(Inductive) Querying Environment for Predictive Clustering Trees

Dragi Kocev [1], Saso Dzeroski [2], Jan Struyf [3] and Suzana Loskovska [1]

[1]Faculty of Electrical Engineering, Dept. of Computer Technology and Information Science, Karpos II bb, 1000 Skopje, Macedonia
[2]Jozef Stefan Institute, Dept. of Knowledge Technologies, Jamova 39, 1000 Ljubljana, Slovenia
[3]Katholieke Universiteit Leuven, Dept. of Computer Science Celestijnenlaan 200A, B-3001, Leuven, Belgium

Abstract. Inductive databases tightly integrate databases with data mining. Besides data, an inductive database also stores models that have been obtained by running data mining algorithms on the data. By means of a querying environment, the user can query the database and retrieve particular models. In this paper, we propose such a querying environment. It can be used for building new models and for searching through the database of previously built models. The models that we consider are so-called predictive clustering trees (PCTs). PCTs generalize decision trees and can be used for several prediction and clustering tasks, among others, (multi-objective) classification and regression. Our querying environment supports queries that contain size, error, and syntactic constraints on the PCTs and helps the user to quickly obtain the models of interest.

1. Introduction

The idea behind inductive databases [3] is to tightly integrate databases with data mining. An inductive database not only stores data, but also models that have been obtained by running mining algorithms on the data. By means of a querying environment, the end user can retrieve particular models. For example, the user could query the system for a decision tree that is smaller than 15 nodes, has an accuracy above 80%, and that includes a particular subtree. Alternatively, the user could ask the system to build a new tree that satisfies a given set of such conditions.

In this paper, we propose a graphical querying environment for building new decision trees and for searching through previously built trees, which have been stored in a database. We consider so-called predictive clustering trees (PCTs) [2]. PCTs generalize decision trees and can be used for several prediction and clustering tasks, among others, (multi-objective) classification and regression [4].

Our querying environment supports three types of constraints (criteria) to select PCTs: maximum size, maximum error, and syntactic constraints.

The rest of this paper is organized as follows. In Section 2, we discuss how PCTs can be constructed that satisfy the given set of constraints. Section 3 gives a brief description of CLUS, which is the system that we will use to build the PCTs.

The environment itself is discussed in Section 4. Finally, Section 5 states the main conclusions.

2. Constraint-based building of trees

Decision trees [1] are among the most well known predictive modeling techniques and are used in classification and regression tasks. A decision tree can be used to predict a value for the target attribute of a new data instance, i.e., the class in a classification task, or a numeric value in the case of a regression task. In this paper, we consider predictive clustering trees (PCTs), which generalize decision trees. In particular, we are interested in PCTs that can predict the value of more than one target attribute [24]. Figure 1 shows an example of such a PCT. Each leaf of the tree stores a vector of which the components are the predictions for the different target attributes.

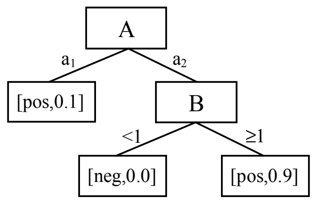


Figure 1: A multi-objective predictive clustering tree predicting both a nominal and a numeric target attribute

The advantages of using multi-objective trees are that (1) a single multi-objective tree is usually much smaller than the total size of a set of single-objective trees, one for each target variable and (2) a PCT explicates dependencies between the different target variables [24].

Constraint-based induction of PCTs is performed as follows. First a large tree is built based on a training data. For this, a generic top-down decision tree induction algorithm can be used, similar to that of C4.5 [1]. Details can be found in [24]. In a second step, the tree is pruned to satisfy the user constraints. This two-step process has the advantage that the original tree can be stored in the inductive database and used for answering inductive queries with different constraints. The algorithm that we use for pruning PCTs has been proposed by Struyf and Dzeroski [4] and is an extension of the algorithm for classification trees developed by Garofalakis et al. [4]. It is a dynamic programming algorithm that searches for a subtree of the given tree that satisfies constraints on the size (number of nodes) and training set error of the tree. In particular, the following two constraints are supported.

