From Inductive Databases to DL8.5

Siegfried Nijssen and Pierre Schaus

ICTEAM, UCLouvain
siegfried.nijssen@uclouvain.be

Abstract. The core idea of inductive databases being to create systems that can be queried declaratively for patterns and models, a natural question was how such systems would work for one of the most popular models in machine learning, the decision tree. In our work first on this topic, in 2007, we introduced DL8, an algorithm that showed how to calculate optimal decision trees from itemsets. Driven by an interest in imposing additional requirements on predictive models, such requirements of interpretability and fairness, in recent years the calculation of optimal decision trees has gained a lot of interest again. In this context, an extension of DL8, DL8.5, obtained better performance than competitor systems. This abstract describes how research on inductive databases led to these recent systems and provides an overview of these systems.

Keywords: Inductive databases · Decision Trees · Constraint-based Mining

1 Introduction

A core idea of inductive databases was to create systems similar to database systems that would allow a user to query a database for patterns and models in a declarative manner: a user could specify a number of different requirements in a declarative language, after which this system would be responsible for finding the patterns or models satisfying the requirements. Initially, a lot of this research focused on developing systems for patterns: indeed, in the first iterations of the international workshop on Knowledge Discovery using Inductive Databases (KDID), the majority of the papers focused on pattern mining [19, 8], and languages were studied that would allow a user to impose constraints on patterns.

However, as it was recognized that predictive models are very important in machine learning and data mining, subsequent studies considered inductive databases that would support models as well. One such model was the decision tree, as reported on in the KDID workshop of 2006 [13, 20]. The general idea was to create a system that would allow users to formulate and answer queries such as:

Given a database D
Find a classification tree T
Such that

- $accuracy(T)$ is maximal
- $depth(T) < maxdepth$
- $\forall l \in leaves(T)$: $support(l) \geq minsup$

Here *maxdepth* and *minsup* are parameters specified by the user, and users would have the ability to add and remove constraints, as required by the application context. These queries would be expressed in a form of SQL [13] (following the ADReM approach using Mining Views) or using a form of logic [20]; an underlying algorithm would find trees satisfying these conditions.

The constraints proposed initially were relatively simple in nature, involving characteristics such as accuracy, size and depth. However, soon it was realized that other constraints could also be relevant, such as on the privacy preserving nature or the cost of the decision trees, and these were added to the language of constraints that inductive databases could support [22].

For many years, this work went unnoticed. However, in recent years the interest in imposing additional requirements on predictive models has risen significantly: issues of explainability, fairness, and privacy have grown dramatically in importance. This has led to a renewed interest in the question of how to impose requirements on decision tree models. Here, the work on inductive databases has recently been shown to be particularly relevant. In this abstract, we present a short summary of how the work on inductive databases led to recent state-of-the-art algorithms for learning optimal decision trees.

2 Algorithms in Inductive Databases

While creating a language for formulating some form of decision tree queries is not very difficult, an important challenge for inductive database systems was how to create algorithms for answering these queries. Traditional algorithms for learning decision trees, such as CART, are heuristic in nature and have no functionality to ensure that good trees are found under constraints [10]. Indeed, one variant of the decision tree learning problem under constraints was shown in 1976 to be NP complete and similar problems are assumed to be NP complete as well [16].

A first attempt to address this inductive databases was made by Fromont et al. [13] in 2006. This work proposed an exhaustive search algorithm that would enumerate all possible decision trees up to a certain size, filtering out those that would not meet the constraints specified by the user. When accuracy was used as optimisation criterion, a bound based on accuracy was also used to prune the search space.

The scalability of this approach was still very limited; it only worked on all but the smallest databases. A challenge remained how to answer inductive queries on more realistic databases.

One idea that surfaced here was based on the observation that significant effort in the inductive databases community had been spent on how to build efficient algorithms for pattern mining and itemset mining in particular [14, 6].

We asked the question: given the high performance of itemset mining algorithms, can databases with patterns be used to efficiently find decision trees under constraints?

In 2007, this led to our proposal for the DL8 algorithm ('Decision trees from Lattices') [21]. The core idea underlying this algorithm is that a decision tree consists of paths, paths are itemsets, and hence decision trees can be constructed from itemsets. Indeed, our initial implementation in fact operated on the output of a frequent itemset mining algorithm (Eclat).

