SIMILARITY CONSTRAINTS IN BEAM-SEARCH INDUCTION OF PREDICTIVE CLUSTERING TREES

Dragi Kocev [1], *Sašo Džeroski* [1] *and Jan Struyf* [2]

[1]Jožef Stefan Institute, Department of Knowledge Technologies, Jamova 39, SI-1000, Ljubljana, Slovenia, {Saso.Dzeroski, Dragi.Kocev}@ijs.si
[2]Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Leuven, Belgium, Jan.Struyf@cs.kuleuven.be

ABSTRACT

We investigate how inductive databases (IDBs) can support global models, such as decision trees. We focus on predictive clustering trees (PCTs). PCTs generalize decision trees and can be used for prediction and clustering, two of the most common data mining tasks. Regular PCT induction builds PCTs top-down, using a greedy algorithm, similar to that of C4.5. We propose a new induction algorithm for PCTs based on beam-search. This has three advantages over the regular method: (1) it returns a set of PCTs satisfying the user constraints instead of just one PCT; (2) it better allows for pushing of user constraints into the induction algorithm; and (3) it is less susceptible to myopia. In addition, we propose similarity constraints for PCTs, which improve the diversity of the resulting PCT set.

1. INTRODUCTION

Inductive databases (IDBs) [1][2] represent a database view on data mining and knowledge discovery. IDBs contain not only data, but also models. In an IDB, ordinary queries can be used to access and manipulate data, while inductive queries can be used to generate, manipulate, and apply models. For example, "find a set of accurate decision trees that have at most ten nodes" is an inductive query.

IDBs are closely related to constrained based mining [3]. Because the inductive queries can include particular constraints, the IDB needs constrained based mining algorithms that can be called to construct the models that satisfy these constraints. The above example query includes, for example, the constraint that the trees can contain at most ten nodes.

Much research on IDBs focuses on local models, i.e., models that apply to only a subset of the examples, such as item sets and association rules. We investigate how IDBs can support global models. In particular, we consider predictive clustering trees (PCTs) [4]. PCTs generalize decision trees and can be used for both prediction and clustering tasks. We define PCTs in Section 2.

Regular PCT induction builds PCTs top-down using a greedy algorithm similar to that of C4.5 [5]. This has three main disadvantages w.r.t. inductive databases: (1) it returns only one PCT. This is incompatible with the IDB view that inductive queries should return the set of all models satisfying the constraints in the query. (2) many useful constraints cannot be pushed into the induction algorithm. Size constraints, such as the one in our example query, must be handled partly during post-pruning [6]. (3) because the algorithm is greedy it is susceptible to myopia: it may not find any tree satisfying the constraints even though several exist in the hypothesis space.

In this paper, we propose a new induction algorithm for PCTs that addresses these three problems to a certain extent. The algorithm employs beam-search. Beam-search considers at each step of the search the k best models according to a particular evaluation score. Therefore, it trivially returns a set of models instead of just one model. Beam-search also supports pushing of size constraints into the induction algorithm, as we will show in Section 3. Finally, beam-search is known to be less susceptible to myopia than regular greedy search.

An important disadvantage of using beam-search is that the beam tends to fill up with small variations of the same PCT, such as trees that differ only in one node. To alleviate this, we propose similarity constraints for PCTs. We show that these constraints improve beam diversity.

2. PREDICTIVE CLUSTERING TREES

PCTs [4] are generic decision trees that can be used for a wide variety of data mining tasks including different types of prediction and clustering. PCTs have been applied to multi-objective classification and regression [7], hierarchical and multi-label classification [8], and clustering of time series [9].

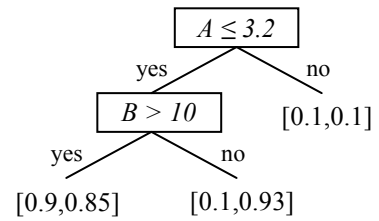

Figure 1: A PCT predicting two numeric attributes.

An example of a multi-objective PCT predicting two numeric attributes is show in Figure 1. Each leaf stores a

vector with as components the predictions for the different target variables.