1. (maximum size) Given an upper bound on the tree size (S), find a tree that minimizes training set error and that has at most S nodes.
2. (maximum error) Given an upper bound on the training set error (E), find a smallest tree of which the training set error is at most E.

The error measure that is used in the constraints depends on the type of PCT. For (multi-objective) classification trees, it is the number of incorrectly predicted classes, and for (multi-objective) regression trees, it is the root mean squared error (RMSE).

Besides size and error constraints, syntactic constraints are also important in practice. Syntactic constraints can be used as follows in the context of decision trees. Suppose that a domain expert knows which attributes are important for a given application. A syntactic constraint can then be used to mine for a decision tree with such an attribute in the top node. Although other trees with different attributes in the top node might be equally accurate, the one with the attribute selected by the expert will probably be easier to interpret. More generally, the user could specify a partial tree, i.e., a subtree including the root. The result will then be a decision tree that includes the given partial tree.

The ability to use syntactic (partial tree) constraints allows for a greater involvement of the user in the construction of the decision tree and a greater user influence on the final result. Some domain knowledge of the user can be taken into account in this way.

3. CLUS System

CLUS[1] [4,5] is a generic system for constructing PCTs. We will use this system in our querying environment.

CLUS takes two files as input: a file with the training data represented as a table with one row for each instance and one column for each attribute (in the .arff format, i.e., the format of the Weka data mining system [8]), and a settings file. The settings file is used to set a number of parameters of CLUS and to specify the mining task. The latter is accomplished by defining three subsets of the attributes. These are called the descriptive attributes D, the clustering attributes C, and the target attributes T. The attributes of D are used in the internal nodes of the tree, the attributes in C are used while building the tree (for computing the heuristic), and the attributes in T are used to compute the values that are stored in the leaves.

CLUS uses a standard recursive top-down induction algorithm to construct decision trees. The same algorithm is used for inducing all types of decision trees. The main difference lies in the heuristic evaluation function that is used. For classification trees, the information gain and gain ratio heuristics are implemented in CLUS. To construct multi-objective classification trees, CLUS uses a trivial extension to information gain, in which the entropy measure is replaced by the sum of the entropies computed for the different target attributes [5]. For regression trees, multi-objective regression trees, and clustering trees, CLUS uses a heuristic that is based on intra-cluster variation. CLUS also supports the size, error, and syntactic constraints that we discussed in the previous section.

4. Querying Environment for Predictive Clustering Trees

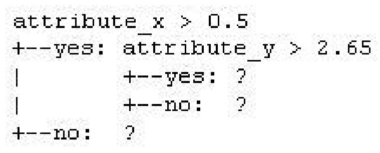
The main goal of this paper is to create a querying environment for specifying inductive queries for building trees and queries for searching certain trees in the

[1] CLUS is available at http://www.cs.kuleuven.be/~dtai/clus.

database. We use CLUS for building the trees. The environment is implemented as a Graphical User Interface (GUI).

First we select a data set (in the Weka .arff format [8]) and after that we can select descriptive, target, and clustering attributes (Section 3). After that we can define error, size and syntactic constraints. The maximum error can be set for each target attribute separately (if we wish some target attribute to be predicted more accurately) or one can set the overall error (summed over all targets). Also, we can set the maximum size of the tree.

