Computer Science and Information Systems 17(2):459–486 https://doi.org/10.2298/CSIS190928006S

Option predictive clustering trees for multi-target regression*

Tomaž Stepišnik[1,2], Aljaž Osojnik[1,2], Sašo Džeroski[1,2], and Dragi Kocev[1,2]

[1] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
{tomaz.stepisnik,aljaz.osojnik,saso.dzeroski,dragi.kocev}@ijs.si

Abstract. Decision trees are one of the most widely used predictive modelling methods primarily because they are readily interpretable and fast to learn. These nice properties come at the price of predictive performance. Moreover, the standard induction of decision trees suffers from myopia: a single split is chosen in each internal node which is selected in a greedy manner; hence, the resulting tree may be sub-optimal. To address these issues, option trees have been proposed which can include several alternative splits in a new type of internal nodes called option nodes. Considering all of this, an option tree can be also regarded as a condensed representation of an ensemble. In this work, we propose to learn option trees for multi-target regression (MTR) based on the predictive clustering framework. The resulting models are thus called option predictive clustering trees (OPCTs). Multi-target regression is concerned with learning predictive models for tasks with multiple numeric target variables. We evaluate the proposed OPCTs on 11 benchmark MTR data sets. The results reveal that OPCTs achieve statistically significantly better predictive performance than a single predictive clustering tree (PCT) and are competitive with bagging and random forests of PCTs. By limiting the number of option nodes, we can achieve a good trade-off between predictive power and efficiency (model size and learning time). We also perform parameter sensitivity analysis and bias-variance decomposition of the mean squared error. Our analysis shows that OPCTs can reduce the variance of PCTs nearly as much as ensemble methods do. In terms of bias, OPCTs occasionally outperform other methods. Finally, we demonstrate the potential of OPCTs for multifaceted interpretability and illustrate the potential for inclusion of domain knowledge in the tree learning process.

Keywords: multi-target regression, option trees, interpretable models, predictive clustering trees, bias-variance decomposition of error.

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 (scalar-valued) class of a previously unseen example. However, in many real life problems of predictive modelling the output (target) is structured, e.g. it is a vector of classs values or a tuple of target variables. There can be dependencies between the class values/targets (e.g., they can be organized

* The authors wish it to be known that the first two authors should be regarded as joint first authors.

into a tree-shaped hierarchy or a directed acyclic graph) or some internal relations between the class values may exist (e.g., as in sequences).

In this work, we concentrate on the task of predicting multiple numeric variables. Examples thus take the form $(\mathbf{x_i}, \mathbf{y_i})$, where $\mathbf{x_i} = (x_{i1}, \ldots, x_{ik})$ is a vector of k input variables and $\mathbf{y_i} = (y_{i1}, \ldots, y_{it})$ is a vector of t target variables. This task is known under the name of *multi-target regression* (MTR) [4, 23, 24] (also known as multi-output or multivariate regression). MTR is a type of structured output prediction task which has applications in many real life problems, where we are interested in simultaneously predicting multiple numeric variables. Prominent examples come from ecology and include predicting the abundance of different species living in the same habitat [12] and predicting properties of forests [21, 29]. Due to its applicability to a wide range of domains, this task is recently gaining increasing interest in the research community.

Several methods for addressing the task of MTR have been proposed [24, 31]. These methods can be categorized into two groups of methods [2]: (1) local methods, that predict each of the target variable separately and then combine the individual model predictions to get the overall model prediction and (2) global methods, that predict all of the variables simultaneously (also known as 'big-bang' approaches). In the case of local models, for a domain with t target variables one needs to construct t predictive models – each predicting a single target. The prediction vector (that consists of t components) of an unseen example is then obtained by concatenating the predictions of the multiple single-target predictive models. Conversely, in the case of global models, for the same problem one needs to construct only one model. In this case, the prediction vector of an unseen example is obtained by passing the example through the model and getting its (complete) prediction.

In the past, several researchers proposed methods for solving the task of MTR directly and demonstrated their effectiveness [1, 8, 10, 21, 24, 20, 30]. 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 happen that the number of components in the output is very large (e.g., predicting the bioactivity profiles of compounds described with their quantitative structure-activity relationships on a large set of proteins), 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 state-of-the-art methods for MTR are based on tree and ensemble learning [8, 24, 27, 31]. Trees for MTR (from the predictive clustering framework) inherit the properties of regression trees: they are interpretable models, but their construction is greedy. The performance of trees is significantly improved when they are used in an ensemble setting [24, 20]. However, the myopia, i.e., greediness, of the tree construction process can lead to learning sub-optimal models. One way to alleviate this is to use a beam-search algorithm for tree induction [22], while another approach is to introduce option splits in the nodes [9, 25].

Beside the many appealing properties of trees, their predictive performance is often limited. In a variety of machine learning tasks including MTR, learning ensembles of trees typically significantly improve the predictive power of the single models [24]. However, learning ensembles has a significant drawback – lack of a possibility to interpret the obtained predictive model. It is not feasible to analyze each tree in an ensemble due to both

the number of trees and the randomized procedure that is used to learn them. Typically, this means that there is a trade-off between interpretability and predictive performance.

To address this issue, we propose to extend predictive clustering trees (PCTs) for MTR towards option trees, i.e., we introduce option predictive clustering trees (OPCTs). An option tree can be seen as a condensed representation of an ensemble of trees which shares a common substructure. More specifically, the heuristic function for split selection can return multiple values that are close to each other within a predefined range. These splits are then used to construct an option node. For illustration, see Figure 1.

The contributions of this work can be summarized as follows:

- We introduce a new method for addressing the task of MTR that provides a balance between interpretability and predictive performance: An option PCT can be treated as an ensemble, or the best subtree can be extracted and analyzed as a single tree.
- We analyze the computational complexity of the proposed method and compare it to the complexity of the competing methods.
- We perform an empirical evaluation of the performance of the proposed method along three dimensions: predictive power, time efficiency and size of the models.
- We further analyse the predictive performance by examining the bias-variance decomposition of the models' errors. We perform the analysis for the proposed method as well as the competing methods. As far as we are aware, such an analysis has not been performed before for MTR methods.

This work extends our earlier work presented in [26] along two major directions. First, we perform a theoretical analysis of the computational complexity of the proposed method and compared it to the computational complexity of learning a single PCT and ensembles of PCTs. Second, we calculate the bias-variance decompositions of mean squared errors to determine which components of the error are addressed by different parameter selections and competing methods.

The remainder of this paper is organized as follows. Section 2 proposes the algorithm for learning option PCTs for MTR. Next, Section 3 outlines the design of the experimental evaluation and the details on the bias-variance decomposition of the error. Section 4 continues with a discussion of the results. Finally, Section 5 concludes and provides directions for further work.

2. Option predictive clustering trees

The predictive clustering trees 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 [3], which is available for download at http://clus.sourceforge.net.

PCTs are induced following the *top-down induction of decision trees* (TDIDT) algorithm [7]. There are however two major differences: the heuristic score and the prototype function are instantiated by the PCT algorithm based on the task that is being addressed (classification, regression or multi-target prediction etc.). The heuristic score used for selecting the tests in the internal nodes of a regular PCT is the maximization of the variance reduction resulting from the partitioning of the instances into subtrees corresponding to

the outcome of the tests. By doing so, we also maximize cluster homogeneity, and, consequently, improve the predictive performance. The PCT algorithm also defines the prototype functions used in each tree leaf to give predictions for new examples based on the task at hand (e.g., uses averaging for MTR).

Option predictive clustering trees (OPCT) extend the usual PCT framework, by introducing option nodes into the tree building procedure outlined in Algorithm 1. Option decision trees were first introduced as classification trees by [9] and then analyzed in more detail by [25]. [17] analyzed regression option trees in the context of data streams.

The major motivation for the introduction of option trees is to address the myopia of the *top-down induction of decision trees* (TDIDT) algorithm [7]. Viewed through the lens of the predictive clustering framework, a PCT is a non-overlapping hierarchical clustering of the whole input space. Each node/subtree corresponds to a clustering of a subspace and prediction functions are placed in the leaves, i.e., lowest clusters in the hierarchy. An OPCT, however, allows the construction of an overlapping hierarchical clustering. This means that, at each node of the tree several alternative hierarchical clusterings of the subspace can appear instead of a single one.

When using an OPCT for prediction on a new example, we produce the prediction by aggregating over the predictions of the alternative subtrees (overlapping clusters) the example may encounter. However, as not all parts of the tree (hierarchical clustering) are necessarily overlapping, the example may encounter only nonoverlapping (sub)clusters. In that case, we produce the prediction as we would with a regular PCT.

