasets is shown in Figure 3(c). This curve shows that the ensembles for predicting multiple targets simultaneously perform better than the ones predicting the targets separately across all ensemble sizes (except with 100 trees where random forests for multiple targets is worse than random forests for single target). To test which differences in performance are statistically significant, we perform Friedman tests. First, we check at which ensemble size the difference is no longer statistically significant for

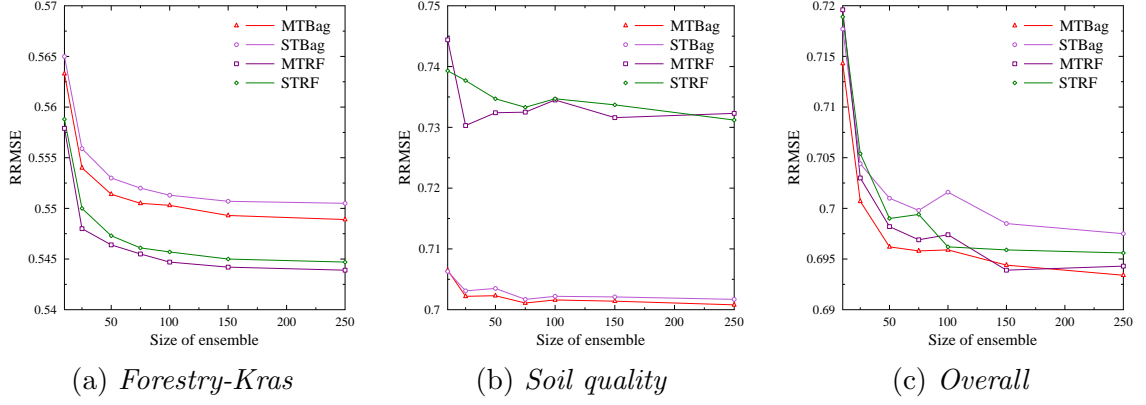


Figure 3: Saturation curves for the prediction of multiple continuous targets. These curves are obtained by averaging the $RRMSE$ values for all of the target variables. Smaller $RRMSE$ values mean better predictive performance. The algorithms are abbreviated as follows: random forests for prediction of multiple targets – $MTRF$, random forests for prediction of single target – $STRF$, bagging for prediction of multiple targets – $MTBag$ and bagging for prediction of single target – $STBag$.

each method separately. In this case, for all algorithms, the difference is not statistically significant after 50 trees are added. Thus, we compare the performance of the algorithms after 50 trees and after 250 trees (the maximal number of trees).

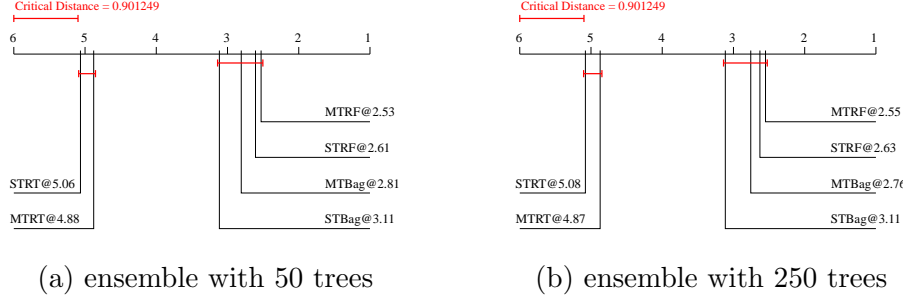


Figure 4: Average rank diagrams at significance level of 0.05 for prediction of multiple continuous targets. The difference in the performance of the algorithms connected with a red line are not statistically significant. The numbers after the name of the algorithm indicate its average rank. The abbreviations are same as in Figure 3 with addition of predicting clustering tree for multiple continuous targets – $MTRT$ and predictive clustering tree for single continuous target – $STRT$.

