Analysis: Proc. 3rd International Symposium (Amsterdam, The Netherlands).

The Biases of Decision Tree Pruning Strategies

Tapio Elomaa

Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland
elomaa@cs.helsinki.fi

Abstract. Post pruning of decision trees has been a successful approach in many real-world experiments, but over all possible concepts it does not bring any inherent improvement to an algorithm's performance. This work explores how a PAC-proven decision tree learning algorithm fares in comparison with two variants of the normal top-down induction of decision trees. The algorithm does not prune its hypothesis *per se*, but it can be understood to do pre-pruning of the evolving tree. We study a backtracking search algorithm, called *Rank*, for learning rank-minimal decision trees. Our experiments follow closely those performed by Schaffer [20]. They confirm the main findings of Schaffer: in learning concepts with simple description pruning works, for concepts with a complex description and when all concepts are equally likely pruning is injurious, rather than beneficial, to the average performance of the greedy top-down induction of decision trees. Pre-pruning, as a gentler technique, settles in the average performance in the middle ground between not pruning at all and post pruning.

1 Introduction

Pruning trades accuracy of the data model on the training data with syntactic simplicity in the hope that this will result in a better expected predictive accuracy on the unseen instances of the same domain. Pruning is just one *bias*—preference strategy—among a large body of possible ones, and it only works when it is appropriate from the outset. Schaffer [20] demonstrated that it is a misconception that pruning a decision tree, as compared to leaving it intact, would yield better expected generalization accuracy except for a limited set of target concepts. Suitably varying the setting of the learning situation gives rise to the better performance of the strategy which does not prune the tree at all. In particular, according to Schaffer's [20] experiments pruning a decision tree when classification noise prevails degrades the performance of decision tree learning.

We explore the practical value and generality of another kind of decision tree learning bias; one that is not geared towards avoiding overfitting nor minimizing the complexity of the resulting decision tree. In this approach one characteristic figure of the data model is, however, optimized. Inspired by the theoretical results of Ehrenfeucht and Haussler [6], we test how optimizing a secondary size and shape related parameter—the *rank* of a decision tree—affects the utility of the resulting bias. By changing the values of the input parameters of the learning algorithm called *Rank* [7] we are tuning the fitting of the evolving decision tree to the training data. This is, in a sense, tempering with the level of pruning, but it happens in a controlled way: the fitting is taken into account already in growing the decision tree, when choosing the attributes and deciding the tree's shape. The combined data fit endorsement and model complexity penalization yields, in Domingos' [3, 5] terms, *representations-oriented evaluation*.

The rank-optimizing algorithm works by backtracking, i.e., the algorithm may choose to revoke decisions it has made previously. The advantage of optimizing a size-related parameter is that the learning algorithm cannot suffer from pathology that affects decision tree learning algorithms using lookahead [12] nor is it as sensitive to the size of the training data as standard greedy top-down induction of decision trees [13]. Practical experience has shown the rank of a decision tree to be a very stable measure [7].

Elomaa and Kivinen [8] and Sakakibara [19] proved that rank-bounded decision trees can be learned—within the PAC framework of Valiant [1, 22]—when classification noise prevails. The erroneous examples dictate that the fitting of the decision tree cannot be perfect. Relaxed fitting is obtained by pruning, not following full tree construction, but rather in *pre-pruning* fashion. Pre-pruning is a familiar technique from the early empirical decision tree learning algorithms like ID3 [14, 15]. Recently it has, again, been advocated as an alternative to post pruning of decision trees [10, 4]. Pre-pruning also implements early stopping of training [23], which aims to prevent overtraining and, through it, overfitting.

Our experiments confirm, in a slightly different setting, what was already reported by Shaffer [20]: over the space of equiprobable random Boolean concepts the *Naive* strategy to decision tree learning, which does not even attempt to prune the decision tree built, outperforms the *Sophisticated* one, which may choose to prune the tree, independent of the classification noise level affecting the domain. The latter strategy, though, has constantly obtained better results in empirical experiments. One opposite trend to those reported by Schaffer comes up in our experiments; viz. as the noise rate approaches maximal .5, the better performances start to even out between the learning strategies, rather than to steadily pile up in favor of the Naive strategy. This result is what one would intuitively expect.