PCTs can be constructed with a greedy top-down induction algorithm. The algorithm is similar to that of C4.5 [5], except that the heuristic for selecting the tests in the internal nodes and the procedure for computing the labels in the leaves is different. For example, to construct the PCT in Figure 1, the heuristic for selecting the tests is minus the intra-cluster variation (ICV) summed over the subsets induced by the test and the label of a given leaf is the average of the target vectors of the examples sorted in the leaf. Intra-cluster variation is defined as $ICV(D) = N \cdot \sum_{t=1}^{T} Var[y_t]$, with N the number of examples in the subset D, T the number of target variables, and $Var[y_t]$ the variance of target variable t in D. Minimizing ICV results in homogeneous leaves, which in turn results in accurate predictions.

PCTs are implemented in the CLUS system. CLUS supports various types of PCTs. CLUS implements syntactic constraints and constraints on the size and/or accuracy of the trees [7]. More information about PCTs and CLUS can be found at: http://www.cs.kuleuven.be/~dtai/clus/.

3. BEAM-SEARCH

Fig. 1 shows the beam-search algorithm that we propose. The beam is a set of PCTs ordered by their heuristic value. The algorithm starts with a beam that contains precisely one trivial PCT: a leaf covering all the training data D.

Each iteration of the main loop creates a new beam by refining the PCTs in the current beam. That is, the algorithm iterates over the trees in the current beam and computes for each PCT its set of refinements. A refinement is a copy of the given PCT in which one particular leaf is replaced by a depth one sub-tree (i.e., an internal node with a particular attribute-value test and two leaves). Note that a PCT can have many refinements: a PCT with N leaves yields N × M refined trees, with M the number of possible tests that can be put in a new node. In CLUS, M is equal to the number of attributes. That is, CLUS considers for each attribute only the test that maximizes the heuristic value. This approach limits the number of refinements of a given PCT and increases the diversity of the trees in the beam[1].

CLUS computes for each generated refinement its heuristic value. If this value is larger than the value of the worst PCT in the beam or if there are fewer than k trees (k is the beam-width), then it adds the new PCT to the beam and, if this exceeds the beam-width, removes the worst tree from the beam.

The algorithm ends if a given stopping-criterion is met, such as the beam no longer changes. Note that this occurs if none of the trees in the beam yields any valid refinements. A refinement is valid in CLUS if it does not violate any of the constraints imposed by the user, such as maximum depth, maximum size, or minimum number of examples in each cluster.

```
procedure Beam-Search(D,k) returns Beam
i := 0
leaf := create-leaf(D)
H := heuristic(leaf, D)
beam_0 := { (H, leaf) }
while not stop-criterion(beam_i)
  beam_{i+1} := beam_i
  for each tree ∈ beam_i
    R := refinements(tree, D)
    for each ref-tree ∈ R
      H := heuristic(ref-tree, D)
      H_min := min-heuristic(beam_{i+1})
      if H > H_min or |beam_{i+1}| < k then
        beam_{i+1} := beam_{i+1} ∪ { (H, ref-tree) }
      if |beam_{i+1}| > k then
        beam_{i+1} := remove-min(beam_{i+1})
  i := i + 1
return beam_i
```

Figure 2: The beam-search algorithm of CLUS.

The heuristic value computed for a tree in beam-search mode differs from the heuristic used in the top-down algorithm from Section 1. The heuristic value in the latter is local, i.e., computed only based on the examples in the node that is being constructed. In beam search mode, the heuristic is global, measuring the quality of the entire tree. The heuristic that we use is:

$$H(T) = -\frac{1}{|D|} \cdot \left(\sum_{leaf_i \in T} ICV(D_i)\right) - \alpha \cdot size(T),$$

with T the given tree, D all training data, and D_i the examples sorted into $leaf_i$. It has two components: the first one is minus the intra-cluster variation of the PCT and the second one is a size penalty. The latter biases the search to smaller trees.

4. SIMILARITY CONSTRAINTS

So far, the heuristic value only takes the error (ICV) and the size of the PCT into account (Section 3). In this section, we define soft similarity constraints, which can be included in the heuristic and bias the search to a set of trees that is less similar.

