

Option Predictive Clustering Trees for Multi-target Regression

Aljaž Osojnik^{1,2(✉)}, Sašo Džeroski^{1,2}, and Dragi Kocev^{1,2}

¹ Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
{aljaz.osojnik,saso.dzeroski,dragi.kocev}@ijs.si

² International Postgraduate School, Jožef Stefan Institute, Ljubljana, Slovenia

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 extend predictive clustering trees for multi-target regression by considering option nodes, i.e., learn option predictive clustering trees (OPCTs). Multi-target regression is concerned with learning predictive models for tasks with multiple continuous target variables. We evaluate the proposed OPCTs on 11 benchmark MTR datasets. The results reveal that OPCTs achieve statistically significantly better predictive performance than a single PCT. Next, the performance is competitive with that of bagging and random forests of PCTs. Finally, we demonstrate the potential of OPCTs for multifaceted interpretability and illustrate the potential of inclusion of domain knowledge in the tree learning process.

Keywords: Multi-target regression · Option trees · Interpretable models · Predictive clustering trees

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 (i.e., the target) is structured, e.g., is a vector of class values of a tuple of target variables. There can be dependencies between the class values/targets (e.g., they can be organized into a tree-shaped hierarchy or a directed acyclic graph) or some internal relations between the class values (e.g., as in sequences).

In this work, we concentrate on the task of predicting multiple continuous variables. Examples thus take the form $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i = (x_{i1}, \dots, x_{ik})$ is a vector of k input variables and $\mathbf{y}_i = (y_{i1}, \dots, y_{it})$ is a vector of t target variables. This task is known under the name of *multi-target regression* (MTR) [1] (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 continuous variables. Prominent examples come from ecology and include predicting the abundance of different species living in the same habitat [2] and predicting properties of forests [3,4]. 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 [1,5]. These methods can be categorized into two groups of methods [6]: (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 [1,4,7–9]. The global methods have several advantages over the local methods. First, they exploit and use the dependencies that exist between the components of the structured output in the model learning phase, which can result in better predictive performance. Next, they are typically more efficient: it can easily happen that the number of components in the output is very large (e.g., 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 [1,5]. Trees for MTR (from the predictive clustering framework) inherit the properties of regression trees: they are interpretable models, but the learning them is greedy. The performance of the trees is significantly improved when they are used in an ensemble setting [1,7]. 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 [10], while another approach is to introduce option splits in the nodes [11,12].

In this work, we propose to extend predictive clustering trees (PCTs) for MTR towards option trees, hence we propose to learn option predictive clustering

trees (OPCTs). An option tree can be seen as a condensed representation of an ensemble of trees which share 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 Fig. 1.

The remainder of this paper is organized as follows. Section 2 proposes the algorithm for learning option PCTs for MTR. Next, Sect. 3 outlines the design of the experimental evaluation. Section 4 continues with a discussion of the results. Finally, Sect. 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 [13], which is available for download at <http://clus.sourceforge.net>.

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 Buntine [11] and then analyzed in more detail by Kohavi and Kunz [12]. Ikononovska et al. [14] 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 [15]. 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 splitting a leaf, 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 split could be selected over 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 [12].

An option node is introduced into the tree when it would be hard to determine the best split, i.e., 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

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

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. 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, vbut, uses exponentially more resources and is not advised [12].