When using TDIDT to construct a predictive clustering tree, and in particular when partitioning the data, all possible splits are evaluated by using a heuristic and the best one is selected. However, other splits may have very similar heuristic values. The best partition could be obtained with another split as a consequence of noise or of the sampling that generated the data. In this case, selecting a different split could be optimal. To address this concern, the use of option nodes was proposed [25].

An option node is introduced into the tree when it would be hard to determine the best split, that is when the best splits have similar heuristic values. When this occurs, instead of selecting only the best split, we select several of them. Specifically, we select up to 5 splits s, called *options*, that satisfy the following

$$\frac{\mathrm{Heur}(s)}{\mathrm{Heur}(s_{best})} \geq 1 - \varepsilon \cdot d^{level},$$

where s_{best} is the best split, ε determines how similar the heuristics must be, $d \in [0, 1]$ is a decay factor and $level$ is the level in the tree of the node we are attempting to split. This equation assumes that heuristic scores are to be maximized (higher value is better). After we have determined the candidate splits, we introduce an option node whose children are split nodes obtained by using the selected splits, i.e., an option node contains options as its children. Selecting more than 5 options is possible, but, uses more resources and is not advised [25]. A drawback of this method of split selection is that if some attributes are highly correlated to the attribute that produces the best split, those attributes will likely be selected for other splits in the option node. In this case the option node will only increase the learning time.

As usual, we define the level of a node to be the number of its ancestor nodes, however, we do not count option nodes. This is motivated by the predictive clustering viewpoint,

i.e., the option nodes only mark that there are overlapping clusters and not that there is an additional level of clustering.

The use of a decay factor makes the selection criterion more stringent in the lower nodes of the tree. The intuition behind this is that higher up, the split selection is more important and a larger error would be inferred by introducing a non-optimal split. However, as we get deeper into the tree, the use of a non-optimal split makes decreasing impact. This intuition also allows us to prohibit the use of option nodes on levels 3 and greater, which severely mitigates the problem of combinatorial explosion.

Outline of the entire tree building process is presented in algorithm 1. The variable *candidates* is a list of at most 5 tests with the highest heuristic values. Every test is represented as a triplet (t, h, P), where t is the actual test function, h its heuristic value and P the partitions produced by the test.

When using a small ε, e.g., $\varepsilon = 0.1$, we are selecting only options whose heuristics are within 10% of the best split. However, the use of larger ε, in the extreme case even $\varepsilon = 1$, can also be motivated through the success of methods such as random forests and ensembles of extremely randomized trees. Allowing the selection of splits whose heuristic values are considerably worse than the heuristic value of the best split might not necessarily reduce the performance of the tree, but actually increase it.

Algorithm 1 The top-down induction algorithm for option PCTs.

Procedure OptionPCT

Input: A data set E, parameter ε, decay factor d, current tree level l
Output: An option predictive clustering tree

```
candidates = FindBestTests(E, 5)
if |candidates| > 0 then
    if |candidates| = 1 or l > 2 then
        (t*, h*, P*) = candidates[0]
        for each E_i ∈ P* do
            tree_i = OptionPCT(E_i, ε, d, l + 1)
        return node(t*, ⋃_i {tree_i})
    else
        (t*_0, h*_0, P*_0) = candidates[0]
        nodes = {}
        for each (t*_i, h*_i, P*_i) ∈ candidates do
            if h*_i / h*_0 ≥ 1 − ε · d^l then
                for each E_j ∈ P*_i do
                    tree_j = OptionPCT(E_j, ε, d, l + 1)
                nodes = nodes ∪ {node(t*, ⋃_j {tree_j})}
        if |nodes| > 1 then
            return option_node(nodes)
        else
            return nodes[0]
else
    return leaf(Prototype(E))
```

Once an OPCT is built, we want to use it for prediction. In a regular PCT, it is simple to produce a prediction for a new example. It is sorted into a leaf (reached according to the splits of the tree) where a prediction is made by using a prototype function. When traversing an example through an OPCT, we behave the same when we encounter a split or leaf node. If we traverse an example to an option node, however, we clone the example for each of the options and traverse one of the copies down each of the options. This means that in an option node an example is (by proxy of its copies) traversed to multiple leaves, where multiple predictions are produced. To obtain a single prediction in an option node, we aggregate the obtained predictions. When addressing multi-target regression this is generally done by averaging all the predictions per target.

An option tree is usually seen as a single tree, however, it can also be interpreted as a compact representation of an ensemble. To generate the ensemble of the *embedded trees*, we start recursively from the root node and move in a top-down fashion. Each time we encounter an option node we copy the tree above (and in "parallel") for each of the options and replace the option node with only the option, i.e., single split. This produces one tree for each option while removing the option node in question. We repeat this procedure on all the generated trees until we are left with no option nodes. This is illustrated in Figure 1. For this reason, we will sometimes refer to option trees as pseudo-ensembles.

A given OPCT is also an extension of the PCT that would be learned on the same data. By definition, whenever we introduce an option node, we include the best split (in terms of the heuristic)[3]. In regular construction of PCTs, this is the only split we consider. Consequently, the PCT is embedded in the OPCT. We can extract it if we, in a top-down fashion, select the best option in each option node.

Let's consider a full option node, that is one where we have the full 5 options. Let's assume that there are no option nodes further down in the tree. Since we have 5 options and no options lower in the tree, we have a total of 5 embedded trees. Now, let's consider the size of the embedded ensemble, when we add two such nodes under a split node, i.e., each leaf of a split node was extended into a full option node. To construct an embedded tree we can now choose one of the 5 options when the test is satisfied and one of 5 options when it is not. This results in 25 different embedded trees. It can be inferred, that to calculate the number of embedded trees for an option node we need to sum up the number of embedded trees for each of its options. In a (binary) split node, however, we multiply the numbers of embedded trees of the subtree that satisfies the split and of the subtree that does not, to obtain the total number of embedded trees.

Given the construction constraints described above, we know that option nodes with up to 5 options can appear only on the first three levels, i.e., levels 0, 1 and 2. If we now consider a full option tree, we calculate a maximum of $(5^2 \cdot 5)^2 \cdot 5 = 5^7 = 78125$ embedded trees. However, many of these trees overlap to a large extent.

Note that a given example will not traverse the entire tree. For example, in Figure 1, if an example reaches S_1 and is traversed into the left child L_1, the same result would happen in both the first and second embedded tree. The example is "agnostic" of any option nodes in the right child of S_1. Therefore, in a general option tree, a given example will visit only up to $5^3 = 125$ leaves, as it will only traverse down one side of the tree in each split node.

[3] Not only is the best split included, other splits are compared to it to determine their inclusion in the option tree.

In other words, there are a maximum of 125 different predictions that would be aggregated in order to obtain the final prediction of an option tree constructed this way. This can be compared to a single tree, where only 1 prediction will be made for each example, or to a tree ensemble, where 1 prediction would be made for each member of the ensemble and then aggregated to produce the final prediction.

[24] show that the computational complexity of building a multi-target PCT is $\mathcal{O}((S+\log N)DN\log N)$, where N is the number of examples in the data set, D is the number of descriptive variables and S is the number of target variables. Let A be the maximum number of options in an option node (in our case, we set A to 5) and B the maximum depth at which an option node can occur (in our case, we set B to 3). When constructing an OPCT, in the worst case scenario, computationally-wise, we have option nodes at all allowed levels with the maximum number of options. From that point onward we essentially construct A^B regular PCTs. Assuming that B is only a fraction of the final tree depth (i.e., $B << \log N$), we conclude that the computational complexity of building an OPCT is A^B times greater than the computational complexity of a regular PCT.

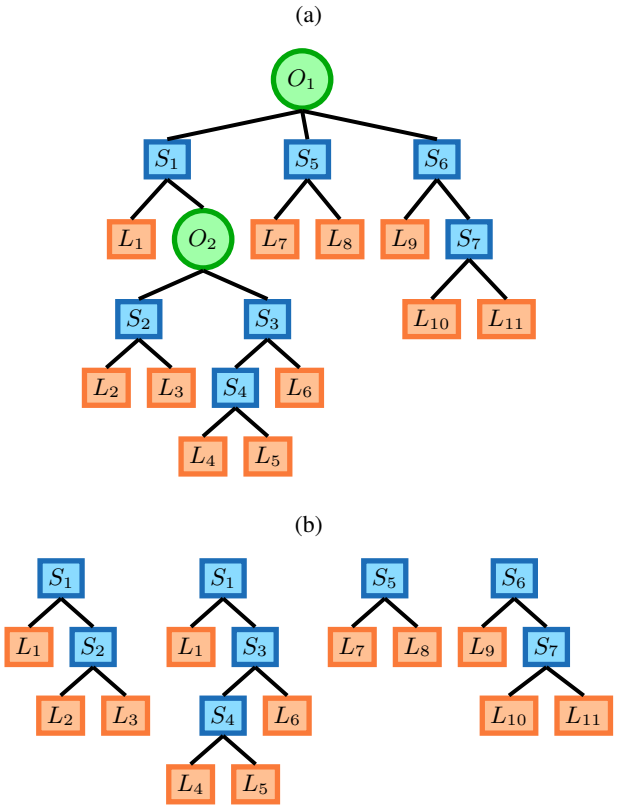


Fig. 1. An option tree (a) and the ensemble of its embedded trees (b). O_i are option nodes, S_j split nodes and L_k leaf nodes.