The algorithm *Rank* has been previously observed to be competitive with other decision tree learning approaches on UCI data sets [7]. Our experiments here further confirm that in learning simple concepts without large amounts of noise prevailing, rank-minimization attains as good results as the straightforward top-down induction of decision trees. However, on concepts that are harder to express by decision trees, *Rank* cannot match the performance of the Naive strategy. As the noise rate increases, the rank-minimization algorithm catches up the advantage the Naive strategy possessed initially. In comparison with the Sophisticated strategy *Rank* steadily outperforms it.

In Section 2 we introduce briefly the learning of rank-bounded decision trees. In Section 3 a set of empirical experiments—inspired by those performed by

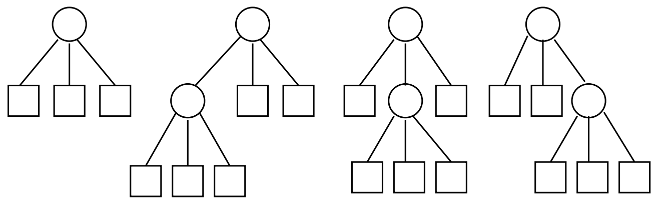

Fig. 1. Tree structures of rank 1 in $\mathrm{DT}_3^1(2)$.

Schaffer [20]—is reported and discussed. Section 4 considers the outcome and implications of the experiments. Section 5 reviews further related research. Finally, Section 6 presents the conclusions of this study.

2 Learning decision trees of minimum rank

Ehrenfeucht and Haussler [6] showed that the function class represented by rank-bounded binary decision trees is learnable in the sense of the basic PAC model [22]. Its superclass—functions determined by arbitrary binary decision trees—is not learnable [9].

There is only one natural way to extend the original definition of the rank to general multivalued decision trees without losing the learnability property.

Definition 1. *The* rank *of a decision tree T, denoted by $r(T)$, is defined as:*

1. *If T consists of a single leaf, then $r(T) = 0$.*
2. *Else if $T_{\max}$ is a subtree of T with the maximum rank $r_{\max}$, then*

$$r(T) = \begin{cases} r_{\max} & \text{if } T_{\max} \text{ is unique,} \\ r_{\max} + 1 & \text{otherwise.} \end{cases}$$

This definition generalizes strictly that of Ehrenfeucht and Haussler [6]: The rank of a binary tree is the same as the rank of the subtree with a higher rank if the two subtrees have different ranks. Otherwise, the rank of the full tree is the rank of its both subtrees incremented by one.

Example 1. Let $\mathrm{DT}_m^r(n)$ denote the set of all m-ary decision trees of rank at most r over n variables. We are concerned with *reduced* decision trees, where each variable appears at most once on any path from the root to a leaf. Observe that in a reduced tree n is the maximum rank of the tree.

Delimiting the rank of a decision tree (together with the value m) determines the tree structures in $\mathrm{DT}_m^r(n)$. $\mathrm{DT}_m^0(n)$ only contains the single-leaf tree structure, independent of the value of m. The number of possible labelings of that only leaf gives the number of functionally different equal-structured decision trees; $\mathrm{DT}_m^0(n)$ always contains n separate one-leaf decision trees. For values $r > 0$ there exists more than just one possible tree structure (assuming $n > 1$).

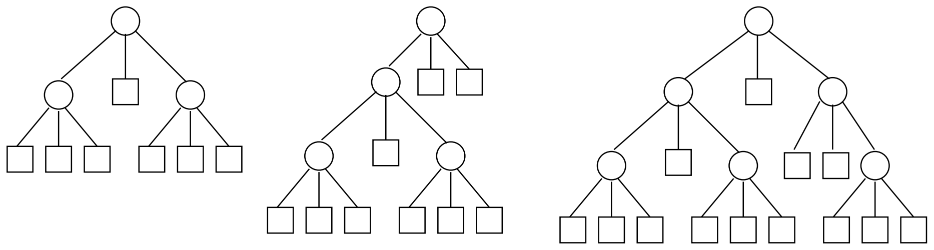

Fig. 2. Examples of tree structures in $\mathrm{DT}_3^2(3)$.

For instance, Fig. 1 illustrates all (reduced) tree structures of rank 1 contained in $\mathrm{DT}_3^1(2)$.

The decision trees are obtained from these structures by assigning each node with a label; the labeling must be legal, i.e., it has to keep the tree reduced. The function represented by a decision tree is not necessarily unique.

As the parameter values m, n, and r grow, the number of legal tree structures in $\mathrm{DT}_m^r(n)$ and their possible labelings go up quickly. The tree structures in Fig. 2 are examples of those belonging to $\mathrm{DT}_3^2(3)$.