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

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

Procedure OptionPCT

Input: A dataset 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
       $(t^*, h^*, \mathcal{P}^*) = candidates[0]$ 
      for each  $E_i \in \mathcal{P}^*$  do
         $tree_i = \text{OptionPCT}(E_i, \varepsilon, d, l + 1)$ 
      return node( $t^*$ ,  $\bigcup_i \{tree_i\}$ )
    else
       $(t_0^*, h_0^*, \mathcal{P}_0^*) = candidates[0]$ 
      nodes = {}
      for each  $(t_i^*, h_i^*, \mathcal{P}_i^*) \in candidates$  do
        if  $\frac{h_i^*}{h_0^*} \geq 1 - \varepsilon \cdot d^l$  then
          for each  $E_j \in \mathcal{P}_i^*$  do
             $tree_j = \text{OptionPCT}(E_j, \varepsilon, d, l + 1)$ 
            nodes = nodes  $\cup$  {node( $t^*$ ,  $\bigcup_j \{tree_j\}$ )}
      if  $|nodes| > 1$  then
        return option_node(nodes)
      else
        return nodes[0]
  else
    return leaf(Prototype( $E$ ))

```

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 would make 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.

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 options whose heuristics are considerably worse than the heuristic of the best split, might not necessarily reduce the performance of the tree, but actually increase it.

Once an OPCT is built, we want to use it for prediction. If we reiterate the methodology described above in a tree-prediction setting, we could say the following. In a regular PCT, it is simple to produce a prediction for a new example. It is sorted into a leaf (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 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 of the generated trees until we are left with no option nodes. This is illustrated in Fig. 1. For this reason, we will sometimes refer to option trees as pseudo-ensembles.

On the other hand, a given OPCT is a direct 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)¹. In regular construction of PCTs, this is the only split we consider. Consequently, the PCT is embedded in the OPCT. Specifically, we can extract it, if we, in a top-down fashion, only select the best option in each option node.

Let’s consider a full option node, i.e., 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

¹ Not only is the best split included, other splits are compared to it to determine their inclusion in the option tree.

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 easily be inferred, that to calculate the number of embedded trees for an option node we need just 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 doesn't, 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 Fig. 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.