In comparison, a bagging ensemble constructs M independent PCTs on bootstrapped training sets, making their computational complexity M times that of a single PCT. Random forests, in addition to boostrapping, only consider $f(D)$ descriptive variables at each node. In the regular case, i.e., when $f(D) = \sqrt{D}$, this yields $\mathcal{O}((S+\log N)\sqrt{D}MN \log N)$ complexity. To summarize, OPCTs and bagging ensembles are constant factor-times slower (A^B and M respectively) than single PCTs, while random forests can be faster than a single PCT for data sets with large numbers of descriptive variables.

3. Experimental design

To evaluate the performance and efficiency of the OPCT method, we construct OPCTs with two parameter configurations, as well as standard PCTs and ensembles of PCTs. We first present the benchmark data sets used for evaluation of the methods and then give the specific experimental setup: parameter selections and evaluation measures.

3.1. Data description

The 11 data sets with multiple numeric targets used in this study come mainly from the domain of ecological modelling. Table 1 outlines the properties of the data sets. The selection contains data sets with various numbers of examples described with different numbers of attributes. For more details on the data sets, we refer the reader to the referenced literature, the repositories available at: http://mulan.sourceforge.net/datasets-mlc.html and http://kt.ijs.si/DragiKocev/PhD/resources/, as well as [24] and the references therein.

3.2. Evaluation of predictive performance and efficiency

We parameterize OPCTs by selecting values for the parameters ε and d. We consider two parameter configurations: $(\varepsilon = 1, d = 1)$ and $(\varepsilon = 0.2, d = 1)$. When $\varepsilon = 1$, there are no constraints on the heuristic value of the selected splits with regards to the best

Table 1. Properties of the data sets with multiple numeric targets (regression data sets): M is the number of instances, $|D|$ the number of descriptive attributes, and T the number of target attributes.

Name of data set	M	$\|D\|$	T
Collembola [18]	393	47	3
EDM [19]	154	16	2
Forestry-Kras [29]	60607	160	11
Forestry-Slivnica-LandSat [28]	6218	150	2
Forestry-Slivnica-IRS [28]	2731	29	2
Forestry-Slivnica-SPOT [28]	2731	49	2
Sigmea real [11]	817	4	2
Soil quality [12]	1944	142	3
Vegetation Clustering [16]	29679	65	11
Vegetation Condition [21]	16967	40	7
Water quality [14]	1060	16	14

test. However, since only the 5 best splits are selected, the risk that a split which would decrease the predictive performance would be selected is relatively low. This setting most resembles the ensemble setting, where greater variation is desired. The other parameter configuration provides a balance between predictive performance and efficiency. In our previous work [26], we used $\varepsilon = 0.5$ for the efficient version, but it turned out to be too close to the ensembles still. So in this paper, we selected $\varepsilon = 0.2$.

Next, we define the parameter values used in the algorithms for constructing single PCTs and ensembles of PCTs. The multi-target PCTs 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.

We consider two ensemble learning techniques: bagging [5] and random forests [6]. These are the most widely used tree-base ensemble learning methods. The construction of both ensemble methods takes as an input parameter the size of the ensemble, i.e., number of base predictive models to be constructed. We constructed ensembles with 100 base predictive models [24]. Furthermore, the random forests algorithm takes as input the size of the feature subset that is randomly selected at each node. For this purpose, we use the square root of the number of descriptive attributes $\lceil \sqrt{|D|} \rceil$.

We use 10-fold cross-validation to estimate the predictive performance of the used methods. We assess the predictive performance of the algorithms using several evaluation measures. In particular, since the task we consider is MTR, we employed three well known measures: the correlation coefficient (CC), mean squared error (MSE) and relative root mean squared error (RRMSE). We present here the bias-variance analysis of MSE and statistical comparison of methods according to RRMSE, where similar conclusions hold also for the other two measures. Finally, the efficiency of the proposed methods is measured with the time needed to learn a model and the size of the models (in terms of total number of leaf nodes). While OPCTs can be learned in parallel on node splitting level, our implementation does not support it yet. For this reason we used no parallelization in our experiments, to provide a fairer comparison.

In order to assess the statistical significance of the differences in performance of the studied algorithms, we adopt the recommendations by [13] for the statistical evaluation of the results. In particular, we use the Friedman test for statistical significance. Afterwards, to detect where statistically significant differences occur (i.e., between which algorithms), we use the Nemenyi post-hoc test.

We present the results of the statistical analysis with *average ranks diagrams*. The diagrams plot the average ranks of the algorithms and connect those whose average ranks differ by less than a given value, called *critical distance*. The critical distance depends on the level of the statistical significance (set in our case to 0.05). The difference in performance of the algorithms connected with a line is not statistically significant at the given significance level.

3.3. Parameter sensitivity analysis

This set of experiments is designed to provide more insight into how $varepsilon$ and d parameters affect the performance and efficiency of OPCTs. For ε we consider the values

$\{0.1, 0.2, 0.5, 1.0\}$, corresponding in order from the most stringent to the least stringent construction criterion. If we were to select $\varepsilon = 0$, the resulting OPCT would almost always directly coincide with a regular PCT, as no split would likely reach exactly the same heuristic value as the best split. Hence, the only way an option node would be induced is if two splits had the exact same heuristic value, therefore this configuration is not of interest.

As discussed above, the higher in the tree an option node is induced, the higher the variation in the learned subtrees. Induction of option nodes in lower levels of the tree not only contributes to the combinatorial explosion of the number of trees (and consequently the use of resources), but also generates less variation in the predictions, since the subtrees affected cover a smaller number of examples. Hence, we wish to curtail the number of options induced in option nodes lower in the tree. If we select a decay factor of $d = 1$, the depth of the option node induction will have no impact, while selecting a decay factor of 0.5 will effectively double the effective heuristic requirement $\varepsilon \cdot d^l$ at each level. For example, for $\varepsilon = 0.5$ and $d = 0.5$, the requirement would be 0.5 at the root level, 0.25 at the first level and 0.125 at the third level. We use the following values of the decay factor: $\{0.5, 0.9, 1\}$, these are the most to the least constrictive in terms of the construction of the OPCT.

Note that, on a given data set, all values of ε and d could produce the same OPCT, if the splits have very similar heuristic values, e.g., there could always be 5 splits that are within $10\% \cdot d^l$ of the heuristic value of the best split. Therefore, the evaluation of how ε and d affect both the predictive performance and efficiency must by design be evaluated on multiple data sets. The parameterized version of the OPCT method for a given ε and d is denoted OPCTeεdd, e.g., OPCTe0.5d0.9.

In order to facilitate replication of all the experiments, we implemented the method proposed in this paper in the latest version of CLUS, already available in the public CLUS repository at http://clus.sourceforge.net.

3.4. Bias-variance decomposition

We analyze the predictive performance of OPCTs as well as the performance of the competing methods by using bias-variance decomposition of the mean squared error [15]. The analysis allows us to investigate the source of errors of the methods. We perform it for each target separately. To calculate the decomposition, we need to predict the targets of a single data sample several times using models trained on different training sets.

Suppose we have a set of m predictive models learned using the same method on different, but related, training sets. They give m predictions for a target variable for each of the n data samples. Let y_i be the real target value of the i-th data sample and let f_{ij} be the models' prediction for y_i obtained on the j-th training set. Then the mean squared error (MSE), bias and variance can be calculated as follows:

$$\bar{f}_i = \frac{1}{m} \sum_{j=1}^{m} f_{ij}$$

$$\text{Bias} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{f}_i)^2$$

$$\text{Variance} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}(f_{ij} - \bar{f}_i)^2$$

$$\text{MSE} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}(f_{ij} - y_i)^2.$$

It is easy to show that MSE = Bias + Variance.

More complex models tend to have lower bias because they can better accommodate individual variations in the data, but this also increases their sensitivity to changes in the training set, leading to increased variance. Therefore, to minimize the error of the model a balance between bias and variance should be attained.

As mentioned earlier, the bias-variance decomposition procedure requires several predictions for each example obtained from models trained on different training sets using the same method. Standard 10-fold cross validation will therefore not suffice. To this end, after we split a data set into 10 folds, we select each of the folds for test set once and use the remaining non-test folds as a source of training sets. From this source we generate 20 training sets using bootstrapping and we learn predictive models on each bootstrap sample. The learned models are then applied to the examples from the test folds to obtain the multiple predictions for each data example.