To quantify the similarity of two trees, we define a distance metric between trees. The distance metric is computed based on the predictions of the trees. We first define a distance metric for single-objective regression and classification. For regression, we have:

$$d(T_1, T_2) = \frac{1}{M\text{-}m} \sqrt{\frac{\sum_{j=1}^{N} \left(p(T_1, t_j) - p(T_2, t_j)\right)^2}{N}}$$

[1] The number of possible tests on a numeric attribute A is typically huge: one test $A < a_i$, for each possible split point a_i. Clus only constructs one refined tree for the split that yields the best heuristic value.

with $d(T_1, T_2)$ the distance between tree T_1 and T_2, $p(T_i, t_j)$ the prediction of tree T_i for instance t_j, N the number of instances in the training set, M the maximum value of $p(T_i, t_j)$, and m the minimum value of $p(T_i, t_j)$. This corresponds to the mean normalized Euclidean distance between the predictions. The normalization ensures that the distance will be in the interval (0,1).
For classification, we use:

$$d(T_1, T_2) = \sqrt{\frac{\sum_{j=1}^{N} \delta(p(T_1, t_j), p(T_2, t_j))}{N}}$$

with $d(T_1, T_2)$ the distance between tree T_1 and T_2, $p(T_i, t_j)$ the prediction of tree T_i for instance t_j, N the number of instances in the training set, and $\delta(a,b) = \begin{cases} 0 \text{ if } a = b \\ 1 \text{ if } a \neq b \end{cases}$.
The heuristic value for a new candidate tree is modified by adding the average of the distances to all trees in the beam as follows:

$$H(T) = -\frac{1}{|D|} \cdot \left(\sum_{leaf_i \in T} ICV(D_i) \right) - \alpha \cdot size(T) - \beta \cdot SIM(T)$$

with $SIM(T) = 1 - \frac{d(T, T_{cand}) + \sum_{j=1}^{k} d(T, T_j)}{k}$ the average similarity to the trees in the beam and the candidate tree, and k the beam-width.
This heuristics is recomputed for each tree in the beam and the candidate tree. If the candidate tree has score greater than the minimal then it enters the beam and the tree with minimal score is left-out.

5. EXPERIMENTAL SETUP

We have implemented the beam-search algorithm and the similarity constraints in the CLUS system. We compare CLUS with the regular top-down induction algorithm (TD) for PCTs to CLUS with beam-search (BS) and beam-search with similarity constraints (BS-s) on 9 regression and 8 classification data sets from the UCI machine learning repository [10]. We set the parameters of the beam-search algorithms ad-hoc to the following values k = 10, α = 0.1, and β = 1. For TD, we set a size constraint so that the trees can contain at most 7 nodes [6]. This value is set such that the trees are approximately the same size as the trees obtained with beam-search.
Note that the heuristics defined in Section 2-3 are designed for regression data. For the classification data sets, we use different heuristics that are obtained by replacing in the regression variants $ICV(D_i)$ by $|D_i| \cdot Entropy(D_i)$ with $Entropy(D_i)$ the class entropy of the set D_i.
For each algorithm, we measure the predictive performance of the resulting PCT and its size. For classification data we report accuracy and for regression data the Pearson correlation coefficient. The values listed for the beam-search algorithms are those of the best scoring model.
To quantify the effect of the similarity constraints, we report for the two beam-search algorithms the *Beam Similarity*, which is the average similarity of the trees in the beam. *Beam Similarity* is computed as follows:

$$Beam\ Similarity = \frac{\sum_{i=1}^{k} S(T_i)}{k},$$

with $S(T_i) = 1 - \frac{\sum_{j=1}^{k} d(T_i, T_j)}{k}$ the similarity of tree T_i w.r.t. the other trees in the beam, $d(T_1, T_2)$ the distance between tree T_1 and T_2, and k the beam-width.

6. RESULTS AND DISCUSSION

Table 1 and Table 2 present the results. For most data sets, the results are accuracy or correlation-wise comparable. The most noticeable differences are obtained for the datasets pyrim, pollution and segment. Here the correlation or the accuracy of beam-search with similarity constraints is considerably better than that of top-down tree induction.
The effect of including the similarity constraints can be seen from the reported beam similarity. For all data sets, beam similarity reduces by using the similarity constraints.