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.

3 Experimental Design

To evaluate the performance and efficiency of the OPCT method, we construct OPCTs by using different values for the algorithms' parameters, as well as standard PCTs and ensembles of PCTs. We first present the benchmark datasets used for evaluation of the methods and then give the specific experimental setup: parameter instantiations and evaluation measures.

3.1 Data Description

The 11 datasets with multiple continuous targets used in this study come mainly from the domain of ecological modelling. Table 1 outlines the properties of the datasets. This selection contains datasets with various number of examples described with different numbers of attributes. For more details on the datasets, we refer the reader to the referenced literature.

3.2 Experimental Setup

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 that of MTR, we employed three well known measures: the correlation coefficient (CC), root mean squared error ($RMSE$) and relative root mean squared error ($RRMSE$). We present here only the results in terms of $RRMSE$, but similar conclusions hold for the other two measures. Finally, the efficiency of the proposed methods is measured with the time needed to construct a model and the size of the models (in terms of total number of leaf nodes available for making a prediction).

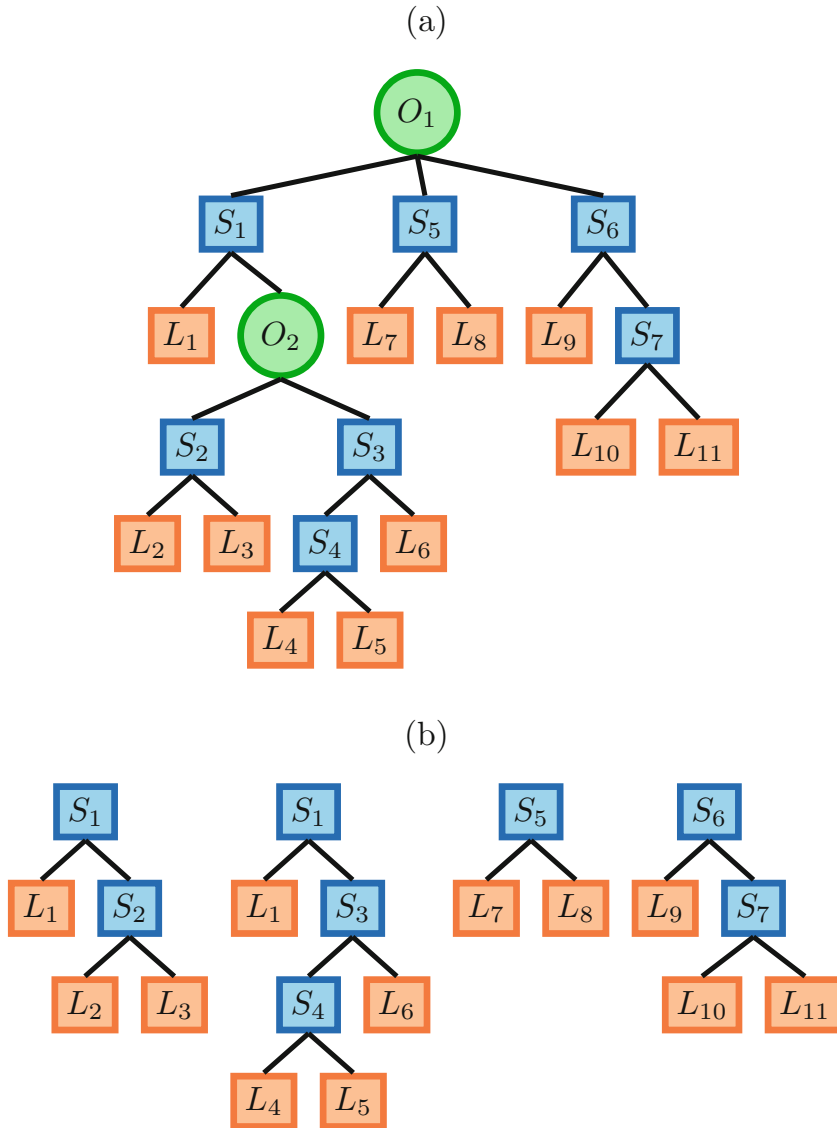


Fig. 1. An option tree (a) and the ensemble of its embedded trees (b). O_i are option nodes, S_j split nodes and L_k leaf nodes.

We parameterize OPCTs by selecting values for the parameters ε and d . When $\varepsilon = 1$, there are no constraints on the heuristic value of the selected splits with regards to the best 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. 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. This configuration is not of interest, so 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.

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.

Table 1. Properties of the datasets with multiple continuous targets (regression datasets): N is the number of instances, $|D|/|C|$ the number of descriptive attributes (discrete/continuous), and T the number of target attributes.

Name of dataset	N	$ D / C $	T
Collembola [16]	393	8/39	3
EDM [17]	154	0/16	2
Forestry-Kras [3]	60607	0/160	11
Forestry-Slivnica-LandSat [18]	6218	0/150	2
Forestry-Slivnica-IRS [18]	2731	0/29	2
Forestry-Slivnica-SPOT [18]	2731	0/49	2
Sigma real [19]	817	0/4	2
Soil quality [2]	1944	0/142	3
Vegetation clustering [20]	29679	0/65	11
Vegetation condition [4]	16967	1/39	7
Water quality [21]	1060	0/16	14

