When a Decision Tree Learner Has Plenty of Time

Saher Esmeir and **Shaul Markovitch**
Computer Science Department
Technion—Israel Institute of Technology, Haifa 32000, Israel
{esaher , shaulm}@cs.technion.ac.il

Abstract

The majority of the existing algorithms for learning decision trees are greedy—a tree is induced top-down, making locally optimal decisions at each node. In most cases, however, the constructed tree is not globally optimal. Furthermore, the greedy algorithms require a fixed amount of time and are not able to generate a better tree if additional time is available. To overcome this problem, we present a lookahead-based algorithm for anytime induction of decision trees which allows trading computational speed for tree quality. The algorithm uses a novel strategy for evaluating candidate splits; a stochastic version of ID3 is repeatedly invoked to estimate the size of the tree in which each split results, and the split that minimizes the expected size is preferred. Experimental results indicate that for several hard concepts, our proposed approach exhibits good anytime behavior and yields significantly better decision trees when more time is available.

Introduction

"What is wanted is a resource constrained algorithm that will do the best it can within a specified computational budget... This would make a challenging thesis topic!" (Quinlan 1993, chap. 11)

The majority of the existing algorithms for learning decision trees, such as CART (Breiman *et al.* 1984), ID3 (Quinlan 1986), and C4.5 (Quinlan 1993) are greedy—a tree is induced top-down, making locally optimal decisions at each node. While performing quite well for many learning tasks, for many other hard concepts the greedy methods fail to successfully learn a good tree. Furthermore, they are not able to exploit additional time resources for building better trees.

In many applications that deal with hard problems, we are ready to allocate much more resources than required by simple greedy algorithms, but still cannot afford algorithms of exponential complexity. One commonly proposed approach for hard problems is anytime algorithms (Boddy & Dean 1994), which can trade computation speed for quality. There are two main classes of anytime algorithms, namely *contract* and *interruptible* (Russell & Zilberstein 1991). A contract algorithm is one that gets its resource allocation as a parameter. If interrupted before the allocation is completed, it might not yield any useful results. An interruptible algorithm is one whose resource allocation is not given in advance and thus must be prepared to be interrupted and return a valid solution at any moment. In this paper we describe and motivate LSID3, a contract anytime algorithm for inducing decision trees that has recently been introduced (Esmeir & Markovitch 2004).

Contract Anytime Induction of Decision Trees

The most common method for learning decision trees is top-down induction: start from the entire set of training examples, partition it into subsets by testing the value of an attribute, and then recursively call the induction algorithm for each subset. Our proposed anytime approach adopts the general top-down scheme and invests additional time resources for making better split decisions.

One way to exploit additional time is to evaluate an attribute by measuring gain-k, the information gain it yields at depth k below the current node, instead of gain-1 as in ID3. This approach was the basis for the IDX algorithm (Norton 1989). We refer to this lookahead-based variation of ID3 as *ID3-k*. At each node, ID3-k chooses to split on the attribute that maximizes gain-k. ID3-k can be viewed as a contract algorithm parameterized by k. However, despite its ability to exploit additional resources when available, the anytime behavior of ID3-k is problematic. The run time of ID3-k grows exponentially as k increases, limiting the flexibility of the algorithm. Furthermore, it is quite possible that looking ahead to depth k will not be sufficient to correctly estimate the usefulness of a split. Invoking exhaustive lookahead will obviously lead to optimal splits but its computational costs are prohibitively high. In what follows we describe LSID3, a novel algorithm for choosing split attributes that overcomes the above-mentioned drawbacks of ID3-k.

Following Occam's razor, smaller decision trees are likely to be more accurate. LSID3 evaluates candidate splits by directly estimating the size of the minimal tree under each one and prefers the split with the smallest associated tree. For an attribute a, the estimation is based on Monte-Carlo sampling of the space of consistent trees rooted at a. We estimate the (unknown) minimum by the size of the smallest tree in the sample. The number of trees in the sample depends on the available resources, and the quality of the estimation is ex-

```
Procedure LSID3-Choose-Attribute($E, A, r$)
    If $r = 0$ Return ID3-Choose-Attribute(E, A)
    Foreach $a \in A$
        Foreach $v_i \in \mathrm{domain}(a)$
            $E_i \leftarrow \{e \in E \mid a(e) = v_i\}$
            $min_i \leftarrow \infty$
            Repeat $r$ times
                $T \leftarrow \mathrm{SID3}(E_i, A - \{a\})$
                $min_i \leftarrow \min(min_i, \mathrm{Size}(T))$
        $total_a \leftarrow \sum_{i=1}^{|\mathrm{domain}(a)|} min_i$
    Return $a$ for which $total_a$ is minimal
```