The syntactic constraints can be set either visually or textually. If we wish to set the trees textually we must oblige to the CLUS representation of the trees (Figure 1). When we are setting the tree visually we can manipulate the tree and perform the following operations: (1) add node, (2) change node and (3) delete node. After setting all the constraints we run the CLUS system. The models that we obtain are stored in a database and we can search through them later on. Existing models can also be used to construct new syntactic constraints by adding, deleting or changing some nodes.



```
attribute_x > 0.5
+--yes: attribute_y > 2.65
|       +--yes: ?
|       +--no:  ?
+--no:  ?
```

Figure 2: Textual representation of a partial tree (new nodes can be added at the positions of the question marks)

When we search through previously built models we can set constraints (criteria) for the search. We can again specify overall error, the maximum size of the tree, and syntactic constraints. In this case, syntactic constraints can also be expressed as subsumption constraints, e.g., search for all trees that include a particular subtree. All the constraints can be logically connected as illustrated in Table 1. The subsumption constraints can be specified either visually or textually. The same tree manipulations are supported as for the partial tree constraints discussed above (add, change and delete node). Figure 3 presents an example of the subtree tests that are supported. In this figure, both tree (b) and tree (c) are a subtree of tree (a).

Table 1: Logical connections between search criteria

Connection between constraints (criteria)
E and S and SC
(E and S) or SC
E or (S and SC)
E or S or SC
E and (S or SC)
(E or S) and SC
E or S
E and S
E=Error; S=Size; SC=Subsumption Constraint

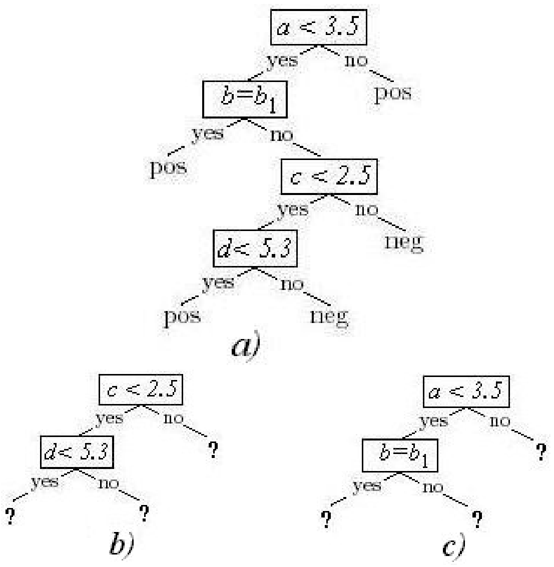
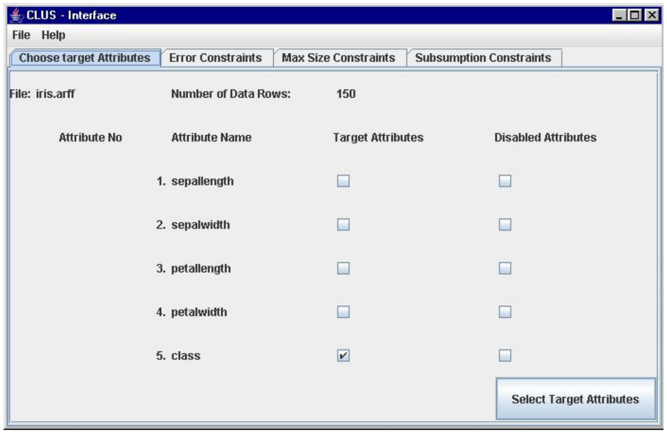


Figure 3: Subsumption constraints

4.1 Example Using the Environment

The environment allows us to learn new models and search through previously built models very easy. This we will demonstrate with some screen-shots of the environment. We shall use the iris dataset (the dataset is from UCI repository).

First, we select the iris.arff file and we get the frame shown in Figure 4. We can select target and disabled attributes. After the selection we can set the constraints discussed previously.



File: iris.arff Number of Data Rows: 150

Attribute No	Attribute Name	Target Attributes	Disabled Attributes
1.	sepallength	☐	☐
2.	sepalwidth	☐	☐
3.	petallength	☐	☐
4.	petalwidth	☐	☐
5.	class	✓	☐

Figure 4. Selection of target and disabled attributes

In this example, we chose to predict the value of the class. After that, we can set the maximum error costraint and/or maximum size constraint and/or syntactic constraint by selecting the adequate tab. We have set the error constraint to 5% (accuracy 95%) and maximum size is 15. Setting the subsumption constraints is shown in Figure 5.