Note that, on a given dataset, all of the 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 datasets. The parameterized version of the OPCT method for a given ε and d is denoted $\text{OPCT}_{\varepsilon d}$, e.g., $\text{OPCT}_{0.5d0.9}$.

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 [22] and random forests [23]. 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 [1]. 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 apply the logarithmic function of the number of descriptive attributes $\lfloor \log_2 |D| \rfloor + 1$, as recommended by Breiman [23].

In order to assess the statistical significance of the differences in performance of the studied algorithms, we adopt the recommendations by Demšar [24] 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 to compare all the methods among each other.

We present the results from 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.

4 Results and Discussion

We discuss the results from the experimental evaluation along three major dimensions. First, we select the optimal parameter values for the construction of OPCTs. Second, we compare the performance of OPCTs with a single PCT and ensembles of PCTs. The comparison is performed on their predictive power, time consumption and size of the produced models. Third, we examine the interpretability of OPCTs in juxtaposition to PCTs.

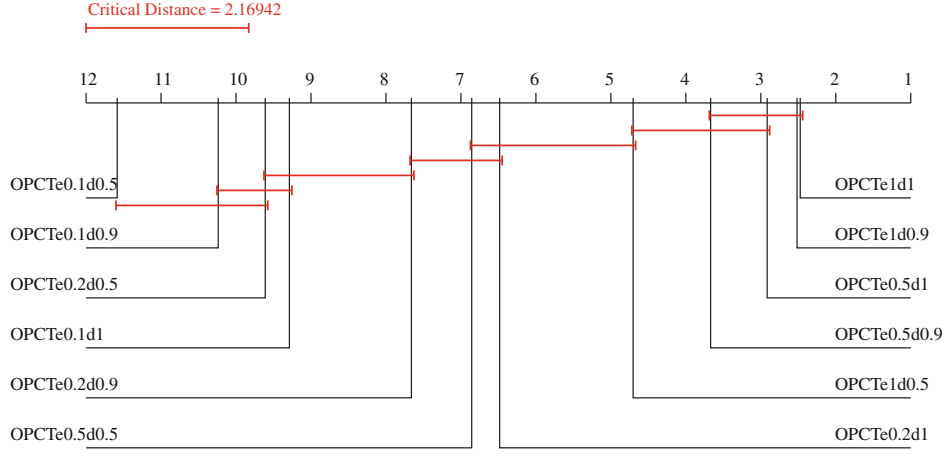


Fig. 2. Average rank diagram (in terms of predictive performance) for OPCTs obtained with different values of the parameters ε (e) and d .

4.1 Parametrization of OPCTs

In Fig. 2, we depict the performance of the different OPCTs obtained by using the experimental design outlined above. We can note that lower values for both the parameters leads to performance degradation. This is somewhat expected, because by imposing more stringent values, we are forcing the algorithm not to introduce option nodes, hence the obtained OPCTs are small. On the opposite side of the spectrum, we have the OPCTs obtained with larger values for the parameters. These achieve the best predictive performance and the corresponding OPCTs are large.

We can also observe that the ε parameter has a stronger influence on the performance than d . Namely, the OPCTs constructed with selecting 0.1 and 0.2 as values for ε (no matter the value of d) are the ones with the weakest predictive power. Furthermore, we can also note that the larger values for d also lead to better predictive performance. The best predictive performance is obtained when setting both parameters to 1. Such a setting produces large OPCTs and requires the most time to construct the OPCTs. All in all, we can recommend the use of two instantiations of the OPCT algorithm: OPCTe1d1 ($\varepsilon = 1, d = 1$) and OPCTe0.5d1 ($\varepsilon = 0.5, d = 1$). These two variants represent the parameters for the best performance and the trade-off between predictive power and efficiency, respectively.