Compared to earlier algorithms, a key element of the DL8 algorithm is that it solves the problem of finding a decision tree using dynamic programming. For the following problem:

[Depth-Constrained Accurate Trees]
Given a Boolean database D with examples labeled in two classes
Find a classification tree T
Such that

- $error(T)$ is minimal
- $depth(T) \leq maxdepth$

the following recursive equation is at the basis of this dynamic programming approach:

$$min\_error(I) = \begin{cases} \min_{F \in \mathcal{F}} \sum_{t \in tests(F)} min\_error(I \cup \{t\}) & \text{if } |I| < maxdepth; \\ leaf\_error(I) & \text{if } |I| = maxdepth. \end{cases}$$

Here the recursion starts at $min\_error(\emptyset)$, I is an itemset, $\mathcal{F}$ is the set features in the database, $tests(F)$ returns two items for each feature (one item for the value 0, one item for the value 1), and $leaf\_error(I)$ is the number of examples in the minority class of the examples covered by the itemset I. In words, the tree with minimal error is obtained by picking in the root the feature which leads to the lowest sum of errors for the left-hand and right-hand child.

The dynamic programming approach is based on storing the value min_error for itemsets such that they do not need to be recalculated later on when the same features are considered in a different order. This trick has as effect that the algorithm does not need to enumerate all trees, but only needs to enumerate itemsets. We showed in 2007 that this makes it possible to find optimal decision trees for a much larger number of databases [21]. Here we used a customized search algorithm that exploited many of the implementation tricks of itemset mining algorithms at the time.

While in our example above we use a depth constraint, in our original publication we did not give this constraint much attention; we introduced an optimisation which allowed the algorithm to find optimal decision specifically in cases where this constraint was absent, and did our experiments for that setting. In these experiments we found that in some cases optimal decision trees have better predictive performance than heuristically learned trees, in other cases their performance is worse.

We subsequently showed that this approach of dynamic programming over itemsets can also be used in other settings: when adding a *regularisation term* in the optimisation criterion, when taking into account *costs* and when taking into account a score for *discrimination* [22].

3 Optimal Decision Trees using MIP

For a number of years there was little interest in the problem of finding optimal decision trees. A publication by Bertsimas and Dunn [7] in 2017 seems to have been important in the revival of the topic, and has attracted a wide interest in the topic. Their work showed the following:

- for the problem of finding Depth-Constrained Accurate Trees, optimal decision trees in many cases perform better than heuristically learned trees;
- the problem of finding optimal trees can be formalized as a Mixed Integer Programming problem, allowing the use of Mixed Integer Programming solvers to find such trees.

This led to a large interest in the topic, in particular among researchers with an expertise in Mixed Integer Programming. A number of subsequent publications demonstrated improvements in how to model decision trees in MIP [23, 2, 9] and applied the MIP approach to include other forms of optimisation criteria, such as based on fairness [1, 17]. In particular the fact that additional constraints can be added by adding constraints in the MIP formulation, makes this approach attractive, as it means no new algorithms need to be developed and expertise in mathematical modeling is sufficient to solve other forms of learning problems.

4 Reviving DL8

Bertsimas and Dunn were clearly unaware of prior work on learning decision trees in the inductive database context; hence, their publication also did not include a comparison with our earlier work on DL8. In 2019 we decided to compare our original DL8 implementation with one of the optimized versions of the MIP approach, BinOCT [23], on exactly the same optimisation problem, that is, a problem with a depth constraint. The results are reproduced in Figure 1, where it is shown how much time is needed to find an optimal tree for a number of instances (where each instance corresponds to a dataset). The comparison showed that the performance of the MIP based approach was much worse than that of the DL8 algorithm. BinOCT could only find optimal trees within reasonable time for a depth of 2 and finding these trees still took much more time than required by DL8. This led us to give DL8 a second look, in particular in the context of depth-constrained decision trees. This led to the following additional contributions.