Figure 1: Attribute selection in LSID3

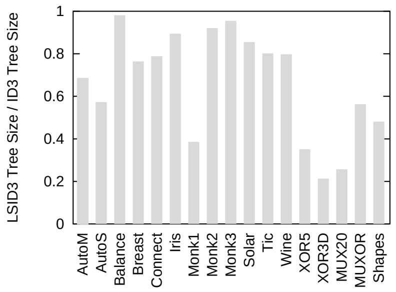

Figure 2: The relative size of the trees produced by LSID3 in comparison to ID3

pected to improve with the increased sample size. To obtain the sample, we designed a stochastic version of ID3, called *SID3*. In SID3, rather than choosing an attribute that maximizes the information gain (as in ID3), the splitting attribute is chosen semi-randomly. The likelihood that an attribute will be chosen is proportional to its information gain.

To evaluate an attribute a, LSID3 partitions the set of examples according to the values a takes and repeatedly invokes SID3 to sample the space of trees consistent with each subset. Summing up the minimal tree size for each subset gives an estimation of the minimal total tree size under a. LSID3 is a contract algorithm parameterized by r, the sample size. When $r = 0$ LSID3 is defined to be identical to ID3. Figure 1 formalizes the attribute selection in LSID3.

Experimental Results

A variety of experiments were conducted to test the performance and anytime behavior of LSID3. The first set of experiments compares ID3, ID3-k($k = 2$), C4.5, and LSID3($r = 5$), in terms of generalization accuracy and size (number of leafs) of the induced trees. Several UCI (Blake & Merz 1998) and other relatively hard synthetic datasets were used. The reported results are the average of 10 runs of 10-fold cross-validation experiment (Bouckaert 2003). Figure 2 gives the differences in tree size of LSID3(5), and ID3. Figure 3 compares the accuracy of LSID3 to that of ID3, ID3-k, and C4.5. The full results, including significance tests, are available in (Esmeir & Markovitch 2004).

In all cases, as Figure 2 implies, LSID3 produced smaller

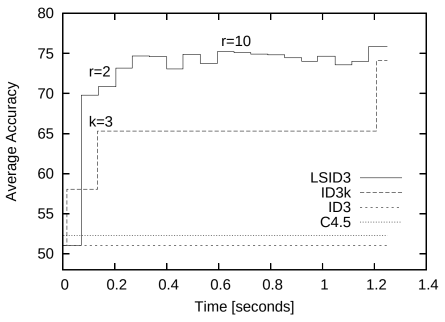

Figure 4: Results for the Multiplex-XOR dataset

trees than ID3 and these improvements were found to be significant. In most cases, the LSID3 trees were also smaller than the trees produced by ID3-k. Because the trees of C4.5 are pruned, their size is not comparable to that of LSID3 that produces consistent trees. Reducing the tree size is usually beneficial only if the associated accuracy is not reduced. Analyzing the accuracy of the produced trees, as plotted in Figure 3, shows that LSID3 outperforms ID3, ID3-k and C4.5 for most datasets.

To test the anytime behavior of LSID3 and ID3-k, we invoked LSID3 and ID3-k with different values of r and k. Figure 4 shows the average results for the Multiplex-XOR dataset. The X axis represents the run time in seconds. ID3 and C4.5, which are constant time algorithms, finish quickly and do not improve with time. We can see that the accuracy of both anytime algorithms improves with time. This improvement is larger at the beginning and diminishes over time. The anytime behavior of LSID3 is better in comparison to ID3-k. For ID3-k, the gaps in time between the points grow exponentially, although successive values of k were used. As a result, any extra time budget that falls into one of these gaps cannot be exploited. For LSID3, the difference in the run-time for any 2 successive values of r is almost the same. Except of a small period of time, ID3-k is dominated by LSID3. Additional anytime graphes that demonstrate the improvement in accuracy and tree size are available in (Esmeir & Markovitch 2004).

Why Anytime Decision Tree Learners?

LSID3, the core result of (Esmeir & Markovitch 2004), tackles the problem of anytime induction of decision trees. In this section we motivate LSID3 and discuss its relevance to AI. We first describe the advantages of decision trees in the context of AI applications. We then refer to a major limitation of greedy induction algorithms—their inability to exploit additional resources, and suggest the anytime approach to overcome this problem. Finally we list several AI applications that can benefit from our proposed anytime framework.