4.2 Predictive Performance and Efficiency

Figure 3 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. We can first note that both OPCTs (OPCTe1d1 and OPCTe0.5d1) achieve statistically significantly better predictive performance than the single PCT. Second, there is no statistically significant difference in the performance of the ensemble methods and the OPCTs. Furthermore, the two ensemble methods are bounded by the two OPCT variants: the best performing method is the

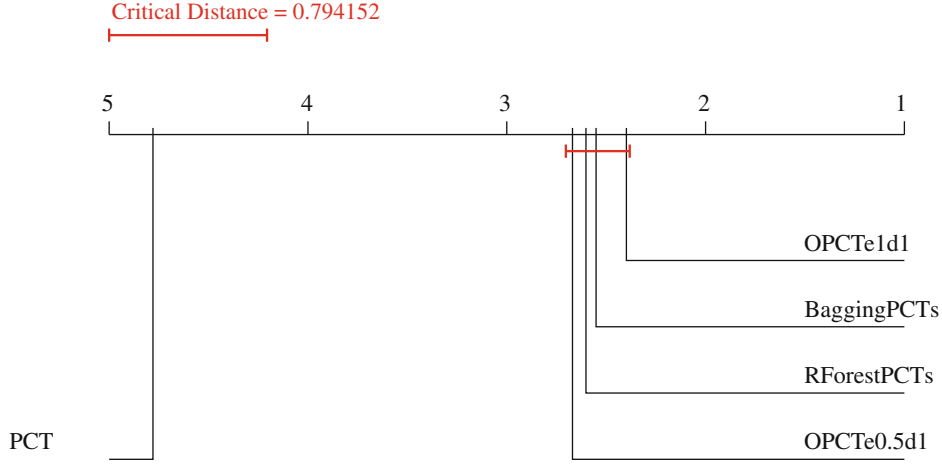


Fig. 3. Comparison of OPCTs predictive performance of OPCTs to the competing methods: average rank diagram in terms of predictive performance.

OPCTe1d1, followed by Bagging of PCTs, Random forests of PCTs, and the OPCTe0.5d1 method.

Next, we compare the methods in terms of their efficiency. First of all, learning PCTs is the most efficient method both in terms of time needed for model construction and model size. More specifically, single PCT models are statistically significantly smaller than all of the other models and are obtained statistically significantly faster than Bagging of PCTs and OPCTe1d1.

If we focus on the efficiency of the OPCT models, OPCTe1d1 is the least efficient: it produces models with largest numbers of leaves and takes the most time for model construction. Recall that this is due to the fact that, in the worst case, OPCTe1d1 is actually a condensed representation of what amounts to 125 PCTs. We also note that reducing the value of the ε parameter to 0.5 yields smaller models that are constructed faster than both bagging and random forests of PCTs. All in all, OPCTe0.5d1 offers the best trade-off between predictive performance and efficiency and should be considered for further use. However, if a given task requires the best predictive performance then one should use OPCTe1d1 (Fig. 4).

4.3 Interpretability of OPCTs

OPCTs, as well as 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 Sect. 2. A PCT and an OPCT learned on the EDM dataset, and their relations are illustrated in Fig. 5.

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

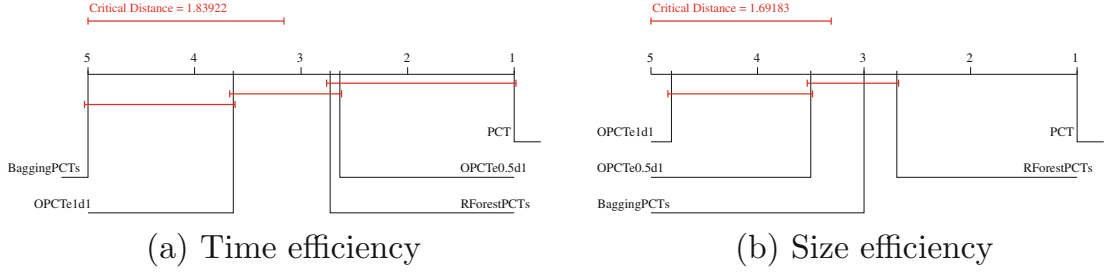


Fig. 4. Comparison of the efficiency of OPCTs to that of the competing methods as measured by the time needed to learn a model and the size of the models (number of leaf nodes).

importance as opposed to the sampling of the dataset 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 dataset, but also observe the effect of averaging multiple predictions on the predictive performance by comparing the collapsed PCT and the original (pseudo-ensemble) OPCT.

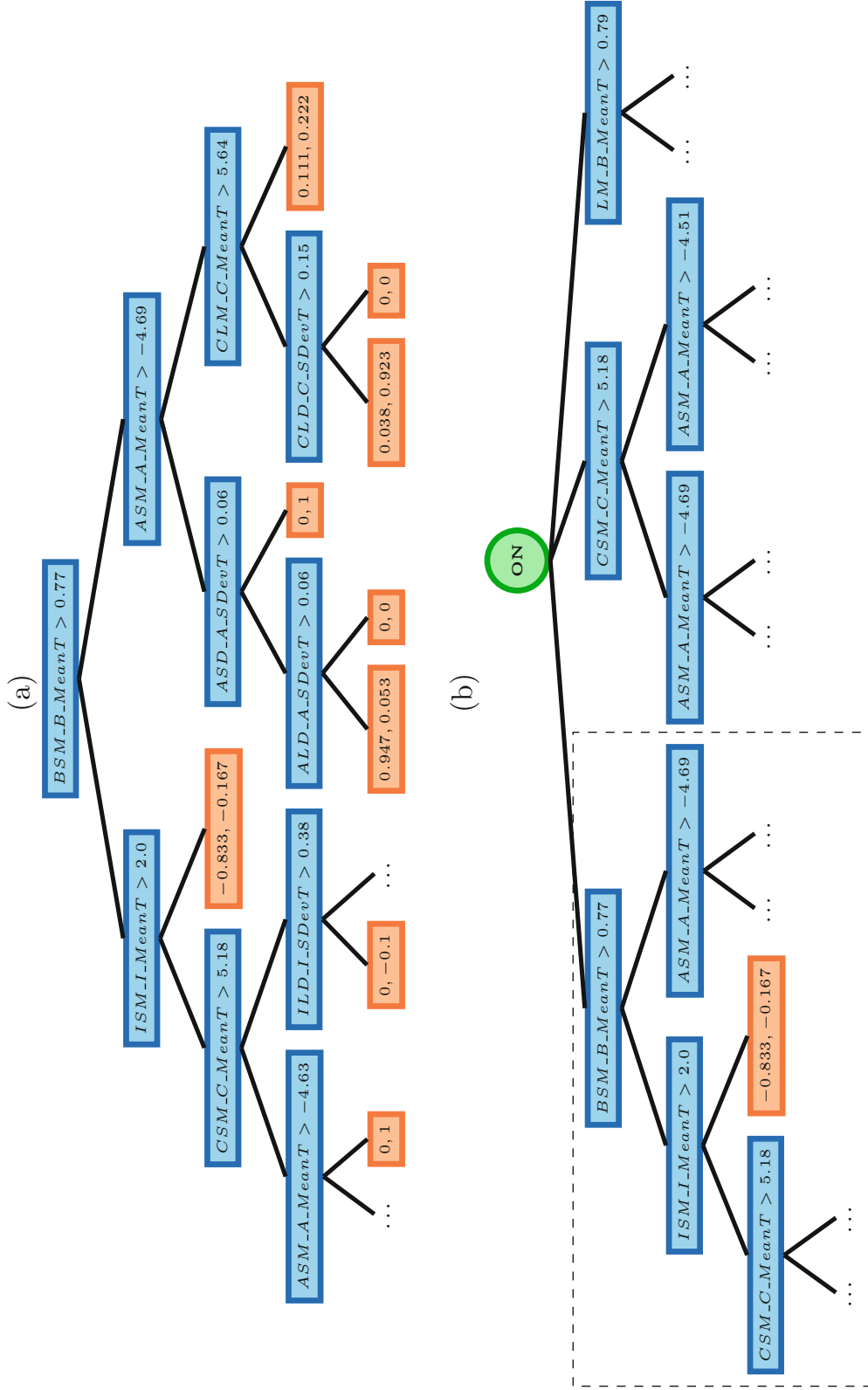