The statistical tests in Figure 4 show that the difference in the performance of the ensemble methods is not statistically significant at the level of 0.05. However, best performing method is random forests for predicting multiple targets and worst performing method is bagging for predicting the multiple targets separately. If more trees are added, the ordering

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

Finally, we compare the algorithms by their running time and the size of the models when the ensembles consist of 50 trees (see Figure 5). The statistical tests show that both random forests and bagging for predicting multiple targets simultaneously outperform significantly, in terms of size of models, the ensembles that predict multiple targets separately. In terms of time efficiency, random forests for multiple targets outperform significantly both ensemble methods for predicting the targets separately. Also, bagging for multiple targets are significantly faster to construct than bagging for separate prediction of the targets.

Let us further examine the speed-up and the size of the models ratios. Random forests for predicting multiple targets simultaneously are ~ 3.3 times faster to construct and the models are ~ 3.75 times smaller than random forests for predicting single target. In addition, they are ~ 3.7 times faster to construct and have ~ 1.14 times smaller models than bagging for multiple targets. Furthermore, bagging for predicting multiple targets are ~ 3 times faster and ~ 3.6 times smaller than bagging for predicting single target.

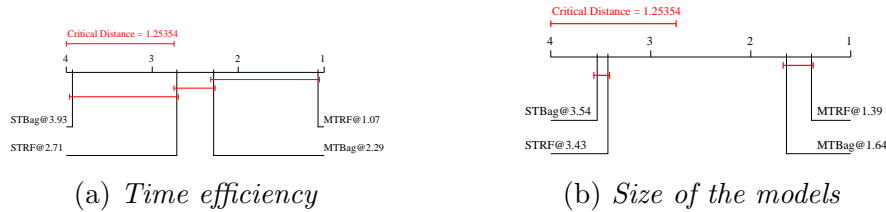


Figure 5: Efficiency of the ensembles for prediction of multiple continuous targets. The size of the ensembles is 50 trees.

To summarize, ensembles for predicting multiple continuous targets simultaneously perform better than ensembles predicting multiple targets separately. While the difference in predictive performance is not statistically significant, the differences in efficiency are. Random forests have higher predictive performance than bagging on the larger datasets, while on the medium datasets bagging ensembles are better. In terms of efficiency, the algorithms that predict the multiple targets simultaneously (especially the random forests) should be always preferred.

7.2 Multiple discrete targets

The performance of the algorithms for multi-class classification can be assessed using different measures, some of which we listed in Section 6.3. The evaluation measure should be selected based on the application domain (Sokolova and Lapalme, 2009). In our study, we used micro weighted averaged F-score ($\mu F - score$): reasonable compromise between all measures, since it combines the precision and the recall values.

The results for algorithms that predict multiple discrete targets are presented in Figures 6, 7 and 8. In Figure 6, we present the saturation curves. Next, we discuss the

statistical analysis of the results (Figure 7). At the end, we compare the algorithms by their efficiency (Figure 8).

In Figure 6, we present three saturation curves for the four ensemble methods. Same as for predicting multiple continuous targets, these values are averaged from all target variables for a given dataset (and in Figure 6(c) averaged across all datasets). These saturation curves offer us several insights to the performance of the ensembles on the task of predicting multiple discrete targets. The saturation curves for the smaller datasets (ones with less than 1000 examples) are variable (for instance, see the saturation curve for the *Sigma real* dataset shown in Figure 6(a)). However, we can note that on the smaller ensemble sizes the ensembles that predict the targets simultaneously outperform the ensembles that predict the targets separately.

The saturation curves for the larger datasets (with more than 1000 examples) are more stable and we can observe two types of behaviour: (1) on the datasets with less than 30 descriptive variables, the ensembles for predicting the targets simultaneously outperform the ensembles that predict the targets separately (for instance, see the saturation curve for the *Water quality* dataset shown in Figure 6(b)); (2) on the datasets with more than 30 descriptive variables, the ensembles for predicting the targets simultaneously are better when the size of the ensemble is small than the ensembles that predict the multiple targets separately, while on the ensembles with bigger sizes the situation is reversed. Similar behaviour can be also noticed on the *Overall* saturation curve (Figure 6(c)). Finally, same as for the multiple continuous targets, there is no connection between the predictive performance of the algorithms and the size of the target tuple.