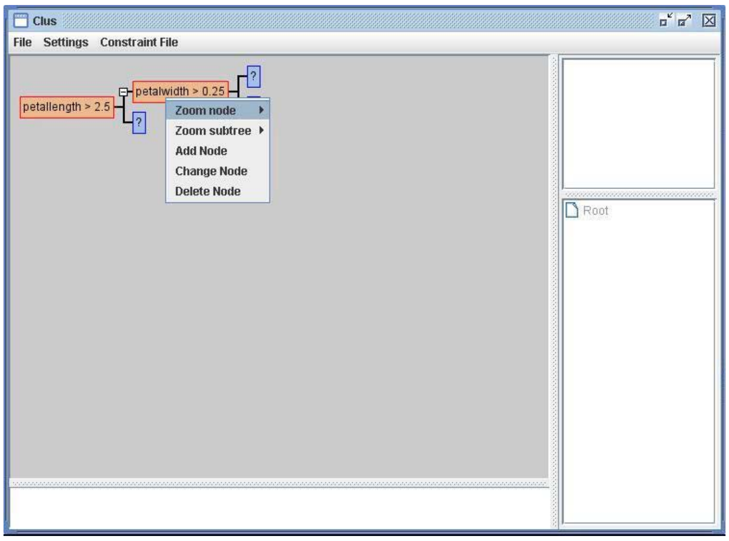
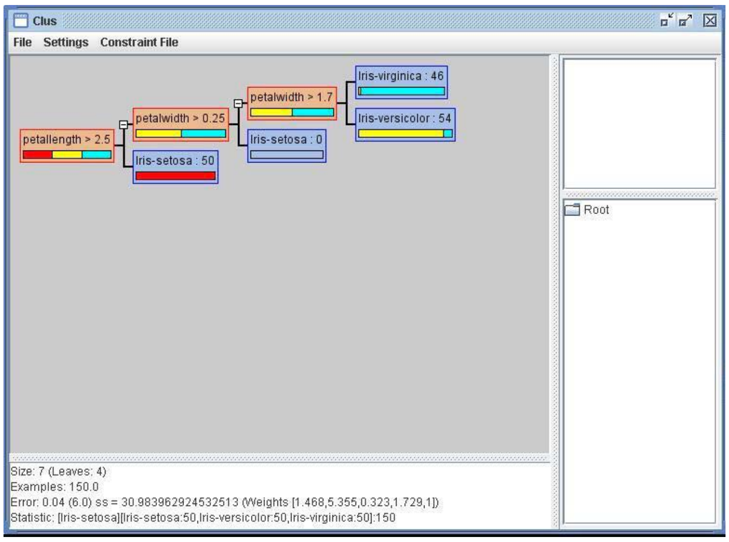


Figure 5. Setting the subsumption constraint



```
Size: 7 (Leaves: 4)
Examples: 150.0
Error: 0.04 (6.0) ss = 30.983962924532513 (Weights [1.468,5.355,0.323,1.729,1])
Statistic: [Iris-setosa][Iris-setosa:50,Iris-versicolor:50,Iris-virginica:50]:150
```

Figure 6. Obtained model from CLUS

The result from CLUS is a tree with size 7 and error 4% (6 misclassified examples) which is built accordingly to syntactic constraint that we have entered. We can re-use model shown in Figure 6. as a new syntactic constraint for new query.

Search through previously built models is suported. First we select a database to perform the search (Figure 7.). After that, we can set the search constraints (by selecting the adequate tab).