4. Results and discussion

We discuss the results from the experimental evaluation along four major dimensions. First, we compare the performance of OPCTs with a single PCT and ensembles of PCTs. We compare them based on their predictive performance, learning time and size of the produced models. Next, present the effect of different parameter values on the construction of OPCTs. Third, we inspect the biases and variances of the algorithms to gain a better understanding of the comparisons of the predictive performance. Finally, we discuss the interpretability of OPCTs in juxtaposition to PCTs.

When analyzing the predictive power of algorithms in sections 4.2 and 4.1, ranking was done separately for every target of every data set, giving us a total of 59 samples. For efficiency we compared times needed to induce a model on entire data sets and the total sizes of these models. The Friedman test showed significant differences at $p < 1 \cdot 10^{-8}$ in all cases and the results of Nemeny post-hoc tests are presented as average ranking diagrams.

4.1. Predictive performance and efficiency

Figure 2 shows the results of the statistical evaluation of the predictive performance of the proposed OPCT method in comparison to a single PCT and ensembles of PCTs. First of all, we note that there is no statistically significant difference in the performance of bagging of PCTs, random forests of PCTs and OPCTe1d1. Notwithstanding this, random forests of PCTs have a slightly better average rank compared to the other two methods

that are very close in rank. Second, while OPCTe0.2d1 is significantly worse than the three aforementioned methods, it is still significantly better than a single PCT.

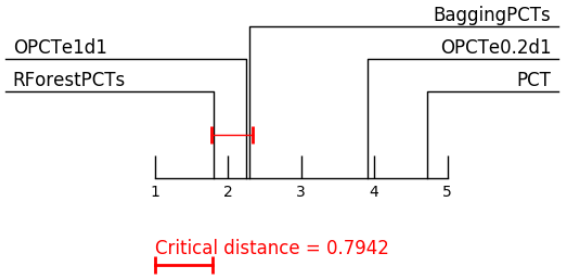
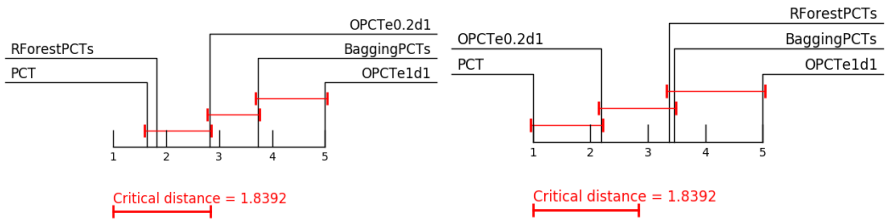


Fig. 2. Comparison of OPCTs predictive performance of OPCTs to the competing methods: Average rank diagram in terms of predictive performance.

In figure 3 we compare the methods in terms of their efficiency. It shows that learning PCTs is the most efficient method both in terms of time needed for model construction and model size, according to the average rank. However, learning OPCTe0.2d1 is not statistically significantly worse in terms of both time and size efficiency than learning a single PCT. Random forests of PCTs are close to single PCTs in terms of time, but not in terms of size, as they are significantly larger. Bagging of PCTs and OPCTe1d1 are the two slowest and largest methods.

Focusing on the efficiency of the OPCT models, we see that OPCTe1d1 is the least efficient: it produces models with the largest numbers of leaves and takes the most time for model construction. Recall that this is consistent with the analysis of computational complexity, as OPCTe1d1 can be seen as a condensed representation of what amounts to 125 PCTs, since the the algorithm selects the 5 best options at the first three levels. Furthermore, reducing the value of the ε parameter to 0.2 offers a good trade-off between predictive performance and efficiency. These models give significantly better predictive performance than a single PCT and are not significantly larger or slower to learn.



(a) Time efficiency (b) Size efficiency

Fig. 3. Comparison of the efficiency of OPCTs to that of the competing methods as measured by the time needed to learn a model and the size of the models (number of leaf nodes).

4.2. Parametrization of OPCTs

Figures 4, 5 and 6 show the average rank diagrams for predictive performance, learning time and size of different OPCTs, respectively, obtained using the experimental design outlined above. Notably, low values of both parameters lead to degradation in predictive performance. This is to be expected, since lower values of parameters force the algorithm to introduce fewer option nodes, resulting in smaller OPCTs. On the opposite side of the spectrum are the OPCTs obtained with large values of the parameters. These achieve the best predictive performance and the corresponding OPCTs are the largest in terms of the number of leaves.

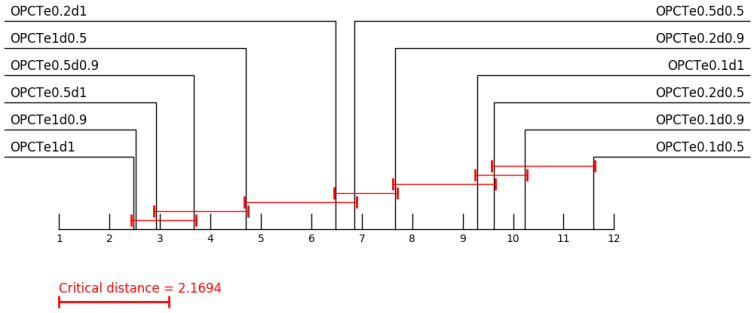


Fig. 4. Average rank diagram in terms of predictive performance for OPCTs obtained with different values of the parameters ε (e) and d.

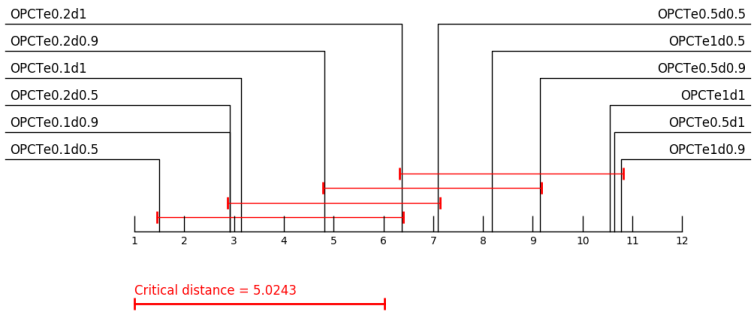


Fig. 5. Average rank diagram in terms of learning time for OPCTs obtained with different values of the parameters ε (e) and d.

Additionally, we observe that the ε parameter has a stronger influence on the performance than d. Namely, the OPCTs constructed using ε values of 0.1 and 0.2 (with the exception of OPCTe0.2d1) are the ones with the weakest predictive power. Furthermore, we also note that larger values for d also lead to better predictive performance. The best

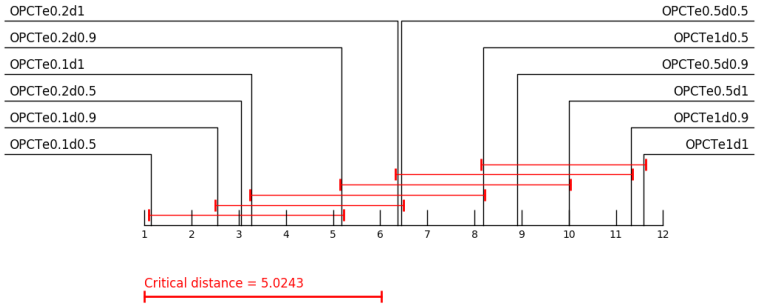


Fig. 6. Average rank diagram in terms of model size for OPCTs obtained with different values of the parameters ε (e) and d.

predictive performance is obtained when using 1 as the value of both parameters. This setting produces the largest OPCTs and requires the most learning time. Tables 2-4 in the Appendix present all values for learning times, model sizes and option node counts for different parameter values.

4.3. Bias-variance decomposition of the errors

Figures 7, 8 and 9 present the bias-variance decomposition of MSE for every target of the *Vegetation Condition*, *Sigmea real* and *Forestry-Slivnica-LandSat* data sets, respectively. The graphs for the remaining data sets are presented in Appendix.

Algorithms are presented in the following order: PCT, bagging of PCTs, random forest of PCTs, then OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1. For every target of every data set values were scaled to $[0, 1]$ by the largest MSE of all algorithms for that target, so that they can be presented on one figure. While the biases (and variances) of the different targets are presented on the same scale, only the values of different algorithms on the same target variable can be reasonably compared among each other.

The decompositions of the errors are presented on the same graph for all targets of a data set to show that they produce very similar results. That is, if a method has smaller bias (or variance) than another method on one target of a data set, it tends to have smaller bias (or variance) on the other targets of that data set as well.