Advantages of Decision Trees

Despite the recent progress in advanced induction algorithms such as SVM (Vapnik 1995), *decision trees* are still considered attractive for many real-life applications, mostly

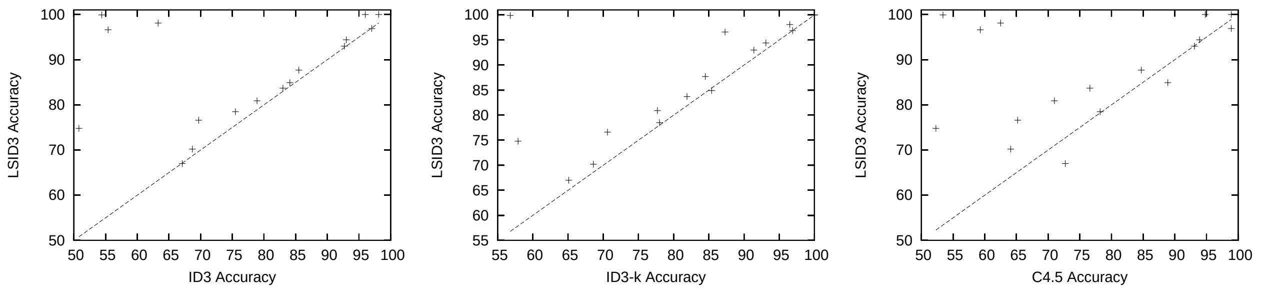

Figure 3: Accuracy comparison for LSID3(5) vs. ID3 (left), ID3-k (center) and C4.5 (right). Each point represents a dataset. The y-axis represents the accuracy of LSID3 while the x-axis represents that of the 2*nd* compared algorithm. The dashed line indicates equality. Points are above it if LSID3 performs better and below it if it is worse.

due to their interpretability (Hastie, Tibshirani, & Friedman 2001, chap. 9). Craven (1996) lists several reasons why the understandability of a model by humans is an important criterion for evaluating it:

- *Validation:* When a classifier is comprehensible, its users can know how it makes decisions. This can make it more trustworthy and increase user confidence.
- *Discovery:* Analyzing a learnt hypothesis can lead the user to discover unrecognized relationships between the different features.
- *Explanation:* Understandable models can ease the task of explaining the classification of particular instances.
- *Improving predictive accuracy:* Decision trees, due to their interpretability, allow analysis of the learned model, which may lead to a better feature representation and as a result improve predictive accuracy.

As the use of machine learning techniques becomes widespread, many artificial intelligence systems for supporting decision-making are being applied to critical matters, such as disease diagnosis and survivability prediction (Kononenko 2001; Delen, Walker, & Kadam 2005), legal reasoning (Aleven & Ashley 1997), economics (Ghani 2005), and computer security (Maloof 2005). Without the ability to understand how the system makes decisions, human professionals cannot trust it, and therefore will not use it for practical problems. The comprehensibility of decision trees makes it easier to pass this barrier.

Another model evaluation criterion mentioned by Craven (1996) is the flexibility of representation. Under this criterion, decision trees have a great advantage: they can be easily converted into an easily modifiable set of logical rules. When classification cost is an important factor, decision trees may be attractive in that they ask only for the values of the features along a single path from the root to a leaf. In terms of accuracy, decision trees were shown to be competitive with other classifiers for several learning tasks.

Anytime Algorithms: Between Greedy and Exponential

The majority of the existing algorithms for learning decision trees build a decision tree top down. The space of possible decision trees obtainable from this procedure is huge and a major question is what strategy should be followed to direct the search. Most algorithms follow Occam's razor and take simplicity as their *preference bias* (Mitchell 1997, chap. 3). Smaller decision trees have several advantages in addition to their probable greater accuracy, such as greater statistical evidence at the leaves, better comprehensibility, lower classification costs, and low storage costs.

Finding the smallest consistent decision tree is an NP-complete problem (Murphy & McCraw 1991). Therefore, most existing algorithms take a greedy approach and use local heuristics to evaluate the benefit from a split. The greedy methods perform quite well for many learning tasks. One problem with them, however, is their inability to exploit additional learning time. This problem is intensified for hard concepts, where the greedy methods fail to produce good models. For example, in our experiments with 17 UCI and synthetic datasets, we found that ID3 and C4.5 finished building a tree in less than 0.1 second for 16 tasks and less than 4 seconds in the slowest case. However, the performance of ID3 and C4.5 on hard concepts, such as parity, was very poor and could not be improved if more time were available. Parity-like concepts naturally arise in real-world data, such as the Drosophila Survival concept (Page & Ray 2003). When applied on some of the 17 datasets, ID3-k with exhaustive lookahead, that can find optimal trees, did not finish even after few days.

Similar situations, where the solution can be either of exponential complexity and therefore prohibitively expensive or non-optimal and fast, are common in many AI domains, such as search, games, and constraint satisfaction problems. An interesting question is what to do when one has more time than required by the greedy solution but cannot afford the exponential one. Anytime algorithms (Boddy & Dean 1994), which can trade computation speed for solution quality, provide means of obtaining better solutions than greedy algorithms within a reasonable resource budget. Anytime techniques have been applied to several AI domains, such as heuristic search, CSPs, and SAT solvers (Wallace & Freuder 1995; Hansen, Zilberstein, & Danilchenko 1997).