Fig. 5. A regular PCT (a) and an OPCT (b) learned on the EDM dataset. The left child of a split node corresponds to the subtree where the test is satisfied. Note that the regular PCT is included in the OPCT as the subtree enclosed in the dashed rectangle.

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 continuous variable values. We consider learning of a global model, i.e., of a single model that predicts all of the 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.

The proposed method was experimentally evaluated on 11 benchmark MTR datasets. We first determined the optimal parameters of the algorithm. The results show that both parameters that control the number of option nodes (the range of heuristic scores considered ε and the decay factor d) need larger values to achieve better predictive performance. We then compared the performance of learning OPCTs (in terms of predictive power and efficiency) to the induction of standard PCTs and to two ensemble learning methods, i.e., bagging and random forests of PCTs.

The evaluation revealed that OPCTs yield statistically significantly better predictive performance than a single PCT. Next, the predictive performance of the OPCTs is not statistically significantly different than that of the other two ensemble methods, but OPCT with ε and d set to 1 achieves the best predictive performance on average. Moreover, in terms of efficiency, an OPCT with ε set to 0.5 and d set to 1 is faster to learn than the ensemble models and the OPCT with ε and d set to 1. 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. First of all, 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 and the program Knowledge Technologies (P2-0103).

References

1. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. *Pattern Recogn.* **46**(3), 817–833 (2013)
2. 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. *Ecol. Model.* **191**(1), 131–143 (2006)
3. Stojanova, D., Panov, P., Gjorgjoski, V., Kobler, A., Džeroski, S.: Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecol. Inform.* **5**(4), 256–266 (2010)
4. 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. *Ecol. Model.* **220**(8), 1159–1168 (2009)
5. Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., Vlahavas, I.: Multi-target regression via random linear target combinations. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) *ECML PKDD 2014. LNCS (LNAI)*, vol. 8726, pp. 225–240. Springer, Heidelberg (2014). doi:[10.1007/978-3-662-44845-8_15](https://doi.org/10.1007/978-3-662-44845-8_15)
6. Bakır, G.H., Hofmann, T., Schölkopf, B., Smola, A.J., Taskar, B., Vishwanathan, S.V.N.: *Predicting Structured Data. Neural Information Processing. The MIT Press*, Cambridge (2007)