The results for the *Vegetation Condition* data set given in Figure 7 represent the most common behavior. The bias of the methods is fairly similar, with less restrictive OPCTs (larger parameter values) being slightly better in this regard. PCTs have the largest variance (and consequently total error), while random forests of PCTs have the smallest. Increasing the parameter values of OPCTs also reduces the variance, bringing it on par with that of bagging and random forests.

The error decomposition for the *Sigmea real* data set given in Figure 8 stands out, since OPCTs have larger bias and variance than other algorithms for all parameter values. Increasing parameter values again reduces the variance although to a lesser extent, but bias for the second target is increased at the largest values.

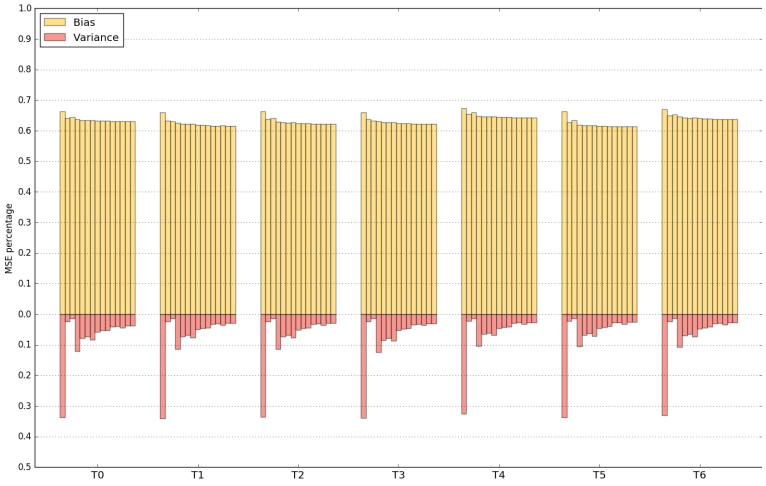


Fig. 7. Bias-variance decomposition of MSE for every target of the *Vegetation Condition* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

Fig. 8. Bias-variance decomposition of MSE for every target of the *Sigmea real* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

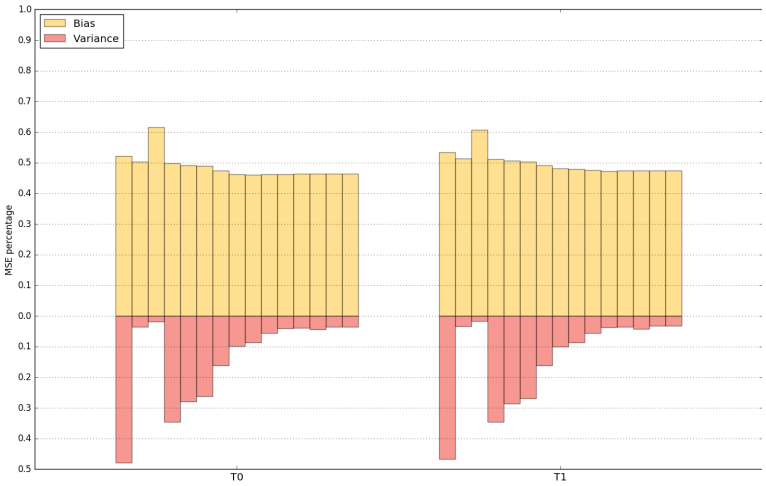


Fig. 9. Bias-variance decomposition of MSE for every target of the *Forestry-Slivnica-LandSat* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

The variance results for the *Forestry-Slivnica-LandSat* data set in Figure 9 are similar to the results for the *Vegetation condition* data set, with the notable difference that the bias of random forest of PCTs is substantially larger than the bias of the other algorithms. This occurred also in a few data sets which are shown in Appendix 5 and is not surprising, since individual trees in a random forest only use a fraction of all attributes to learn.

The results show that the difference in the predictive performance mainly arises from the difference in their variances. With a few exceptions, PCTs have the largest variance, followed by OPCTs with small parameter values. Increasing the parameter values greatly reduces the variance of OPCTs, while bagging and especially random forests have the smallest variances. The biases of different methods are generally very similar. Single PCTs and random forests have larger values on some data sets, while biases of OPCTs produced with large parameter values are occasionally smaller.

4.4. Interpretability of OPCTs

OPCTs, like option trees in general, offer a much higher degree of interpretability than ensemble methods. This is expressed through both the fact that the ”ensemble” of an option tree is represented in a compact form, i.e., a single tree, as well as the fact that many of the embedded trees overlap. Additionally, the regular PCT that would be learned from the same data is always present in the OPCT, as described in Section 2. A PCT and an OPCT learned on the *EDM* data set, and their relationship are illustrated in Figure 10.

Providing a domain expert with an option tree gives them a lot of choices with regards to the model. They can observe the selected options and attempt to determine which of the selected options were selected due to their actual importance as opposed to the sampling of the data set or other artifacts of the data. If they are able to discard all but one of the options in each of the option nodes, we can collapse the OPCT into a single tree,

specifically a PCT, which corresponds not only to the data, but also to the knowledge of the domain expert. In terms of the predictive clustering framework, this means selecting only one of the overlapping hierarchical clusters when multiple clusters are presented. This approach also has the advantage that the domain expert need not be available for interaction when the model is learned, but can assess the OPCT and chose the preferred options later on.

This process can also be looked at through a different lens. As we have introduced option trees (in part) to address myopia, by considering more options and later deciding on only one of them in each option node, we are essentially ”looking ahead” of just the one split and utilizing, in the case of the domain expert, additional domain knowledge.

However, instead of using domain knowledge by proxy of interaction with a domain expert, we could also collapse the learned OPCT to a single PCT by using additional unseen data, i.e., by calculating an unbiased estimate of the predictive performance of the different options present in the OPCT. Since the collection and preparation of additional data examples could be expensive to the point of infeasibility, we could introduce a modified experimental setup. Part of the training data could be separated into a validation set which would not be used for the initial learning of the OPCT, but would be utilized to determine which of the selected options, and consequently embedded trees, has the best predictive performance. We would then collapse the OPCT into a PCT according to this validation set, after which we would test the obtained PCT on the test set. In this scenario, we could not only study the effect of myopia by comparing the collapsed OPCT to a PCT learned on the entire training data set, but also observe the effect of averaging multiple predictions on the predictive performance by comparing the collapsed PCT and the original (pseudo-ensemble) OPCT.

5. Conclusions

In this work, we propose an algorithm for learning option predictive clustering trees (OPCTs) for the task of multi-target regression (MTR). In contrast to standard regression, where the output is a single scalar value, in MTR the output is a data structure – a tuple/vector of numeric variables. We consider learning of a global model, that is a single model that predicts all target variables simultaneously.

More specifically, we propose OPCTs to address the myopia of the standard greedy PCT learning algorithm. OPCTs have the possibility to construct option nodes, i.e., nodes with a set of alternative sub-nodes, each containing a different split. These option nodes are constructed in the cases when the heuristic scores of the candidate splits are close to each other. Furthermore, OPCTs, and option trees in general, can be regarded as a condensed representation of an ensemble of trees.