In order to be able to construct decision trees of minimum rank in noisy domains one has to relax the fitting of examples by letting the decision tree give inconsistent decisions. Typically in top-down induction of decision trees as tightly fitted decision trees as possible are constructed and then, in the second phase, they are pruned back in order to avoid overfitting them to the false trends that happen to prevail in the training set [2, 16]. Obviously, such pruning cannot be used in connection of rank-minimization.

In the algorithm *Rank* growing of a branch of the evolving tree is stopped when relaxed fitting is reached. Two *a priori* determined parameters control the degree of the required fitting. The relaxed fitting amounts to pre-pruning. The construction of a branch of the evolving decision tree is stopped when only a small portion of the examples under consideration deviates from the majority class or a minimal number of the examples belongs to a different class than the majority one [8, 19]. The values for the input parameters determining the relative portion of consistent examples required (parameter γ) and the absolute number of errors allowed per class (parameter κ) before stopping the growing of a branch can be calculated exactly from the values of the standard PAC-parameters sample size, accuracy ε, confidence δ, and noise rate η [1, 22]. The three latter parameters, of course, are unknown in practice.

We take γ and κ simply to be a real-valued and an integer-valued, respectively, input parameter for the learning algorithm. It is typical in inductive learning methods to expect the user to supply a value for a confidence level or a threshold parameter, which corresponds to γ parameter of *Rank*. For instance, C4.5 [16] is an example of such a program. Furthermore, in C4.5 the user is allowed to tune the value of a parameter (`-m`) that corresponds to κ.

Table 1. The programs implementing the relaxed rank-minimization. *StoppingCondition* is the relaxed fitting rule that implements pre-pruning. *Find* is the subprogram that checks, by backtracking search, if a decision tree of the given rank exists. *Rank* is the main program controlling the value of the rank bound.

```
boolean StoppingCondition( Sample S, int $\kappa$, double $\gamma$ )
{
   // $M_i$ is the number of instances of $i$ in $S$. $k$ is the majority class in $S$.
   if ( $M_k \geq \gamma|S|$ or $M_j \leq \kappa$ for all $j \neq k$ (and $M_k > \kappa$) ) return true;
   return false;
}

DecisionTree Find( Sample $S$, int $r$, Variables $V$, int $\kappa$, double $\gamma$ )
{
   if ( StoppingCondition( $S$, $\kappa$, $\gamma$ ) ) return $T = k$;   // $k$ is the majority class in $S$.
   if ( $r = 0$ ) return none;
   for ( each informative variable $v \in V$ ) {
      for ( each $k \in [m]$ ) $T_k^v \leftarrow Find(S_k^v, r-1, V \setminus \{v\}, \kappa, \gamma)$;
      if ( $\forall k \in [m] : T_k^v \neq$ none )
         {$T \leftarrow MakeTree(v, T_1^v, \ldots, T_m^v)$; return $T$;}
      if ( $T_k^v =$ none for a single value $k = \ell \in [m]$ ) {
         $T_\ell^v \leftarrow Find(S_\ell^v, r, V \setminus \{v\}, \kappa, \gamma)$;
         if ( $T_\ell^v \neq$ none ) $T \leftarrow MakeTree(v, T_1^v, \ldots, T_m^v)$;
         else $T \leftarrow$ none;
         return $T$;
      }
   }
   return none;
}

DecisionTree Rank( Sample $S$, Variables $V$, int $R$, int $\kappa$, double $\gamma$ )
{
   $r \leftarrow R$; $T \leftarrow Find(S, R, V, \kappa, \gamma)$;
   if ( $T =$ none ) {
      repeat { $r \leftarrow r + 1$; $T \leftarrow Find(S, r, V, \kappa, \gamma)$; }
      until $T \neq$ none or $r = |V|$;
      if ( $T =$ none ) $T \leftarrow T = k$;   // $k$ is the majority class in $S$.
      return $T$;
   }
   $r \leftarrow r(T)$;
   while ( $r > 0$ ) {
         $Q \leftarrow Find(S, r-1, V, \kappa, \gamma)$;
         if ( $Q \neq$ none ) { $r \leftarrow r(Q)$; $T \leftarrow Q$; }
   }
   return $T$;
}
```

The programs implementing the learning algorithm *Rank* are described as code in Table 1. The main program *Rank* inputs five parameters: the training sample S, the variable set V, the initial rank candidate R, and the values for the parameters κ and γ. Depending on whether a decision tree is found by the subprogram *Find* using the initial rank candidate or not, a search proceeding to different directions for the true rank of the sample has to be performed. If, at the end, no decision tree is found—due to too tight fitting requirements—a single-leaf tree predicting the majority class of the sample is returned.

