

# Option Predictive Clustering Trees for Hierarchical Multi-label Classification

Tomaž Stepišnik Perdih<sup>1,2(✉)</sup>, Aljaž Osojnik<sup>1,2</sup>, Sašo Džeroski<sup>1,2</sup>,  
and Dragi Kocev<sup>1,2</sup>

<sup>1</sup> Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia  
{tomaz.stepisnik, aljaz.osojnik, saso.dzeroski, dragi.kocev}@ijs.si

<sup>2</sup> Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

**Abstract.** In this work, we address the task of hierarchical multi-label classification (HMLC). HMLC is a variant of classification, where a single example may belong to multiple classes at the same time and the classes are organized in the form of a hierarchy. Many practically relevant problems can be presented as a HMLC task, such as predicting gene function, habitat modelling, annotation of images and videos, etc. We propose to extend the predictive clustering trees for HMLC – a generalization of decision trees for HMLC – toward learning option predictive clustering trees (OPCTs) for HMLC. OPCTs address the myopia of the standard tree induction by considering alternative splits in the internal nodes of the tree. An option tree can also be regarded as a condensed representation of an ensemble. We evaluate OPCTs on 12 benchmark HMLC datasets from various domains. With the least restrictive parameter values, OPCTs are comparable to the state-of-the-art ensemble methods of bagging and random forest of PCTs. Moreover, OPCTs statistically significantly outperform PCTs.

## 1 Introduction

Supervised learning is one of the most widely researched areas of machine learning, where the goal is to learn, from a set of examples with known class, a function that outputs a prediction for the class of a previously unseen example. The most widely studied machine learning task is binary classification where the goal is to classify the examples into two groups. The task where the examples can belong to a single class from a given set of  $m$  classes ( $m \geq 3$ ) is known as multi-class classification. The case where the output is a real value is called regression.

In many real life problems of predictive modelling the target is structured (e.g., the target is a vector of values with dependencies between them, or a time series). In this work, we focus on the task of hierarchical multi-label classification (HMLC). HMLC is a variant of classification, where a single example may belong to multiple classes at the same time and the classes are organized in the form of a hierarchy. An example that belongs to some class  $c$  automatically belongs to all super-classes of  $c$ : This is called the hierarchical constraint. Problems of this kind can be found in many domains including text classification, functional genomics,

and object/scene classification. Silla and Freitas [19] give a detailed overview of the possible application areas and the different approaches to HMLC.

Decision tree based methods take a very notable place among approaches to HMLC. When used as base predictive models in an ensemble, they can yield a state-of-the-art performance [13, 18]. A prominent global tree method for HMLC is a predictive clustering tree (PCT) for HMLC [20]. PCTs for HMLC inherit the properties of decision trees: they are interpretable models, but learning them is greedy. The performance of the trees is significantly improved when they are used in an ensemble setting [13]. However, the 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 [12], while another approach is to introduce option splits in the nodes [5, 14].

In this work, we propose to extend predictive clustering trees (PCTs) for HMLC 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 HMLC. 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 [1], which is available at <http://clus.sourceforge.net>.

Option predictive clustering trees (OPCT) extend the usual PCT framework, by introducing option nodes into the tree building procedure. Option decision trees were first introduced as classification trees by Buntine [5] and then analyzed in more detail by Kohavi and Kunz [14]. Ikonomovska et al. [10] analyzed regression option trees in the context of data streams. We also evaluated OPCTs for the multi-target regression task [16].

The major motivation for the introduction of option trees is to address the myopia of the *top-down induction of decision trees* (TDIDT) algorithm [4]. 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

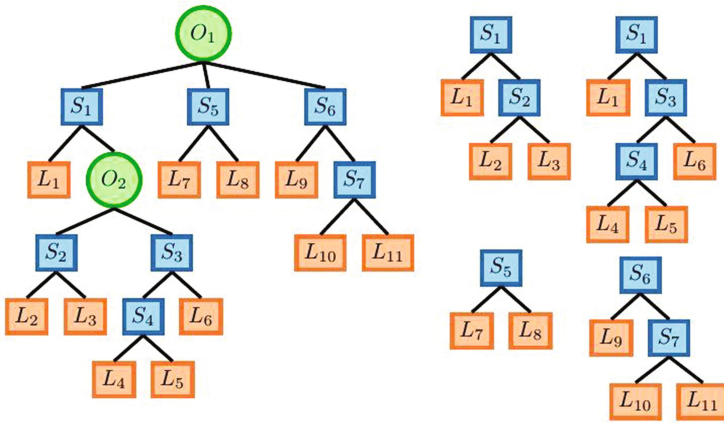
can appear instead of a single one. When using TDIDT to construct a predictive clustering tree, and in particular when partitioning the data, all possible splits are evaluated by using a heuristic and the best one is selected. However, other splits may have very similar heuristic values. The best partition 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 [14].