First, we studied whether the idea of using bounds during a branch-and-bound search could be added to DL8. This turned out to be possible, and was

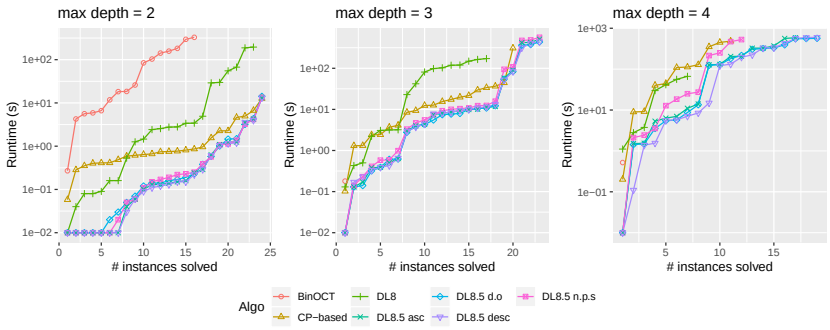

Fig. 1. Cumulative number of instances solved over time

implemented in an algorithm that we called DL8.5. Results for this algorithm are also shown in Figure 1 a significant further speed-up with respect to DL8 was obtained, making the gap with the MIP-based approach even larger.

A perceived strength of the MIP-based approach is that it allows to model other learning problems simply by changing the MIP formulation, similar to how inductive databases would allow to find many different types of models. To make DL8.5 also more generally usable, we created a library in Python which allows the user also to implement their own scoring function in Python [3]; this makes it possible to use DL8.5 easily also for regression problems, clustering problems, and more, simply by writing a small amount of Python code. This brings DL8.5 practically closer to a system that allows writing queries.

We studied whether the idea of finding optimal trees can also be extended towards forests of decision trees, and showed that this is the case for LP-Boost based methods [4]. The addition of further constraints is here however still an unsolved problem.

Compared to MIP, a possible weakness of DL8.5 is its memory use: it stores large amounts of itemsets in order to calculate decision trees from them. In practice this means that the algorithm can run out of memory on machines with small amounts of memory. We addressed this challenge in an extension of DL8.5 which from time to time also removes itemsets from memory; while this could mean these itemsets need to be recalculated later, with proper heuristics we can assure that this does not happen too often. In practice this makes it possible to run the algorithm in memory constrained environments for the price of not too much additional run time; indeed, even under memory constraints DL8.5 remains much faster than MIP-based approaches [5].

Despite its better performance, DL8.5 still takes significant amounts of time for calculating an optimal tree on some datasets. We asked the following question: suppose the user stops the algorithm after a certain amount of time, can we still assure that the algorithm will return a good tree at that moment? This led us to study different orders for traversing the search space [18].

The good performance of DL8.5 has led other people to continue with this algorithmic approach. In particular, Demirovic et al. proposed MurTree [11], which improves the performance of DL8.5 further by proposing a specialised algorithm for trees of depth 2 and by improving the bounds used in the branch-and-bound search. In this work low-level optimisations are used that have some resemblance to the optimisations that were also used in itemset mining algorithms some time ago. This study, which used an implementation created from scratch, confirmed also independently the superior performance of the DL8-type of algorithm over MIP-based approaches for finding decision trees on Boolean data.

An extension of DL8.5's algorithm was also used to find decision trees for certain types of non-linear scoring functions [12].

5 Conclusions

In this paper, we provided a short overview of optimal decision tree learning such as studied initially in the context of the Knowledge Discovery in Inductive Databases workshop, and how this has led to a type of algorithm, DL8, which has recently been studied in detail again and led to numerous new publications.

Given the fact that also in the first half of 2022 already many new publications have appeared on algorithms for finding optimal decision trees, we believe that this work will continue to be relevant. Also other older results in the domains of inductive databases and pattern mining may here gain relevance once more.

For instance, many of the current approaches are focused on depth-constrained decision trees. However, in our past work we showed that optimisations based on *condensed representations* of itemsets allow to find optimal decision trees also without such a constraint. The use of condensed representations may hence also merit a new look in this context.

DL8's performance derives from dynamic programming, where it assumed that the left-hand and right-hand side of a test in a tree can be optimised independently. For some scoring functions this independence does not apply. Search algorithms such as developed by Fromont et al. [13] could still be relevant here.

At its core, DL8 still performs a form of itemset mining. Many ideas present in the itemset mining literature, such as FP-Trees [15], have not been evaluated in the context of DL8 yet.