In the program *Find* it is assumed that all the variables in V are nominal and have the same arity; i.e., that their domain consists of the value set $[m] = \{1, \ldots, m\}$. This assumption is included here for the clarity of the code, it is not a restrictive assumption. *Find* carries out the backtracking search for a decision tree of at most the given rank bound. Observe that this search is not exhaustive—which would require exponential time—but avoids unnecessary recursion by carefully keeping track of the examined rank candidates. The asymptotic time requirement of this algorithm is linear in the size of the sample and exponential in the rank of the sample. However, the rank of the sample is constant, and usually very low in practice; hence, the algorithm is feasible. By S_k^v we denote the set of those examples in the sample S in which variable v has value k.

Finally, *StoppingCondition* checks whether the relaxed stopping condition holds in the sample under consideration. In other words, it returns value true if the conditions of pre-pruning are fulfilled and branch growing can be terminated.

3 Empirical evaluation

We carry out a set of experiments inspired by those of Schaffer [20]. In them two strategies to decision tree learning implemented by the CART learning algorithm [2] were compared: Naive strategy chose the decision tree grown without a chance of ever pruning it and the Sophisticated strategy had the possibility of pruning the tree produced. The purpose of duplicating a part of these experiments is to assess the overall performance of the *Rank* algorithm and its pre-pruning strategy in contrast with two versions of the more familiar greedy top-down induction procedure.

3.1 Experiment setting

Each experiment consists of 25 trials. We take all the trials into account, but pay special attention to the *discrepant trials*, those in which the the trees of the strategies differ in their accuracy. All experiments concern learning of Boolean functions on five attributes: named `a` through `e`. The number of training examples given to the learning algorithms is 50 randomly allotted instances. The prediction accuracy of a tree is analytically determined.

In testing the statistical significance of our findings we follow Schaffer [20] and use the non-parametric one-sided binomial sign test. Given n discrepant trials in an experiment, the test lets us reject the hypothesis that the strategies perform equally well with confidence $1 - \sum_{i=0}^{r} \left(\binom{n}{i}/2^n\right)$, where r, $0 \leq r \leq \lfloor n/2 \rfloor$, is the number of "wins" recorded by the strategy obtaining less wins. All trials, except when otherwise stated, have a .1 classification noise affecting them; i.e., each training example has a one tenth chance of having a corrupted (complemented) class value.

Instead of CART we use C4.5 (release 8) algorithm [16, 17] to implement the basic learning strategies. As Schaffer [20] argues nothing should change through changing the top-down induction algorithm; the relative strengths and weaknesses of the strategies should stay constant. Sophisticated strategy is the default pruning of C4.5 and in the Naive strategy pruning has been turned off.

In C4.5, which uses global post pruning, the performance of the final decision tree is less sensitive to the values of the input parameters than that of *Rank*, built based on local pre-pruning. For *Rank* we adjust the values of its parameters κ and γ to suitable values empirically, but then keep the values constant throughout each experiment (25 trials). Only when large amounts of noise affect the learning situation will κ have a value different from 1. We do not leave the order of attribute inspection to be arbitrary as described in the code in Table 1. Instead, we use the evaluation function gini-index [2] to order the attributes. Nevertheless, the rank restriction ultimately decides whether an attribute is chosen to the tree or not.

3.2 Preliminary experiments

Schaffer [20] started with two simple learning tasks, where the Sophisticated strategy emerged superior. In learning the first simple concept $\texttt{Class} = \texttt{a}$, there were 18 of 25 discrepant trials and in each of them the Sophisticated strategy was superior obtaining, indeed, the maximum achievable average accuracy of .9.

The learning algorithm *Rank* also attains the maximum achievable accuracy in each of the 25 trials using parameter value $\gamma = .75$. In this case the correct concept is a decision tree containing (at least) two leaves, which have to accommodate also all the corrupted examples. Therefore, relatively loose fitting is required for learning the correct concept.

In learning the second concept $\texttt{Class} = \texttt{a} \vee (\texttt{b} \wedge \texttt{c})$, there were 17 discrepant trials in Schaffer's experiment. The Sophisticated strategy chose a superior tree in 16 out of them, attaining the average accuracy of .854.

This time a correct decision tree contains at least four leaves. Thus, the fitting of the tree does not need to be as relaxed as in the previous experiment. Using parameter value $\gamma = .85$ *Rank* comes up with the correct concept in 18 of 25 trials attaining the average accuracy of .876.