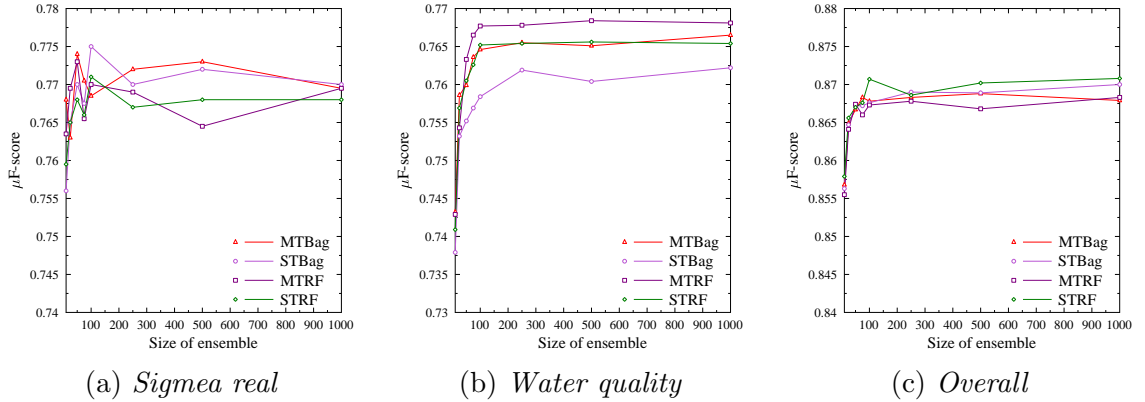


Figure 6: Saturation curves for the prediction of multiple discrete targets. These curves are obtained by averaging the μF – score values for all of the target variables. Bigger μF – score values mean better predictive performance. The algorithms are abbreviated as follows: random forests for prediction of multiple targets – *MTRF*, random forests for prediction of single target – *STRF*, bagging for prediction of multiple targets – *MTBag* and bagging for prediction of single target – *STBag*.

The results from the statistical analysis of the predictive performance (μF – score) are shown in Figure 7. First, for each ensemble method separately, we check at which ensemble size the predictive performance is no longer statistically 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 the random forests and 250 for bagging. After this, we select ensembles sizes of 50 (Figure 7(a)) and 1000 (maximal number of trees, Figure 7(b)) and compare the algorithms.

The statistical tests reveal that there is no statistically significant difference in the performance of the ensemble methods and that all ensemble methods perform statistically significantly better than single tree. When the ensembles have 50 trees, the bagging for predicting the multiple targets simultaneously is best performing method (average rank 2.59) and the remaining methods have smaller and very close to each other average ranks (ranging from 3.0 to 3.11) with random forest for separate prediction of the targets having the smallest average rank. The situation is similar with 1000 trees, with the difference that now random forests for simultaneous prediction of the targets are worst performing method (average rank 3.26) and the other three methods have very close average ranks (from 2.71 to 2.75) with random forest for separate prediction being the best performing method. This just confirms the findings with the saturation curves: adding of trees helps more to the ensembles that predict the targets separately than the ensembles that predict the targets simultaneously.

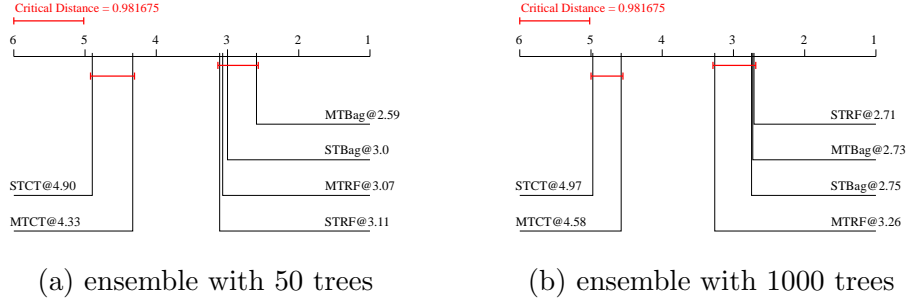


Figure 7: Average ranks diagrams at significance level of 0.05 for prediction of multiple discrete targets. The difference in the performance of the algorithms connected with a red line are not statistically significant. The numbers after the name of the algorithm indicate its average rank. The abbreviations are same as in Figure 6 with addition of predictive clustering tree for multiple discrete targets – *MTCT* and predictive clustering tree for single discrete target – *STCT*.