The procedure of PCT learning for the HMLC task is presented in [13]. We modify it by introducing an option node into the tree when the best splits have similar heuristic values. Instead of selecting only the best split, we select several of them. Specifically, we select splits  $s$ , that satisfy the condition:

$$\frac{\text{Heur}(s)}{\text{Heur}(s_{best})} \geq 1 - e \cdot d^l, \quad (1)$$

where  $s_{best}$  is the best split,  $e$  determines how similar the heuristics must be,  $d \in [0, 1]$  is a decay factor and  $l$  is the depth of the node we are attempting to split. E.g., when  $e = 0.1$ , we are selecting only splits whose heuristics are within 10% of the best split at the top level. We define the depth of a node to be the number of its ancestor nodes, excluding option nodes, as they do not split the data. The use of a decay factor makes the selection criterion more stringent in the lower nodes of the tree, where the impact of the split selection is also lower. After we have determined the candidate splits, we introduce an option node whose children are split nodes obtained by using the selected splits.

Introducing an option node with a large number of options is not advised [14] as it can lead to the explosion of model sizes. Therefore, we limit the maximum number of options for a single option node to 5 and also prohibit the induction of option nodes on depth 3 and greater.



**Fig. 1.** An option tree (left) and the ensemble of its embedded trees (right).  $O_i$  are option nodes,  $S_j$  split nodes and  $L_k$  leaf nodes.

Once an OPCT is learned, we use it for prediction. In a regular PCT an example is sorted into a leaf (reached according to the tests in the nodes of the tree) where a prediction is made using a prototype function. Traversing an example through an OPCT is the same for split nodes and leaves. When we encounter an option node, however, we traverse the example down each of the options. This means that in an option node an example is sorted to multiple leaves, where multiple predictions are produced. To obtain a single prediction in an option node, we aggregate the obtained predictions.

An option tree is usually observed as a single tree, however, it can also be interpreted as a compact representation of an ensemble. We can extract *embedded trees* out of an option tree by replacing every option node with one of its options (Fig. 1). A given OPCT is also an extension of the PCT learned on the same data. By definition, whenever we introduce an option node, we include the best split. Consequently, the PCT is an embedded tree in the OPCT, resulting from replacing all option nodes with the best option.

### 3 Experimental Design

We evaluated the performance and efficiency of the proposed OPCT method with different parameter values and compared it to the standard PCTs and ensembles of PCTs. Evaluation was done on 12 datasets from biology, text classification and image annotation domains. They are described in Table 1. The datasets came pre-divided into training and testing sets and we used them in their original format, for easier comparison of the results.

OPCTs are evaluated for various values of parameters  $e$  and  $d$ . For  $e$  we consider values 0.1, 0.2, 0.5 and 1.0, while  $d$  takes values 0.5, 0.9 and 1.0. Notably, different selections of parameters can produce the same OPCT, if for a given dataset the same splits satisfy both criteria. Hereafter, the OPCT method with specific parameter values is denoted OPCT\_eX\_dY (e.g., for  $e = 0.5, d = 0.9$ , OPCT\_e0.5\_d0.9). The border case OPCT\_e1\_d1 always selects the 5 best options regardless of their heuristic score, making this setting similar to ensembles.

For PCTs and OPCTs we use the F-test as a pruning mechanism. Specifically, we check if a split results in a statistically significant improvement over the single node. If no split satisfies the F-test, the learning in the node stops. The significance level for the test was selected from the set of values  $\{0.125, 0.1, 0.05, 0.01, 0.005, 0.001\}$  using internal 3-fold cross validation on the training set.