References

1. Aghaei, S., Azizi, M.J., Vayanos, P.: Learning optimal and fair decision trees for non-discriminative decision-making. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019. pp. 1418–1426. AAAI Press (2019)
2. Aghaei, S., Gómez, A., Vayanos, P.: Learning optimal classification trees: Strong max-flow formulations. CoRR **abs/2002.09142** (2020), https://arxiv.org/abs/2002.09142
3. Aglin, G., Nijssen, S., Schaus, P.: Pydl8.5: a library for learning optimal decision trees. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. pp. 5222–5224. ijcai.org (2020)

4. Aglin, G., Nijssen, S., Schaus, P.: Assessing optimal forests of decision trees. In: 33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021, Washington, DC, USA, November 1-3, 2021. pp. 32–39. IEEE (2021)
5. Aglin, G., Nijssen, S., Schaus, P.: Learning optimal decision trees under memory constraints. In: European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD) (2022)
6. Bayardo, R., Goethals, B., Zaki, M.J. (eds.): FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004, CEUR Workshop Proceedings, vol. 126. CEUR-WS.org (2005), http://ceur-ws.org/Vol-126
7. Bertsimas, D., Dunn, J.: Optimal classification trees. Mach. Learn. **106**(7), 1039–1082 (2017)
8. Boulicaut, J., Dzeroski, S. (eds.): Proceedings of the Second International Workshop on Inductive Databases, 22 September, Cavtat-Dubrovnik, Croatia. Rudjer Boskovic Institute, Zagreb, Croatia (2003)
9. Boutilier, J.J., Michini, C., Zhou, Z.: Shattering inequalities for learning optimal decision trees. In: Schaus, P. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 19th International Conference, CPAIOR 2022, Los Angeles, CA, USA, June 20-23, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13292, pp. 74–90. Springer (2022)
10. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees. Chapman and Hall/CRC (1984)
11. Demirovic, E., Lukina, A., Hebrard, E., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K., Stuckey, P.J.: Murtree: Optimal decision trees via dynamic programming and search. J. Mach. Learn. Res. **23**, 26:1–26:47 (2022)
12. Demirovic, E., Stuckey, P.J.: Optimal decision trees for nonlinear metrics. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021. pp. 3733–3741. AAAI Press (2021)
13. Fromont, É., Blockeel, H., Struyf, J.: Integrating decision tree learning into inductive databases. In: Dzeroski, S., Struyf, J. (eds.) Knowledge Discovery in Inductive Databases, 5th International Workshop, KDID 2006, Berlin, Germany, September 18, 2006, Revised Selected and Invited Papers. Lecture Notes in Computer Science, vol. 4747, pp. 81–96. Springer (2006)
14. Goethals, B., Zaki, M.J. (eds.): FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA, CEUR Workshop Proceedings, vol. 90. CEUR-WS.org (2003), http://ceur-ws.org/Vol-90
15. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. ACM sigmod record **29**(2), 1–12 (2000)
16. Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is np-complete. Inf. Process. Lett. **5**(1), 15–17 (1976)
17. Jeong, H., Wang, H., Calmon, F.P.: Fairness without imputation: A decision tree approach for fair prediction with missing values. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022. AAAI Press (2022)
18. Kiossou, H., Schaus, P., Nijssen, S., Ratheil, V.: Time constrained dl8.5 using limited discrepancy search. In: European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD) (2022)
19. Klemettinen, M., Meo, R. (eds.): Proceedings of the First International Workshop on Inductive Databases, 20 August 2002, Helsinki, Finland. Helsinki University Printing House, Helsinki (2002)

20. Nijssen, S., De Raedt, L.: IQL: A proposal for an inductive query language. In: Dzeroski, S., Struyf, J. (eds.) Knowledge Discovery in Inductive Databases, 5th International Workshop, KDID 2006, Berlin, Germany, September 18, 2006, Revised Selected and Invited Papers. Lecture Notes in Computer Science, vol. 4747, pp. 189–207. Springer (2006)
21. Nijssen, S., Fromont, É.: Mining optimal decision trees from itemset lattices. In: Berkhin, P., Caruana, R., Wu, X. (eds.) Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007. pp. 530–539. ACM (2007)
22. Nijssen, S., Fromont, É.: Optimal constraint-based decision tree induction from itemset lattices. Data Min. Knowl. Discov. **21**(1), 9–51 (2010)
23. Verwer, S., Zhang, Y.: Learning optimal classification trees using a binary linear program formulation. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019. pp. 1625–1632. AAAI Press (2019)