At the end, 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 only state that the random forests for predicting multiple targets simultaneously significantly outperform the bagging for predicting the multiple targets separately. As for the size of the models, we can note the following: (1) the 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 the random forests for separate prediction of the targets.

We further investigate the running times and size of models ratios. The random forests for predicting multiple targets simultaneously are ~ 2.3 times faster to construct and have ~ 2.1 times smaller models than the random forests for separate prediction of the targets.



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

Also, they are ~ 5.6 times faster and have ~ 1.14 times bigger models than bagging for predicting multiple targets simultaneously. Furthermore, bagging for predicting multiple targets simultaneously is ~ 2.5 times faster and have ~ 1.9 times smaller models than bagging for separate prediction of multiple targets.

In summary, the predictive performances of the ensemble methods for predicting multiple targets simultaneously and the ones for separate prediction are not statistically significantly different. However, the ensemble methods for predicting multiple targets simultaneously are better when the number of trees in the ensemble is smaller. Furthermore, they should be preferred if the efficiency of the classifier is an issue. The ensemble methods for simultaneous prediction are faster (especially random forests) and smaller (especially bagging) than the ensemble methods for separate predictions.

7.3 Hierarchical multi-label classification

In this subsection, we present the results for the task of hierarchical classification in a similar way as for the task of predicting multiple targets. We assess the performance of the algorithms using the area under the average precision-recall curve ($AUPRC$) as suggested by Vens et al. (2008). The results are presented with saturation curves (Figure 9), statistical tests (Figure 10) and efficiency evaluation (Figure 11).

The saturation curves for the different domains (functional genomics, image annotation and text classification) show different behaviour, thus we discuss the curves for each domain separately. On the domain of functional genomics, the ensembles for HMLC outperform the ensembles for HSLC when the target hierarchy is organized as DAG (for instance, see the saturation curve for the *SCOP-GO* dataset in Figure 9(a)). Moreover, the random forests for HMLC are the best performing method. The ensembles for HMLC also outperform the ensembles for HSLC on the domain of image annotation/classification (for instance, see the saturation curve for the *ImageCLEF2007-D* dataset in Figure 9(b)). On these datasets, the bagging for HMLC is the best performing method. The situation is different on the text classification domains. Here, the ensembles of HSLC outperform the ensembles of HMLC. We hypothesize that this is because of the large number of descriptive variables. The performance of ensembles of HMLC on text classification datasets should be further investigated.

The overall saturation curve (Figure 9(c)) shows the performance of the algorithms averaged over the datasets from the three domains. Best performing method is random forest for HMLC and worst performing method is bagging for HSLC. To further investigate the