For ensembles, we considered bagging [2] and random forests [3]. For both methods we used 100 trees in the ensemble. Random forests algorithm also takes as input the size of the feature subset randomly selected at each node. For this we used the square root of the number of descriptive variables ( $\lceil \sqrt{|D| + |C|} \rceil$ ).

Performance was measured using Area Under the Average Precision-Recall Curve (*AUPRC*) [20]. For efficiency, we looked at the model size (number of leaves in a tree/ensemble). For statistical comparison of the methods we adopted the recommendations by Demšar [7]. Specifically, we used the Friedman test for statistical significance and Nemenyi post-hoc test to detect between which algorithms the significant differences occur. For both tests we selected confidence

**Table 1.** Descriptions of datasets used for the evaluation. The table shows the number of examples in the training and testing sets ( $N_{tr}/N_{te}$ ), number of descriptive attributes (discrete/continuous,  $D/C$ ), number of labels in the hierarchy ( $|\mathcal{H}|$ ), maximal depth of the labels in the hierarchy ( $\mathcal{H}_d$ ) and average number of labels per example ( $\bar{\mathcal{L}}$ ).

|                       | $N_{tr}/N_{te}$ | $ D / C $ | $ \mathcal{H} $ | $\mathcal{H}_d$ | $\bar{\mathcal{L}}$ |
|-----------------------|-----------------|-----------|-----------------|-----------------|---------------------|
| Diatoms [9]           | 2065/1054       | 0/371     | 377             | 3.0             | 1.95                |
| Enron [11]            | 988/660         | 0/1001    | 54              | 3.0             | 5.30                |
| Expression-FunCat [6] | 2494/1291       | 4/547     | 475             | 4.0             | 8.87                |
| Exprindiv-FunCat [6]  | 2314/1182       | 1252      | 261             | 4.0             | 3.36                |
| ImCLEF07A [8]         | 10000/1006      | 0/80      | 96              | 3.0             | 3.0                 |
| ImCLEF07D [8]         | 10000/1006      | 0/80      | 46              | 3.0             | 3.0                 |
| Interpro-FunCat [6]   | 2455/1264       | 2816      | 263             | 4.0             | 3.34                |
| Reuters [15]          | 3000/3000       | 0/47236   | 100             | 4.0             | 3.20                |
| SCOP-GO [6]           | 6507/3336       | 0/2003    | 523             | 5.5             | 6.26                |
| Sequence-FunCat [6]   | 2455/1264       | 2/4448    | 244             | 4.0             | 3.35                |
| WIPO [17]             | 1352/358        | 0/74435   | 183             | 4.0             | 4.0                 |
| Yeast-GO [6]          | 2310/1155       | 5588/342  | 133             | 6.3             | 5.74                |

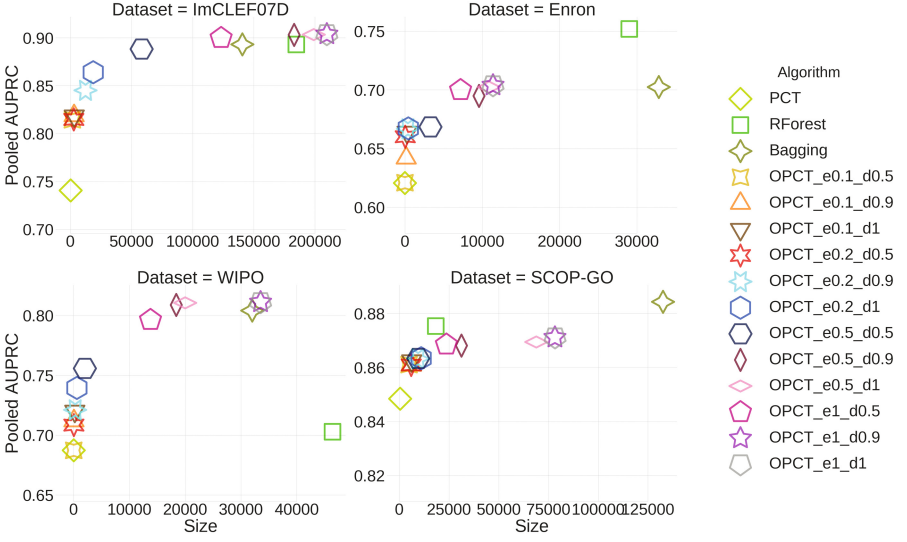
level 0.05. The results of the statistical analysis are presented with *average ranking diagrams*. They plot the average ranks of the algorithms and connect those whose average ranks differ by less than the *critical distance*. The performance of the algorithms connected with a line is not statistically significantly different.