7. CONCLUSIONS AND FURTHER WORK

We propose a new algorithm for inducing predictive clustering trees (PCTs) that employs beam-search. The main advantages of this algorithm are that it induces a set of PCTs instead of just one PCT, that it supports pushing of user constraints, and that it is less susceptible to myopia. Furthermore, we propose soft similarity constraints based on the predictions of the PCTs. The similarity constraints improve beam diversity.
A preliminary experimental evaluation illustrates some of the advantages of the approach. In the future, we plan a more extensive evaluation, among others, quantifying the influence of the similarity constraints on the heuristic value (the effect of the β parameter).
Note that diversity, which is obtained by means of our heuristic, has been shown to increase the predictive performance of classifier ensembles. Therefore, we plan to investigate if beam-search with the similarity constraints can be used to construct an ensemble of PCTs.

Acknowledgments: Jan Struyf is a postdoctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen).

Table 1. *Results for the regression data sets (TD is regular top-down induction, BS is beam-search, and BS-s is beam-search with similarity constraints).*

	No. of Attributes	Correlation			Size			Beam Similarity	
		TD	BS	BS-s	TD	BS	BS-s	BS	BS-s
autoPrice	16	0.8839	0.8464	0.7965	7	3	5	0.8408	0.7247
bodyfat	15	0.9366	0.8748	0.8748	7	5	5	0.9004	0.7803
cpu	8	0.9240	0.8438	0.8438	7	5	5	0.9387	0.9290
housing	14	0.7960	0.7496	0.7496	7	5	5	0.8261	0.7902
pollution	16	0.5012	0.5647	0.9910	7	5	5	0.7296	0.6771
servo	5	0.8885	0.9104	0.9104	7	7	7	0.8933	0.8161
cpu_act	22	0.9568	0.9431	0.9443	7	5	5	0.9482	0.9293
pyrim	28	0.6752	0.6146	0.9021	7	5	3	0.9540	0.6728
machine_cpu	7	0.8395	0.8335	0.7356	7	5	3	0.9324	0.9181

Table 2. Results for the classification tasks *(TD is regular top-down induction, BS is beam-search, and BS-s is beam-search with similarity constraints).*

	No. of Attributes	Accuracy			Size			Beam Similarity	
		TD	BS	BS-s	TD	BS	BS-s	BS	BS-s
car	7	0.7917	0.7917	0.7922	7	5	9	0.6175	0.5824
mushroom	23	0.9941	0.9941	0.9941	5	5	5	0.9741	0.8105
segment	20	0.5558	0.8108	0.8095	7	11	11	0.9256	0.4367
vowel	14	0.2515	0.2818	0.2747	7	9	5	0.4111	0.2677
vehicle	19	0.5118	0.6017	0.6028	7	7	7	0.9101	0.3449
iris	5	0.9400	0.9600	0.9600	7	5	5	0.8978	0.6438
ionosphere	35	0.8860	0.8718	0.8718	7	5	5	0.6824	0.5851
kr-vs-kp	37	0.9043	0.9309	0.8833	7	9	7	0.5663	0.4248

REFERENCES

[1] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of the ACM, 39(11):58-64, 1996.

[2] L. De Raedt. A perspective on inductive databases. SIGKDD Explorations, 4(2):69-77, 2002.

[3] J-F. Boulicaut, B. Jeudy. Constraint-based data mining. The Data Mining and Knowledge Discovery Handbook, O. Maimon and L. Rokach (Eds.), Springer, pp. 399-416, 2005.

[4] H. Blockeel, L. De Raedt and J. Ramon. Top-down induction of clustering trees. In Proceedings of the 15th International Conference on Machine Learning, p. 55-63, 1998.

[5] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann, 1993.

[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. 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, p. 110-121. 2005.

[8] H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski, and A. Clare, Decision trees for hierarchical multilabel classification: A case study in functional genomics, Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, p. 18-29, 2006.

[9] S. Dzeroski, I. Slavkov, V. Gjorgjioski and J. Struyf, Analysis of time series data with predictive clustering trees. In proceedings of the 5th International Workshop on Knowledge Discovery in Inductive Databases, p. 47-58. 2006.

[10] D.J. Newman, S. Hettich, C.L. Blake and C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.