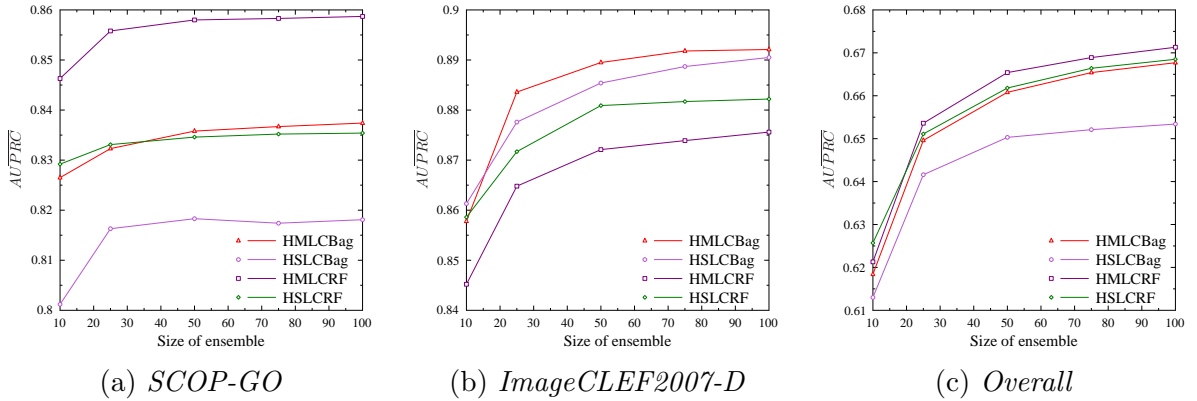


Figure 9: Saturation curves for hierarchical multi-label classification. These curves are obtained by averaging the $AUPRC$ values for all of the target variables. Bigger $AUPRC$ values mean better predictive performance. The algorithms are abbreviated as follows: random forests for hierarchical multi-label classification – *HMLCRF*, random forests for hierarchical single-label classification – *HSLCRF*, bagging for hierarchical multi-label classification – *HMLCBag* and bagging for hierarchical single-label classification – *HSLCBag*.

differences in the performances, we perform statistical analysis for each method separately for all ensemble sizes. We do this to check when adding of trees in the ensemble does not statistically significantly improves the predictive performance. The ensembles for HMLC and random forests for HSLC saturate after 50 trees are added in the ensemble, while bagging for HSLC saturates after only 25 trees. We further compare the performance of the ensembles at 50 trees and 100 trees (results presented in Figure 10).

The average ranks diagram for the ensembles with 50 trees (Figure 10(a)) shows that the performance of the ensembles is not statistically significantly different. Note that the best performing method is random forests for HSLC (average rank 2.25) and worst performing method is bagging for HSLC (average rank 2.85). Similarly, there is no statistically significant difference in performance when the ensembles contain 100 trees. Again, bagging for HSLC (average rank 2.9) is the worst performing method, but bagging for HMLC (average rank 2.2) is now the best performing method. In both cases, the ensemble methods significantly outperform a single predictive clustering trees.

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

We further investigate the speed up and size of the models ratios. The random forests for HMLC are ~ 6.4 times faster and have ~ 4.6 times smaller models than the random forests for HSLC. Similarly, bagging for HMLC is ~ 6.4 times faster and have ~ 3.2 times

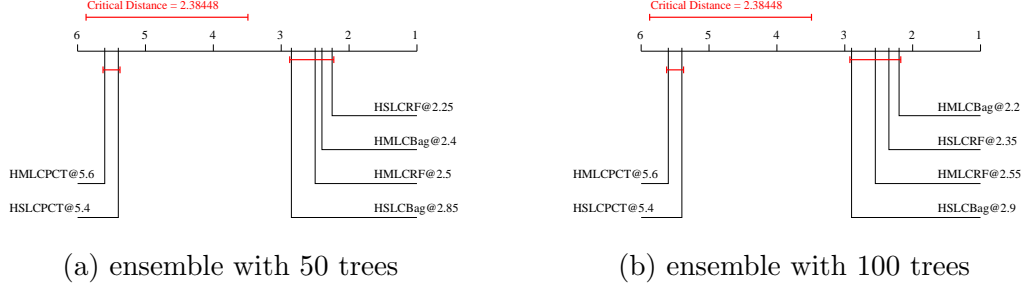


Figure 10: Average ranks diagrams at significance level of 0.05 for hierarchical multi-label classification. The difference in the performance of the algorithms connected with a red line are not statistically significant. The numbers after the name of the algorithm indicate its average rank. The abbreviations are same as in Figure 9 with addition of predicting clustering tree for hierarchical multi-label classification – *HMLCPCT* and predictive clustering tree for hierarchical single-label classification – *HSLCPCT*.

smaller models than bagging for HSLC. Random forests for HMLC are ~ 7.8 times faster and ~ 1.1 times smaller models than bagging for HMLC. All in all, in terms of efficiency, random forests for HMLC outperform the rest of the ensemble methods.

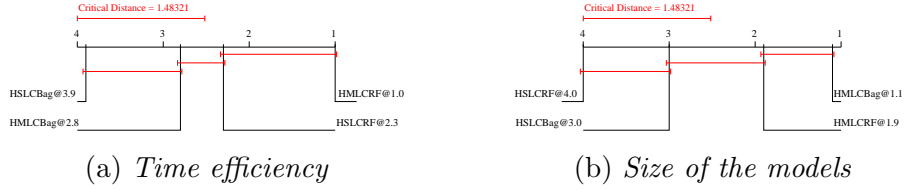


Figure 11: Efficiency of the ensembles for hierarchical multi-label classifications. The size of the ensembles is 50 trees.