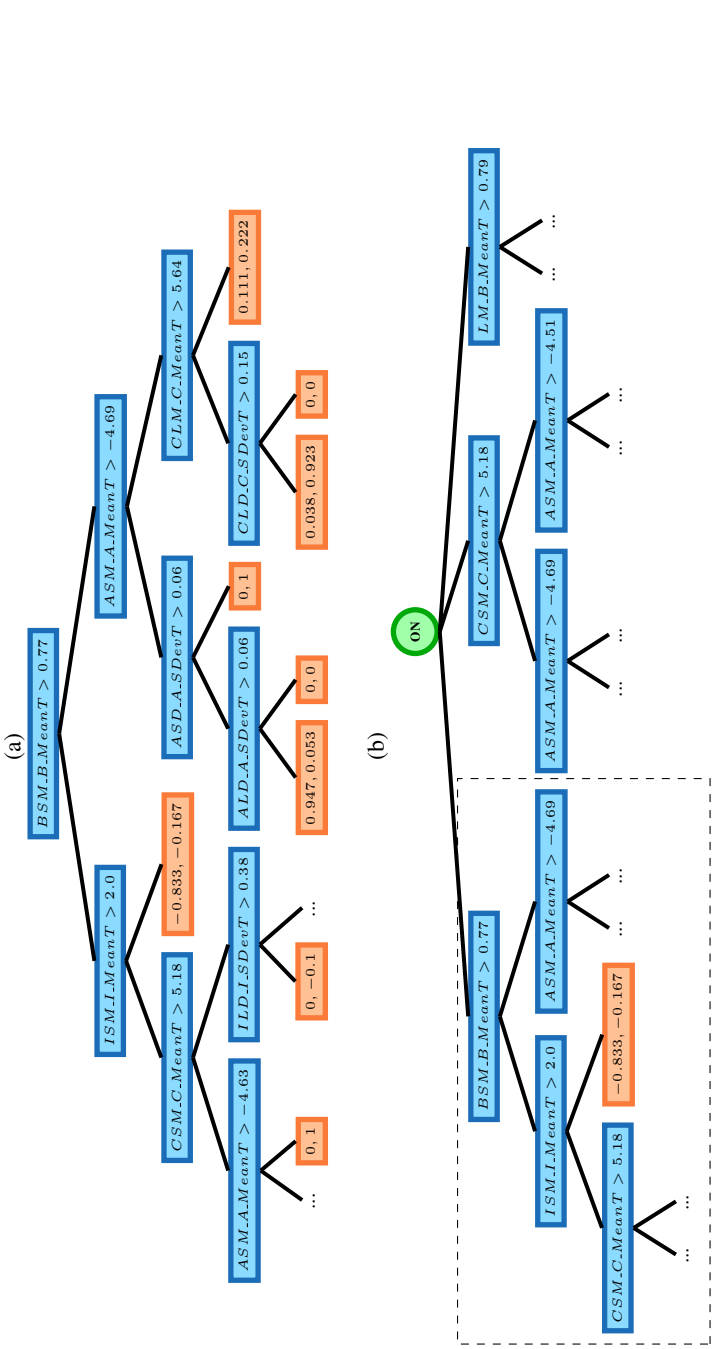


Fig. 10. A regular PCT (a) and an OPCT (b) learned on the EDM data set. The left child of a split node corresponds to the subtree where the test is satisfied. Note that the regular PCT is included in the OPCT as the subtree enclosed in the dashed rectangle.

The proposed method was experimentally evaluated on 11 benchmark MTR data sets. We selected two parameter configurations, OPCTe1d1 ($\varepsilon = 1, d = 1$), the largest OPCT which offers the best predictive performance, and OPCTe0.2d1 ($\varepsilon = 0.2, d = 1$), which provides a balance between predictive performance and efficiency. We compared this pair of OPCTs to regular PCTs and two ensemble learning methods – bagging and random forests of PCTs. The evaluation revealed that both OPCTe1d1 and OPCTe0.2d1 yield statistically significantly better predictive performance than single PCT. Next, the predictive performance of OPCTe1d1 is not statistically significantly different than that of the other two ensemble methods, however, it is slower to train and produces more leaves than the other compared methods. While the predictive performance of OPCTe0.2d1 was not on par with ensemble methods, it was not statistically significantly less efficient than PCTs, while offering significantly better performance.

We also performed parameter sensitivity analysis. The results show that both parameters that control the number of option nodes need large values to achieve good predictive performance. However, limiting the number of option nodes can lead to a more favorable trade-off between performance and efficiency.

We performed bias-variance decomposition of the mean squared error to determine the source of errors of the different methods. The results show the bias of the compared methods exhibits very similar behavior. OPCTs with larger parameter values were occasionally better in this regard, while random forests of PCTs sometimes had larger bias than other methods. On the other hand, the variance component often differed greatly between the algorithms. It was almost always the largest for single PCTs and smallest for random forests of PCTs. Increasing the parameter values for OPCTs reduced the variance and in some cases OPCTe1d1 achieved variance similar to that of the ensemble methods.

Finally, through an example, we illustrated the interpretability of the constructed OPCTs: they offer a multifaceted view on the data at hand.

We plan to extend this work along several directions. We will evaluate the OPCTs in the single tree context, i.e., we will use the induction of OPCTs as a beam-search algorithm for tree induction. Next, we will evaluate the influence of the two parameters at a more fine grained resolution. Finally, we will extend the algorithm towards other output types, i.e., machine learning tasks, such as multi-label classification, hierarchical multi-label classification and time series prediction.

Acknowledgments. We acknowledge the financial support of the European Commission through the grants ICT-2013-612944 MAESTRA and ICT-2013-604102 HBP, as well as the support of the Slovenian Research Agency through a young researcher grant (AO, TSP), the program Knowledge Technologies (P2-0103) and the project J2-9230.

References

1. Appice, A., Džeroski, S.: Stepwise induction of multi-target model trees. In: Machine Learning: ECML 2007, LNCS, vol. 4701, pp. 502–509. Springer (2007)
2. Bakır, G.H., Hofmann, T., Schölkopf, B., Smola, A.J., Taskar, B., Vishwanathan, S.V.N.: Predicting structured data. Neural Inf. Processing, The MIT Press (2007)
3. Blockeel, H., Struyf, J.: Efficient algorithms for decision tree cross-validation. Journal of Machine Learning Research 3, 621–650 (2002)

4. Borchani, H., Varando, G., Bielza, C., Larrañaga, P.: A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 5(5), 216–233 (2015)
5. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
6. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
7. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
8. Breskvar, M., Kocev, D., Džeroski, S.: Ensembles for multi-target regression with random output selections. Machine Learning 107(11), 1673–1709 (2018)
9. Buntine, W.: Learning classification trees. Statistics and Computing 2(2), 63–73 (1992)
10. Corizzo, R., Pio, G., Ceci, M., Malerba, D.: DENCAST: distributed density-based clustering for multi-target regression. Journal of Big Data 6(1), 43 (2019)
11. Demšar, D., Debeljak, M., Džeroski, S., Lavigne, C.: Modelling pollen dispersal of genetically modified oilseed rape within the field. In: The Annual Meeting of the Ecological Society of America. p. 152 (2005)
12. Demšar, D., Džeroski, S., Larsen, T., Struyf, J., Axelsen, J., Bruns-Pedersen, M., Krogh, P.H.: Using multi-objective classification to model communities of soil. Ecological Modelling 191(1), 131–143 (2006)
13. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
14. Džeroski, S., Demšar, D., Grbović, J.: Predicting chemical parameters of river water quality from bioindicator data. Applied Intelligence 13(1), 7–17 (2000)
15. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computing 4(1), 1–58 (Jan 1992)
16. Gjorgjioski, V., Džeroski, S., White, M.: Clustering analysis of vegetation data. Tech. Rep. 10065, Jožef Stefan Institute (2008)
17. Ikonomovska, E., Gama, J., Zenko, B., Dzeroski, S.: Speeding-up hoeffding-based regression trees with options. In: Proceedings of the 28th International Conference on Machine Learning, ICML 2011. pp. 537–544 (2011)
18. Kampichler, C., Džeroski, S., Wieland, R.: 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)
19. Karalič, A.: First order regression. Ph.D. thesis, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia (1995)
20. Kocev, D., Ceci, M.: Ensembles of extremely randomized trees for multi-target regression. In: Discovery Science: 18th International Conference (DS 2015). LNCS, vol. 9356, pp. 86–100 (2015)
21. Kocev, D., Džeroski, S., White, M., Newell, G., Griffioen, P.: Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. Ecological Modelling 220(8), 1159–1168 (2009)
22. Kocev, D., Struyf, J., Džeroski, S.: Beam search induction and similarity constraints for predictive clustering trees. In: Proc. of the 5th Intl Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 4747. pp. 134–151 (2007)
23. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Ensembles of multi–objective decision trees. In: ECML '07: Proceedings of the 18th European Conference on Machine Learning – LNCS 4701. pp. 624–631. Springer (2007)
24. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition 46(3), 817–833 (2013)
25. Kohavi, R., Kunz, C.: Option decision trees with majority votes. In: Proceedings of the 14th International Conference on Machine Learning. pp. 161–169. ICML '97, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997)

26. Osojnik, A., Džeroski, S., Kocev, D.: Option predictive clustering trees for multi-target regression. In: Discovery Science: 19th International Conference (DS 2016). LNCS, vol. 9956, pp. 118–133 (2016)
27. Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., Vlahavas, I.: Multi-target regression via input space expansion: Treating targets as inputs. Machine Learning 104(1), 55–98 (2016)
28. Stojanova, D.: Estimating forest properties from remotely sensed data by using machine learning. Master's thesis, Jožef Stefan IPS, Ljubljana, Slovenia (2009)
29. Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., Džeroski, S.: Estimating vegetation height and canopy cover from remotely sensed data with machine learning. Ecological Informatics 5(4), 256–266 (2010)
30. Struyf, J., Džeroski, S.: Constraint based induction of multi-objective regression trees. In: Proc. of the 4th International Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 3933. pp. 222–233. Springer (2006)
31. Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., Vlahavas, I.: Multi-target regression via random linear target combinations. In: Machine Learning and Knowledge Discovery in Databases: ECML-PKDD 2014, LNCS 8726. pp. 225–240 (2014)