7. Kocev, D., Ceci, M.: Ensembles of extremely randomized trees for multi-target regression. In: Japkowicz, N., Matwin, S. (eds.) *DS 2015. LNCS (LNAI)*, vol. 9356, pp. 86–100. Springer, Heidelberg (2015). doi:[10.1007/978-3-319-24282-8_9](https://doi.org/10.1007/978-3-319-24282-8_9)
8. Struyf, J., Džeroski, S.: Constraint based induction of multi-objective regression trees. In: Džeroski, S., Struyf, J. (eds.) *KDID 2006. LNCS*, vol. 3933, pp. 222–233. Springer, Heidelberg (2006). doi:[10.1007/11733492_13](https://doi.org/10.1007/11733492_13)
9. Appice, A., Džeroski, S.: Stepwise induction of multi-target model trees. In: Kok, J.N., Koronacki, J., Mantaras, R.L., Matwin, S., Mladenič, D., Skowron, A. (eds.) *ECML 2007. LNCS (LNAI)*, vol. 4701, pp. 502–509. Springer, Heidelberg (2007). doi:[10.1007/978-3-540-74958-5_46](https://doi.org/10.1007/978-3-540-74958-5_46)
10. Kocev, D., Struyf, J., Džeroski, S.: Beam search induction and similarity constraints for predictive clustering trees. In: Džeroski, S., Struyf, J. (eds.) *KDID 2006. LNCS*, vol. 4747, pp. 134–151. Springer, Heidelberg (2007). doi:[10.1007/978-3-540-75549-4_9](https://doi.org/10.1007/978-3-540-75549-4_9)
11. Buntine, W.: Learning classification trees. *Stat. Comput.* **2**(2), 63–73 (1992)
12. Kohavi, R., Kunz, C.: Option decision trees with majority votes. In: *Proceedings of the 14th International Conference on Machine Learning, ICML 1997*, pp. 161–169. Morgan Kaufmann Publishers Inc., San Francisco (1997)
13. Blockeel, H., Struyf, J.: Efficient algorithms for decision tree cross-validation. *J. Mach. Learn. Res.* **3**, 621–650 (2002)
14. Ikonomovska, E., Gama, J., Ženko, B., Džeroski, 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)
15. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: *Classification and Regression Trees*. Chapman & Hall/CRC, Boca Raton (1984)
16. 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 Biol. Biochem.* **32**(2), 197–209 (2000)

17. Karalič, A.: First order regression. Ph.D. thesis, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia (1995)
18. Stojanova, D.: Estimating forest properties from remotely sensed data by using machine learning. Master's thesis, Jožef Stefan IPS, Ljubljana, Slovenia (2009)
19. 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)
20. Gjorgjioski, V., Džeroski, S., White, M.: Clustering analysis of vegetation data. Technical report 10065, Jožef Stefan Institute (2008)
21. Džeroski, S., Demšar, D., Grbović, J.: Predicting chemical parameters of river water quality from bioindicator data. *Appl. Intell.* **13**(1), 7–17 (2000)
22. Breiman, L.: Bagging predictors. *Mach. Learn.* **24**(2), 123–140 (1996)
23. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
24. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)