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

8. Conclusions

In this article, we present an approach for learning ensembles for predicting structured outputs. The proposed approach constructs a single model to make a prediction for the whole structure simultaneously. It is general with respect to the type of the output: it can handle multiple target variables and hierarchically structured classes (tree-shaped and DAGs). It is scalable to wide range of datasets with different number of examples and descriptive vari-

ables and different sizes of the structured output. We experimentally evaluate the proposed approach over number of datasets. The conclusions can be summarized as follows.

With the use of ensembles, we lift the predictive performance of a classifier when the target is a structure (the difference in the performance is statistically significant at 0.05). This was previously shown only on problems where the target was a single continuous or single discrete variable. Here, we show that it is valid for the three machine learning tasks at hand: predicting multiple continuous targets, predicting multiple discrete targets and hierarchical multi-label classification. This finding suggests that the non-trivial relations that might exist between the sub-components of the structure are included when combining predictions of several classifiers or when injecting some source of randomness in the learning algorithm.

Next, we look at the saturation point of the ensembles' performances with respect to the number of base classifiers. In majority of the cases, the predictive performance of the ensembles saturates after the 50th tree is added in the ensemble. Exceptions of this are: bagging for HSLC 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 given dataset (summarized by their number of examples and number of descriptive variables). Also, the curves show that on majority of the datasets the ensembles for predicting the structured outputs as a whole have better performance than the ensembles that predict the sub-components. This is the case especially when the ensembles contain smaller number of trees.

Afterwards, we perform tests for checking the statistical significance of the differences in the algorithms' performances. The differences in the performances of ensembles are not statistically significant at 0.05 in any of the tasks. However, the ensembles for predicting structured output often had smaller average ranks than the ensembles for predicting the sub-components of the structure.

At the end, we compare the ensembles by their efficiency: time needed to construct the ensembles and the size of the models. The random forests for predicting structured outputs outperform all other methods in terms of time efficiency. Moreover, they statistically significantly outperform bagging for predicting the sub-components of a structured output on all tasks. Regarding the size of the models, on the task of predicting multiple continuous targets, the random forests for predicting structured outputs are with the smallest models, while on the other two tasks, bagging for predicting structured outputs are the smallest. On the three tasks, the ensembles for predicting structured outputs have smaller models than the ensembles that predict sub-components of a structured output.

To summarize, we present ensemble methods for predicting structured outputs that have good (and in many cases better) predictive performance and are very efficient as compared to the ensembles that predict sub-components of a structured output. Furthermore, the ensembles for predicting structured outputs significantly lift the predictive performance of a single classifier.

Acknowledgments

The authors would like to thank Ivica Dimitrovski from the Faculty of Electrical Engineering and Information Technologies, Skopje, Macedonia for providing the image annotation datasets.

References

- ADIAC. Automatic diatom identification and classification. <http://rbg-web2.rbge.org.uk/ADIAC/>, 2008.
- Rie K. Ando, Tong Zhang, and Peter Bartlett. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6: 1817–1853, 2005.
- Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73:243–272, 2008.
- Michael Ashburner et al. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25:25–29, 2000.
- Arthur Asuncion and David Newman. UCI - machine learning repository, 2007. URL <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- Gökhan H. Bakır, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan. *Predicting structured data*. Neural Information Processing. The MIT Press, 2007.
- Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.
- Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.
- Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1):105–139, 1999.
- Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.
- Shai Ben-David and Reba Schuller Borbely. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine Learning*, 73(3):273–287, 2008.
- Hendrik Blockeel, Sašo Džeroski, and Jasna Grbović. Simultaneous prediction of multiple chemical parameters of river water quality with tilde. In *Proceedings of the 3rd European Conference on PKDD - LNAI 1704*, pages 32–40. Springer, 1999.
- Hendrik Blockeel and Jan Struyf. Efficient algorithms for decision tree cross-validation. *Journal of Machine Learning Research*, 3:621–650, 2002.
- Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-down induction of clustering trees. In J. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann, 1998.