Appendix

Bias-Variance Graphs

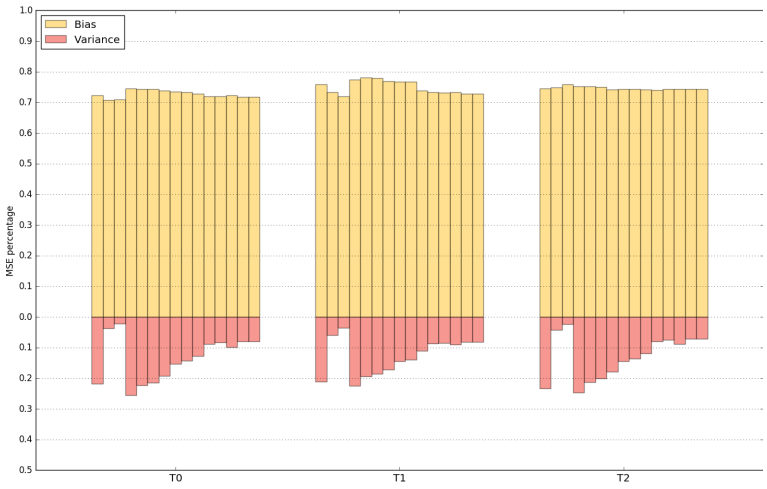


Fig. 11. Bias-variance decomposition of MSE for every target of the *Collembola* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

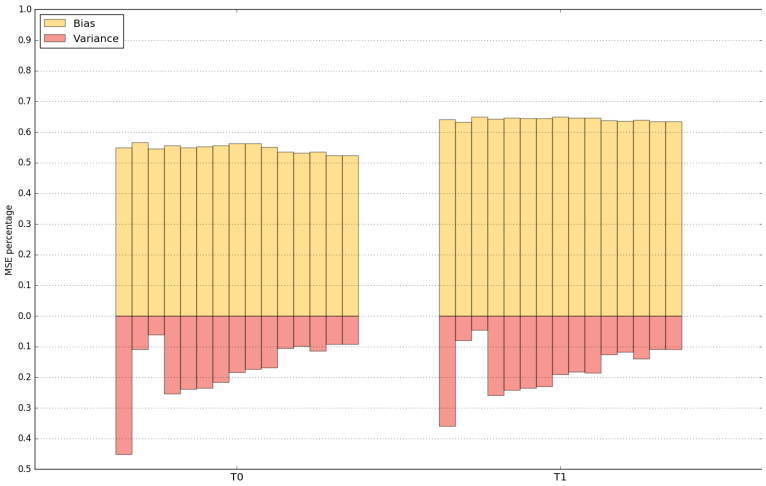


Fig. 12. Bias-variance decomposition of MSE for every target of the *EDM* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

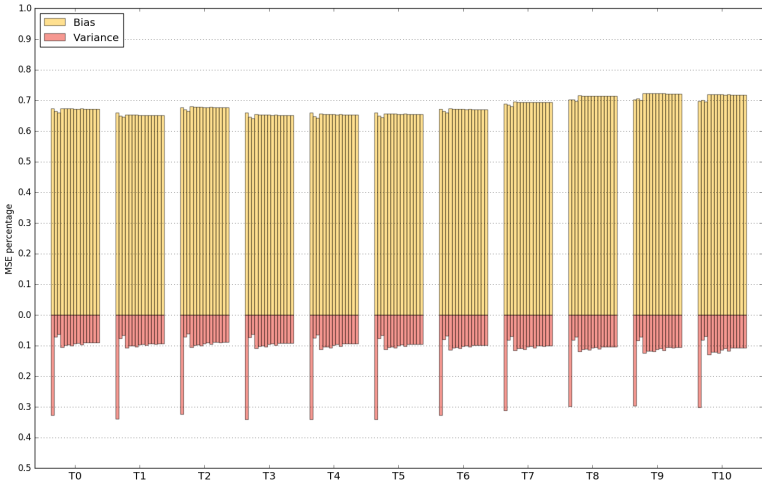


Fig. 13. Bias-variance decomposition of MSE for every target of the *Forestry-Kras* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

Additional tables

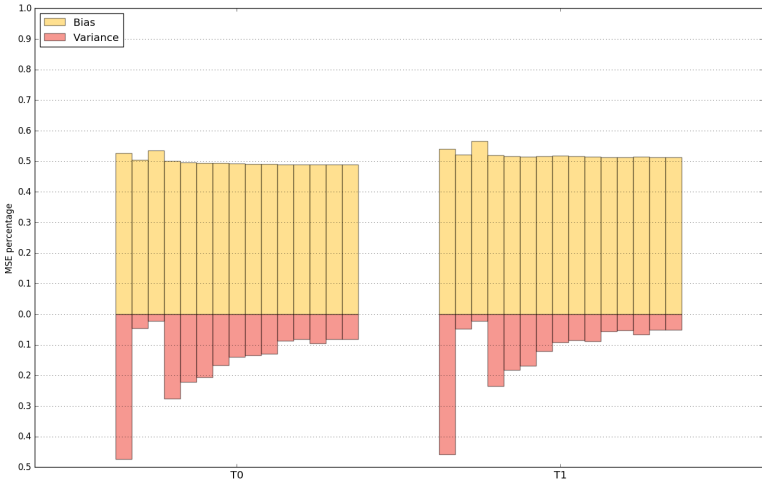


Fig. 14. Bias-variance decomposition of MSE for every target of the *Forestry-Slivnica-IRS* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

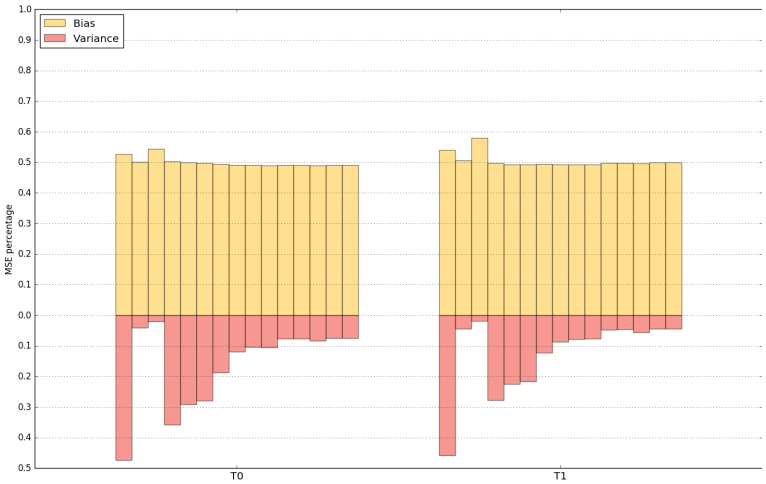


Fig. 15. Bias-variance decomposition of MSE for every target of the *Forestry-Slivnica-SPOT* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

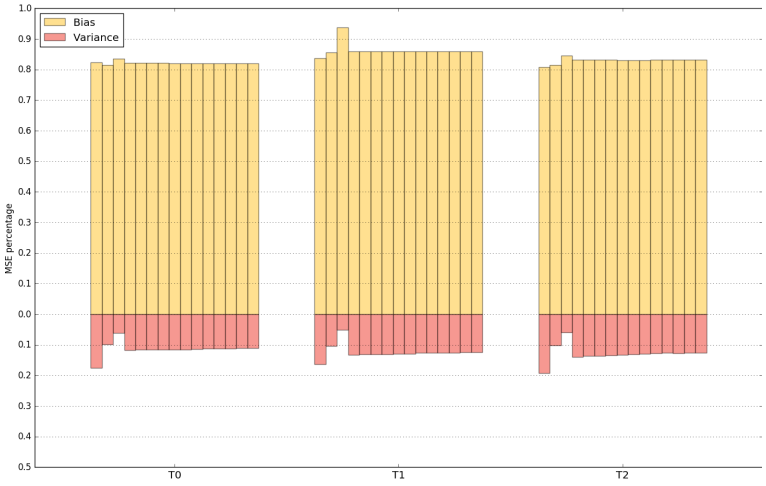


Fig. 16. Bias-variance decomposition of MSE for every target of the *Soil quality* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

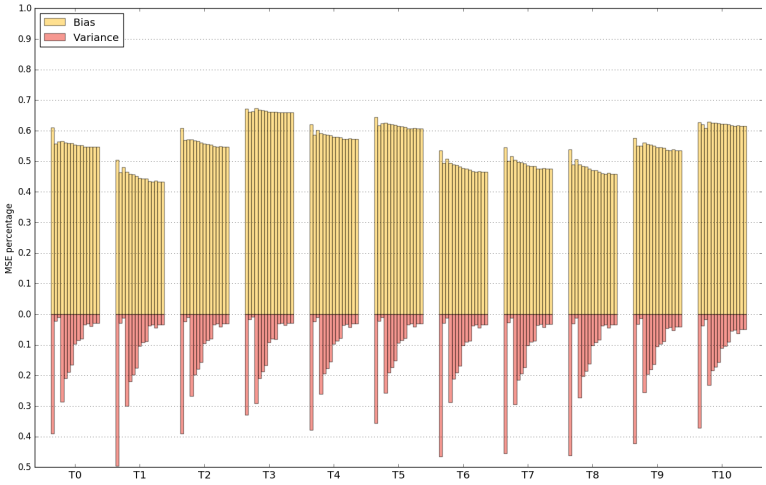


Fig. 17. Bias-variance decomposition of MSE for every target of the *Vegetation Clustering* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