## 4 Results and Discussion

We present our experimental results as graphs with size on the horizontal axis and performance on the vertical axis. Figure 2 shows the results on four datasets. The remaining graphs are very similar and are omitted for brevity. Notably, the figures are on separate scales and on some figures the differences in performance between the different models are very small, e.g., on the SCOP-GO dataset.

Observing the points representing the results of OPCTs, the trade-off between size and performance is clearly visible. This trade-off is achieved as a consequence of different choices of the parameter values. The models’ predictive performance generally rises with increasing model size, indicating that even the largest OPCTs do not overfit the training set, or possibly, different options overfit different parts of the input space. The increase in predictive performance in terms of increasing size also appears to saturate at the higher values of the observed parameter settings. This indicates that learning even larger less-restrictive OPCTs is not likely to provide a significant boost to predictive performance.

Compared to a PCT, OPCTs generally produce more accurate models that are mostly much larger. However, the increase in predictive performance is often

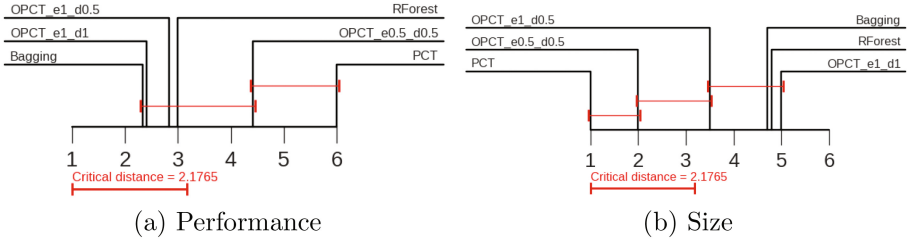


**Fig. 2.** Performances and sizes of models produced by different methods

noticeable even for the lowest parameter values when the difference in size is relatively small. The comparison between OPCTs and ensembles of PCTs is more varied. Bagging of PCTs is usually better than OPCTs (SCOP-GO), though often very slightly (Enron) and sometimes worse (IMCLEF07D). However, the size of a bagging ensemble can considerably surpass the size of even the largest OPCTs. On the Enron dataset, random forests of PCTs outperform all other methods by a solid margin. They also provide good performance on the SCOP-GO dataset with relatively small trees, however, on the WIPO dataset they produce the largest model which only outperforms a PCT.

We selected 3 parameter configurations as trade-off points between predictive performance and model size: OPCT\_e1\_d1, as it offers the best performance, OPCT\_e1\_d0.5, as its performance was similar to that of OPCT\_e1\_d1 but it often produced noticeably smaller models, and OPCT\_e0.5\_d0.5, as it consistently produced much smaller models than other two selected configurations, albeit at the cost of some performance.

We compared the performance and size of these three configurations to that of a PCT and their ensembles, using Friedman test to check if there is a significant difference between the algorithms and the Nemenyi post-hoc test to show where the differences occur. Results are presented in Fig. 3. The performance of a PCT and its size is significantly lower than that of ensembles of PCTs, OPCT\_e1\_d1 and OPCT\_e1\_d0.5. Additionally, the size of OPCT\_e0.5\_d0.5 is significantly lower than that of the four aforementioned methods, but its performance is not. We also observe that the average rank of OPCT\_e1\_d0.5 in performance is on par with ensembles of PCTs (it placed between bagging and random forests), while its average rank in size is noticeably better. As expected, a PCT always produced the smallest model with the worst performance.



**Fig. 3.** Average ranking diagrams of the performance and size of selected methods

## 5 Conclusions

In this work, we proposed an algorithm for learning option predictive clustering trees (OPCTs) for the hierarchical multi-label classification task. The purpose of OPCTs is to address the greediness of the standard algorithm for PCT learning. We experimentally evaluated the proposed method with various parameter values and compared it to PCTs and ensembles of PCTs (bagging and random forests). The results show that increasing the values of  $e$  and  $d$  increases the model performance and size compared to PCTs. At the highest parameter values of  $e = 1$ ,  $d = 1$ , OPCTs are comparable to the state-of-the-art ensemble methods of bagging and random forest of PCTs.