- Hendrik Blockeel, Maurice Bruynooghe, Sašo Džeroski, Jan Ramon, and Jan Struyf. Hierarchical multi-classification. In *KDD-2002 Workshop Notes: MRDM 2002, Workshop on Multi-Relational Data Mining*, pages 21–35, 2002.
- Hendrik Blockeel, Leander Schietgat, Jan Struyf, Sašo Džeroski, and Amanda Clare. Decision trees for hierarchical multilabel classification: A case study in functional genomics. In *Knowledge Discovery in Databases: PKDD 2006 - LNCS 4213*, pages 18–29. Springer Berlin / Heidelberg, 2006.
- Matthew Boutell, Jiebo Luo, Xipeng Shen, and Christopher Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.
- Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001a.
- Leo Breiman. Using iterated bagging to debias regressions. *Machine Learning*, 45(3):261–277, 2001b.
- Leo Breiman and Jerome Friedman. Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(1):3–54, 1997.
- Leo Breiman, Jerome Friedman, R.A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.
- P. J. Brown and J. V. Zidek. Adaptive multivariate ridge regression. *The Annals of Statistics*, 8(1):64–74, 1980.
- Feng Cai and Vladimir Cherkassky. Svm+ regression and multi-task learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 418–424, 2009.
- Andrea Caponnetto, Charles A. Micchelli, Massimiliano Pontil, and Yiming Ying. Universal multi-task kernels. *Journal of Machine Learning Research*, 9:1615–1646, 2008.
- Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.
- Yu Chen and Dong Xu. Global protein function annotation through mining genome-scale data in yeast *saccharomyces cerevisiae*. *Nucleic Acids Research*, 32(21):6414–6424, 2004.
- Amanda Clare. *Machine learning and data mining for yeast functional genomics*. PhD thesis, University of Wales Aberystwyth, Aberystwyth, Wales, UK, 2003.
- Amanda Clare, Andreas Karwath, Helen Ougham, and Ross D. King. Functional bioinformatics for *arabidopsis thaliana*. *Bioinformatics*, 22(9):1130–1136, 2006.
- Damjan Demšar, Marko Debeljak, Sašo Džeroski, and Claire Lavigne. Modelling pollen dispersal of genetically modified oilseed rape within the field. In *The Annual Meeting of the Ecological Society of America*, 2005.

- Damjan Demšar, Sašo Džeroski, Thomas Larsen, Jan Struyf, Jorgen Axelsen, Marianne Bruns-Pedersen, and Paul Henning Krogh. Using multi-objective classification to model communities of soil. *Ecological Modelling*, 191(1):131–143, 2006.
- Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
- Thomas G. Dietterich. Ensemble methods in machine learning. In *Proc. of the 1st International Workshop on Multiple Classifier Systems - LNCS 1857*, pages 1–15. Springer, 2000.
- Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Džeroski. Hierchical annotation of medical images. In *Proceedings of the 11th International Multiconference - Information Society IS 2008*, pages 174–181. IJS, Ljubljana, 2008.
- Sašo Džeroski. Towards a general framework for data mining. In Sašo Džeroski and Jan Struyf, editors, *Knowledge Discovery in Inductive Databases, 5th International Workshop, KDID 2006, Revised Selected and Invited Papers*, volume 4747, pages 259–300, 2007.
- Sašo Džeroski, Damjan Demšar, and Jasna Grbović. Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, 13(1):7–17, 2000.
- André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press, 2001.
- Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.
- Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. of the Thirteenth International Conference on Machine Learning - ICML*, pages 148–156. Morgan Kaufman, 1996.
- Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Journal of Machine Learning Research*, 11(1):86–92, 1940.
- Pierre Geurts, Louis Wehenkel, and Florence D’Alché-Buc. Kernelizing the output of tree-based methods. In *ICML ’06: Proceedings of the 23rd international conference on Machine learning*, pages 345–352. ACM, 2006.
- Valentin Gjorgjioski, Sašo Džeroski, and Matt White. Clustering analysis of vegetation data. Technical Report 10065, Jožef Stefan Institute, 2003.
- William H. Greene. *Econometric analysis*. Prentice Hall, 6th edition, 2007.
- Yuanfang Guan, Chad L. Myers, David C. Hess, Zafer Barutcuoglu, Amy A. Caudy, and Olga G. Troyanskaya. Predicting gene function in a hierarchical context with an ensemble of classifiers. *Genome biology*, 9(S1):S3+, 2008.
- Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