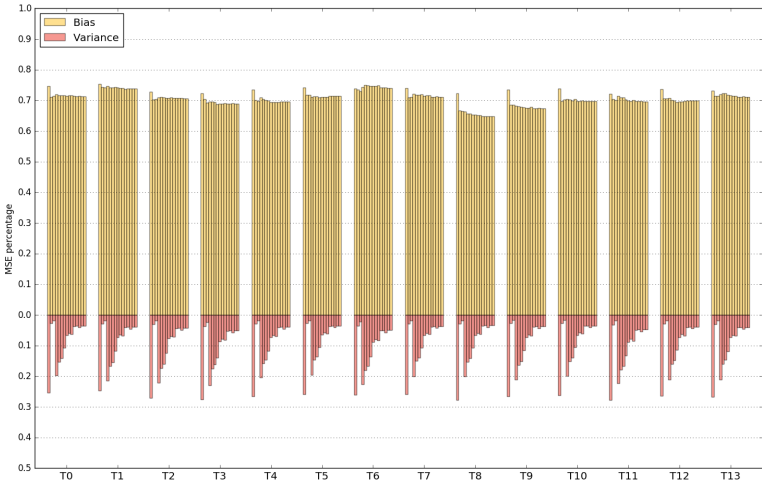


Fig. 18. Bias-variance decomposition of MSE for every target of the *Water quality* data set. Algorithms in order: PCT, bagging of PCTs, random Forest of PCTs and OPCTs with e0.1d0.5, e0.1d0.9, e0.1d1, e0.2d0.5, e0.2d0.9, e0.2d1, e0.5d0.5, e0.5d0.9, e0.5d1, e1d0.5, e1d0.9, e1d1.

	$\varepsilon = 0.1$			$\varepsilon = 0.2$			$\varepsilon = 0.5$			$\varepsilon = 1$		
	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$
collembolaV2	1689	4137	4137	4260	7513	7559	6691	14595	15836	12185	18151	18151
edm1	202	239	239	283	464	601	623	2683	3557	1882	4103	4103
Forestry_Kras	447646	830108	1002953	661259	1498692	1890340	1157139	2273266	2300007	2118682	2300007	2300007
Forestry_LIDAR_IRS	10804	18389	18921	28213	41939	45982	46523	125903	144777	91146	148377	148377
Forestry_LIDAR_Landsat	8372	31281	37406	33537	55194	70284	136907	244006	278533	200001	328002	332533
Forestry_LIDAR_Spot	4598	6932	6932	17374	58422	73037	71913	129843	143634	98279	148141	148339
sigmeareal	165	165	165	165	202	236	732	1164	1391	2484	21100	30855
soil_quality	3031	6385	8038	4617	9767	12184	23214	68372	79504	47981	104918	104918
VegetationCondition	81368	165625	192474	136516	294664	348911	368187	663801	714601	565032	761452	761452
vegetationV2	82363	86915	132580	137050	315931	344118	435360	1247710	1464212	1042378	1488376	1488376
water-quality	4009	7836	8736	6513	12895	16901	13614	51671	55516	37612	57792	57792

Table 2. Number of leaves in OPCTs with different parameter values.

	$\varepsilon = 0.1$			$\varepsilon = 0.2$			$\varepsilon = 0.5$			$\varepsilon = 1$		
	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$
collembolaV2	9	20	20	34	39	39	39	71	76	64	98	98
edm1	10	13	13	14	26	30	36	91	105	77	111	111
Forestry_Kras	28	62	80	44	95	109	87	111	111	111	111	111
Forestry_LIDAR_IRS	11	20	21	28	53	57	54	110	111	104	111	111
Forestry_LIDAR_Landsat	7	11	12	12	21	25	60	97	103	87	111	111
Forestry_LIDAR_Spot	5	5	5	12	52	67	60	107	111	96	111	111
sigmeareal	1	1	1	1	3	4	12	19	24	24	96	110
soil_quality	5	11	12	7	18	22	47	87	89	87	111	111
VegetationCondition	14	31	33	28	49	60	71	104	108	96	111	111
vegetationV2	10	12	15	18	33	35	51	95	108	94	111	111
water-quality	9	24	26	15	36	42	41	109	109	93	111	111

Table 3. Number of option nodes in OPCTs with different parameter values.

	$\varepsilon = 0.1$			$\varepsilon = 0.2$			$\varepsilon = 0.5$			$\varepsilon = 1$		
	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$	$d = 0.5$	$d = 0.9$	$d = 1$
collembolaV2	0.68	1.17	1.16	1.12	1.71	1.75	1.56	3.07	3.23	8.39	11.83	6.23
edm1	0.22	0.23	0.22	0.23	0.23	0.23	0.24	0.34	0.41	0.3	0.44	0.42
Forestry_Kras	449.71	781.2	942.74	653.2	1380.88	1723.76	1065.94	2083.34	2117.13	1920.28	2069.53	2110.88
Forestry_LIDAR_IRS	1.13	1.89	1.83	2.64	3.7	3.82	7.89	10.02	12.8	7.29	11.58	11.39
Forestry_LIDAR_Landsat	6.04	13.55	15.25	14.15	22.11	31.03	51.93	92.44	104.78	74.78	121.95	124.78
Forestry_LIDAR_Spot	0.9	1.19	1.18	2.59	11.17	10.07	11.84	18.36	19.03	13.03	18.37	18.75
sigmeareal	0.58	0.52	0.49	0.44	0.35	0.61	7.1	5.99	6.09	6.05	5.19	1.12
soil_quality	1.0	1.5	1.68	1.25	2.0	12.55	12.02	19.25	19.35	10.47	21.25	20.01
VegetationCondition	20.15	40.44	41.29	30.65	64.23	75.09	80.19	144.58	154.24	121.42	174.21	166.67
vegetationV2	24.45	24.55	37.32	39.35	84.73	96.15	116.34	324.78	381.27	273.54	373.55	375.06
water-quality	0.57	0.74	0.76	0.65	0.98	1.22	1.02	5.83	5.73	2.31	11.96	6.07

Table 4. Learning times for OPCTs with different parameter values.

Tomaž Stepišnik is a PhD student at the Jožef Stefan International Postgraduate School and a young researcher at the Department of Knowledge Technologies, Jožef Stefan Institute. He completed his MSc studies in mathematics at the University of Ljubljana in 2016 and was awarded a Young researcher grant by the Slovenian national research agency. His research interests include the study and development of machine learning algorithms and their application. He was on a research visit at the University of Auckland, NZ in 2018 for 3 months. He reviews for various conferences on the topics of Machine Learning and Data Mining (KDD, DS, ECML PKDD, ...).

Aljaž Osojnik is a researcher at the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia. He completed his MSc and BSc in mathematics at the Faculty of Mathematics and Physics, University of Ljubljana, Slovenia. He completed his PhD in 2017 at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia on the topic of structured output prediction on data streams. His research interests are related to online learning, learning from data streams and privacy-preserving data mining.

Sašo Džeroski received his Ph.D. degree in computer science from the University of Ljubljana in 1995. He is Head of the Department of Knowledge Technologies at the Jožef Stefan Institute and full professor at the JS International Postgraduate School (both in Ljubljana, Slovenia). He is also a visiting professor at the Phi-Lab of ESRIN, European Space Agency (Frascati, Italy). His research interests fall in the field of artificial intelligence (AI), focusing on the development of data mining and machine learning methods for a variety of tasks - including the prediction of structured outputs and the automated modeling of dynamical systems - and their applications to practical problems. In 2008, he was elected fellow of the European AI Society for his ”Pioneering Work in the field of AI and Outstanding Service for the European AI community”. In 2015, he became a foreign member of the Macedonian Academy of Sciences and Arts. In 2016, he was elected a member of Academia Europaea. He is currently serving as Chair of the Slovenian AI Society.

Dragi Kocev is a researcher at the Department of Knowledge Technologies, JSI. He completed his PhD in 2011 at the JSI Postgraduate School in Ljubljana on the topic of learning ensemble models for predicting structured outputs. He was a visiting research fellow at the University of Bari, Italy in 2014/2015. He has participated in several national Slovenian projects, the EU funded projects IQ, PHAGOSYS and HBP. He was co-coordinator of the FP7 FET Open project MAESTRA.

Received: September 28, 2019; Accepted: May 20, 2020.