We identified three interesting parameter selections for OPCTs and performed statistical comparison of these three methods and regular PCTs and their ensembles. The results show that regular PCTs have significantly lower performance and size than other methods with the exception of OPCT\_e0.5\_d0.5. Additionally, OPCT\_e0.5\_d0.5 produces significantly smaller models than bagging of PCTs, random forests of PCTs and OPCT\_e1\_d1. Average performance ranks of bagging, random forests, OPCT\_e1\_d1 and OPCT\_e1\_d0.5 are very similar, while average size rank of OPCT\_e1\_d0.5 is noticeably lower than that of the other three methods. Based on these results, we suggest the parameter values of  $e \in \{0.5, 1\}$  and  $d \in \{0.5, 1\}$  for future analyses.

There are several avenues for further work. Notably, the OPCT methodology described in this paper can be easily applied to the task of multi-label classification. In the future, we also plan to use the OPCT methodology as a part of a guided process to produce regular PCTs through either input from a domain expert, or through the use of additional validation data. Finally, we will investigate the use of OPCTs for performing feature ranking and selection for HMLC.

**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 young researcher grants and the program Knowledge Technologies (P2-0103).

## References

1. Blockeel, H., Struyf, J.: Efficient algorithms for decision tree cross-validation. *J. Mach. Learn. Res.* **3**, 621–650 (2002)
2. Breiman, L.: Bagging predictors. *Mach. Learn.* **24**(2), 123–140 (1996)
3. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
4. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: *Classification and Regression Trees*. Chapman & Hall/CRC, London (1984)
5. Buntine, W.: Learning classification trees. *Stat. Comput.* **2**(2), 63–73 (1992)
6. Clare, A.: *Machine learning and data mining for yeast functional genomics*. Ph.D. thesis, University of Wales Aberystwyth, Aberystwyth, Wales, UK (2003)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)
8. Dimitrovski, I., Kocev, D., Loskovska, S., Dzeroski, S.: Hierarchical annotation of medical images. *Pattern Recogn.* **44**(10–11), 2436–2449 (2011)
9. Dimitrovski, I., Kocev, D., Loskovska, S., Dzeroski, S.: Hierarchical classification of diatom images using ensembles of predictive clustering trees. *Ecol. Inf.* **7**(1), 19–29 (2012)
10. 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)
11. Klimt, B., Yang, Y.: The enron corpus: a new dataset for email classification research. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) *ECML 2004. LNCS*, vol. 3201, pp. 217–226. Springer, Heidelberg (2004). doi:[10.1007/978-3-540-30115-8\\_22](https://doi.org/10.1007/978-3-540-30115-8_22)
12. Kocev, D., Struyf, J., Dzeroski, S.: Beam search induction and similarity constraints for predictive clustering trees. In: Dzeroski, 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)
13. Kocev, D., Vens, C., Struyf, J., Dzeroski, S.: Tree ensembles for predicting structured outputs. *Pattern Recogn.* **46**(3), 817–833 (2013)
14. 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)
15. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.* **5**, 361–397 (2004)
16. Osojnik, A., Dzeroski, S., Kocev, D.: Option predictive clustering trees for multi-target regression. In: Calders, T., Ceci, M., Malerba, D. (eds.) *DS 2016. LNCS*, vol. 9956, pp. 118–133. Springer, Cham (2016). doi:[10.1007/978-3-319-46307-0\\_8](https://doi.org/10.1007/978-3-319-46307-0_8)
17. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Kernel-based learning of hierarchical multilabel classification models. *J. Mach. Learn. Res.* **7**, 1601–1626 (2006)
18. Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., Dzeroski, S.: Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinform.* **11**(2), 1–14 (2010)
19. Silla, C., Freitas, A.: A survey of hierarchical classification across different application domains. *Data Min. Knowl. Disc.* **22**(1–2), 31–72 (2011)
20. Vens, C., Struyf, J., Schietgat, L., Dzeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. *Mach. Learn.* **73**(2), 185–214 (2008)