- Tin K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
- Ronald L. Iman and James M. Davenport. Approximations of the critical region of the friedman statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, 1980.
- Christian Kampichler, Sašo Džeroski, and Ralf 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):197–209, 2000.
- Aram Karalič. *First Order Regression*. PhD thesis, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia, 1995.
- Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski. Ensembles of multi-objective decision trees. In *ECML '07: Proceedings of the 18th European Conference on Machine Learning – LNCS 4701*, pages 624–631. Springer Berlin / Heidelberg, 2007.
- Dragi Kocev, Sašo Džeroski, Matt White, Graeme Newell, and Peter Griffioen. Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling*, 220(8):1159–1168, 2009.
- Ludmila Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
- Hyunju Lee, Zhidong Tu, Minghua Deng, Fengzhu Sun, and Ting Chen. Diffusion kernel-based logistic regression models for protein function prediction. *OMICS: A Journal of Integrative Biology*, 10(1):40–55, 2006.
- Charles A. Micchelli and Massimiliano Pontil. Kernels for multi-task learning. In *Advances in Neural Information Processing Systems 17 - Proceedings of the 2004 Conference*, pages 921–928, 2004.
- Sara Mostafavi, Debajyoti Ray, David Warde-Farley, Chris Grouios, and Quaid Morris. Genemania: a real-time multiple association network integration algorithm for predicting gene function. *Genome biology*, 9(S1):S4+, 2008.
- Peter Bjorn Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, Princeton, NY, USA, 1963.
- Guillaume Obozinski, Gert Lanckriet, Charles Grant, Michael I. Jordan, and William S. Noble. Consistent probabilistic outputs for protein function prediction. *Genome Biology*, 9(S1):S6+, 2008.
- Ross J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1 edition, 1993.
- Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, 2006.

- Carlos Silla and Alex Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, pages 1–42, 2010. ISSN 1384-5810. doi: 10.1007/s10618-010-0175-9.
- Maja Skrjanc, Marko Grobelnik, and Darko Zupanic. Insights offered by data-mining when analyzing media space data. *Informatica (Slovenia)*, 25(3):357–363, 2001.
- Ivica Slavkov, Valentin Gjorgjioski, Jan Struyf, and Sašo Džeroski. Finding explained groups of time-course gene expression profiles with predictive clustering trees. *Molecular BioSystems*, 6(4):729–740, 2010.
- Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.
- Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *AI 2006: Advances in Artificial Intelligence - LNCS 4304*, pages 1015–1021. Springer Berlin / Heidelberg, 2006.
- Daniela Stojanova. Estimating forest properties from remotely sensed data by using machine learning. Master’s thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, 2009.
- Daniela Stojanova, Panče Panov, Valentin Gjorgjioski, Andrej Kobler, and Sašo Džeroski. Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics*, 5(4):256–266, 2010.
- Jan Struyf and Sašo 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*, pages 222–233. Springer, 2006.
- Sebastian Thrun and Lorien Pratt. *Learning to learn*. Kluwer Academic Publishers, 1998.
- Weidong Tian, Lan V. Zhang, Murat Taşan, Francis D. Gibbons, Oliver D. King, Julie Park, Zeba Wunderlich, J. Michael Cherry, and Frederick P. Roth. Combining guilt-by-association and guilt-by-profiling to predict *saccharomyces cerevisiae* gene function. *Genome biology*, 9(S1):S7+, 2008.
- Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis Vlahavas. Multilabel classification of music into emotions. In *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 325–330, 2008.
- Giorgio Valentini and Matteo 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*, pages 133–146, 2009.
- Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

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