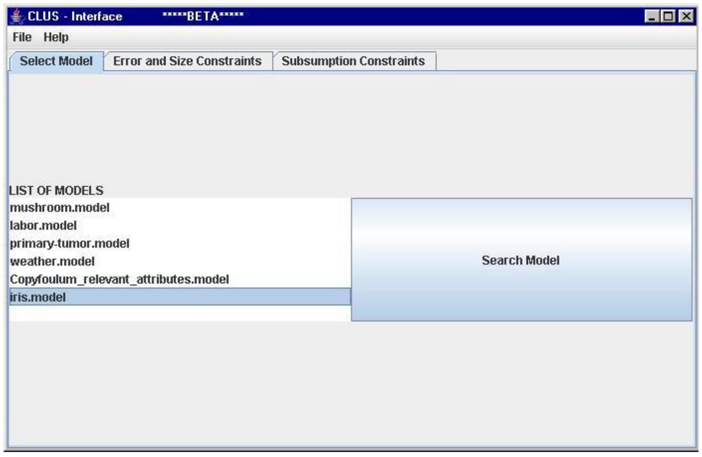


Figure 7. Selecting a set of models

Now, we shall set the error constraint to 5% and maximum size to 10. Additionally we will set the subsumption constraint given in Figure 8. As a result of our search, among other results, we find the model from Figure 6. We can re-use this model for a new subsumption constraint.

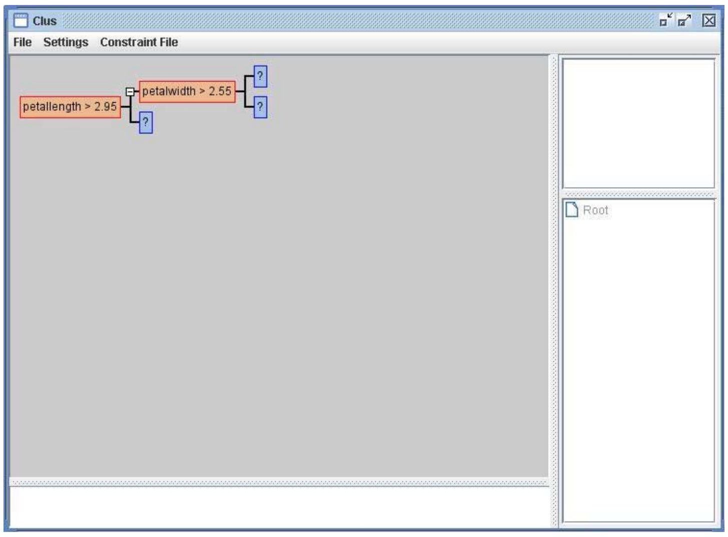


Figure 8. Subsumption constraint for search through models

5. CONCLUSION

While the notion of inductive databases has been propsed a decade ago [3], few implementations exist that support the storage of and query over induced patterns. The authors are not aware of any such work for predictive models. Our work is unique in that respect, especially because the inductive queries supported allow for both inducing models from data and querying over already induced models.

In this paper we proposed a querying environment for predictive clustering trees. Using this environment we can (1) set the constraints with which we wish to build a tree and (2) set search constraints for searching through the database of previously built trees. The supported constraints are maximum error constraints, size constraints and syntactic constraints. When we are building models we can select descriptive, target, and clustering attributes and we can specify the maximum error as error for each target attribute separately and overall error summed over all target attributes. The syntactic constraints can be set either visually or textually. The following manipulations are supported: add node, change node, and delete node. Consistency and reusability (as syntactic constraints for a new query) of the learned models are provided. To search through models logical connections between the search constraints are supported.

References

1. M. Dunham. Data Mining: Introductory and Advanced Topics, Pearson Education 2003, ISBN 0-13-088892-3.
2. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In Proceedings of the 15th International Conference on Machine Learning, pages 55–63, 1998.
3. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of the ACM, 39(11):58–64, 1996.
4. J. Struyf and S. Dzeroski. Constraint based induction of multi-objective regression trees. In Proceedings of the 4th International Workshop on Knowledge Discovery in Inductive Databases, pages 110-121. 2005.
5. J. Struyf. The Clus system (draft). 2005.
6. M. Garofalakis, D. Hyun, R. Rastogi, and K. Shim. Building decision trees with constraints. Data Mining and Knowledge Discovery, 7(2):187–214, 2003.
7. J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann, 1993.
8. I. Witten and E. Frank. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005. 2nd Edition.