AI Applications that Can Benefit from LSID3

The need for anytime learning algorithms arises in many real-life artificial intelligence applications that involve offline learning. For example:

- *Medical tasks:* Assume that a medical center has deiced to acquire a classifier for diagnosis of a particular disease (e.g., (Kononenko 2001)). The center applies C4.5 on thousands of records from previous patients, and after few seconds receives a decision tree. During the coming months, or even years, the same induced decision tree will be used. Obviously, the medical center is willing to wait much longer to obtain a better tree.
- *Planning:* Consider an intelligent planning agent that has to take action rapidly (e.g., (Konur, Ferrein, & Lakemeyer 2004)). This requires better long-term planning. In such cases the agent is ready to invest more resources for the learning phase to improve its real-time performance.
- *Text categorization:* Consider a machine-learning based spam filter (e.g., (Cohen 1996)). As users, we expect the learning component to exploit the time between the arrival of new messages.
- *Computational finance:* Consider an artificial investor that uses machine learning techniques for stock portfolio selection (e.g., (Borodin, El-Yaniv, & Gogan 2004)). If we chose to update the portfolio at specific time points, then we would like the learner to exploit the time between these updates.

In this paper we summarized and motivated LSID3, an anytime algorithm for decision tree induction. LSID3 was shown to exhibit good anytime behavior and to yield significantly better decision trees when more time is available. By using additional resources, LSID3 was able to learn several hard concepts, such as XOR and Multiplexer, that were very difficult for most existing decision tree learners. We described several AI applications where the user is willing to wait longer than the time needed by greedy induction algorithms if a better decision tree can be produced.

Acknowledgements

This work was partially supported by funding from the EC-sponsored MUSCLE Network of Excellence (FP6-507752).

References

Aleven, V., and Ashley, K. D. 1997. Evaluating a learning environment for case-based argumentation skills. In *ICAIL'97*, 170–179. New York, NY, USA: ACM Press.

Blake, C. L., and Merz, C. J. 1998. UCI repository of machine learning databases.

Boddy, M., and Dean, T. L. 1994. Deliberation scheduling for problem solving in time constrained environments. *Artificial Intelligence* 67(2):245–285.

Borodin, A.; El-Yaniv, R.; and Gogan, V. 2004. Can we learn to beat the best stock. *Journal of Artificial Intelligence Research* 21:579–594.

Bouckaert, R. R. 2003. Choosing between two learning algorithms based on calibrated tests. In *ICML'03*, 51–58.

Breiman, L.; Friedman, J.; Olshen, R.; and Stone, C. 1984. *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Cohen, W. W. 1996. Learning to classify English text with ILP methods. In *Advances in Inductive Logic Programming*. IOS Press. 124–143.

Craven, M. W. 1996. *Extracting Comprehensible Models from Trained Neural Networks*. Ph.D. Dissertation, University of Wisconsin, Madison.

Delen, D.; Walker, G.; and Kadam, A. 2005. Predicting breast cancer survivability: a comparison of three data mining methods. *Artificial Intelligence in Medicine* 34(2):113–127.

Esmeir, S., and Markovitch, S. 2004. Lookahead-based algorithms for anytime induction of decision trees. In *ICML'04*, 257–264.

Ghani, R. 2005. Price prediction and insurance for online auctions. In *KDD'05*, 411–418.

Hansen, E. A.; Zilberstein, S.; and Danilchenko, V. A. 1997. Anytime heuristic searc: First results. Technical Report UM-CS-1997-050.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag.

Kononenko, I. 2001. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine* 23(1):89–109.

Konur, S.; Ferrein, A.; and Lakemeyer, G. 2004. Learning decision trees for action selection in soccer agents. In *Workshop on Agents in dynamic and real-time environments, with ECAI'04*.

Maloof, M. A. 2005. *Machine Learning and Data Mining for Computer Security*. New York: Springer-Verlag.

Mitchell, T. M. 1997. *Machine Learning*. McGraw Hill.

Murphy, O. J., and McCraw, R. L. 1991. Designing storage efficient decision trees. *IEEE Transactions on Computers* 40(3):315–320.

Norton, S. W. 1989. Generating better decision trees. In Sridharan, N. S., ed., *IJCAI'89*, 800–805.

Page, D., and Ray, S. 2003. Skewing: An efficient alternative to lookahead for decision tree induction. In *IJCAI'03*.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Russell, S. J., and Zilberstein, S. 1991. Composing real-time systems. In *IJCAI'91*, 212–217.

Vapnik, M. A. 1995. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.

Wallace, R. J., and Freuder, E. C. 1995. Anytime algorithms for constraint satisfaction and SAT problems. In *Working Notes of IJCAI-95 Workshop on Anytime Algorithms and Deliberation Scheduling*.