In sum, these two simple concepts are at least as well learned by the backtracking rank-minimization as by growing a full decision tree and subsequently pruning it. This result supports our earlier finding—on the UCI data—that *Rank* performs comparably with C4.5 on the most commonly used machine learning test domains [7].

3.3 Parity and random functions

Parity function is true precisely when an odd number of the attributes take on the true value. It is hard to express by any representation using only single-attribute values. Parity serves to demonstrate the opposite case in Schaffer's [20] experiments: for it the fully-grown trees are more accurate than the pruned decision trees. Moreover, adding more noise to the training examples allows the Naive strategy to increase its lead over the Sophisticated strategy.

In the basic .1 classification noise case, when the C4.5 algorithm was used to implement the two strategies, the Naive strategy outperformed the Sophisticated one in each trial. The average accuracy of the Naive strategy over the 25 trials was .675 and that of the Sophisticated strategy was .542.

Parity is a worst-case function for rank-minimization since the correct decision tree for it has as high rank as possible. Therefore, any other matching concept of lower rank for the noisy training set is preferred over the true one. In light of this, one cannot expect *Rank* to attain the accuracy level of the Naive strategy. Using parameter value $\gamma = .9$ *Rank* obtained the average accuracy of .552. The number of discrepant cases with the Sophisticated strategy was 19 of 25, out of which in 12 trials *Rank* produced the superior decision tree. Naive strategy was superior to *Rank* in all trials.

A better understanding of the strategies' relative performance is obtained by repeating Schaffer's experiment of learning a random Boolean function on five attributes. In each trial we randomly fix a new target function. Parameter values for *Rank* need to be adjusted differently for each target function.

The number of discrepant trials between *Rank* and the Naive strategy is 23, out of which in 22 Naive produces the superior tree. *Rank* outperforms the Sophisticated strategy in 14 of 16 discrepant trials. The average accuracies over the 25 trials are .777 for the Naive strategy, .728 for the *Rank* algorithm, and .701 for the Sophisticated strategy.

These experiments go to show that when the correct decision tree has a syntactically complex tree description, close fitting of the tree to the training examples gives the best results. Learning random Boolean concepts demonstrates that the majority of concepts falls into this category.

3.4 The effect of classification noise

The experiments reported by Schaffer [20] that bear most generality concern learning random Boolean functions. These experiments correspond to normal randomization analysis and give a measure of the average performance when all concepts are equally likely. However, as often pointed out [11, 18, 20], this does not necessary conform to real-world performance of the strategies.

Schaffer observed that as the classification noise rate of the training examples increases from .05 to .3, the number of discrepant trials steadily grows. The Sophisticated strategy is not able to conquer a larger proportion of those cases. Instead, the Naive strategy keeps advancing its superiority. This is a strong

Table 2. The effect of classification noise: random Boolean functions.

Error rate	Naive > Sophitic.		Naive > *Rank*		*Rank* > Sophistic.		Average accuracies		
							Naive	*Rank*	Sophistic.
.05	22 of 22	.99	19 of 20	.99	15 of 17	.99	.782	.738	.700
.1	23 of 23	.99	22 of 23	.99	14 of 16	.99	.777	.728	.701
.15	18 of 21	.99	15 of 20	.98	17 of 22	.99	.709	.691	.656
.2	21 of 22	.99	17 of 21	.99	18 of 21	.99	.695	.669	.622
.25	19 of 21	.99	12 of 19	.82	11 of 15	.94	.682	.663	.639
.3	15 of 21	.96	13 of 23	.66	12 of 19	.82	.625	.621	.584

demonstration of pruning lacking any inherent advantage in fighting against the effects of noise.

Table 2 lists the results of our experiment. The first column gives the noise rate. The next two record in how many out of the discrepant cases, the Naive strategy outperformed the Sophisticated one, and what is the confidence level by which we can trust the former to be superior in this experiment. Corresponding column pairs are given for the remaining two learning strategy pairs. The three last columns report the average accuracies of the algorithms over the 25 trials.

Our results are parallel to those reported by Schaffer; the Naive strategy is superior to the Sophisticated one independent of the error rate affecting the class attribute. However, there is also an opposite trend in here; when only a small error probability affects the learning situation the Naive strategy is with a very high confidence level significantly better than the Sophisticated strategy. As the error rate approaches .5, this confidence level starts to decline.

This effect is clearer in the comparison with the *Rank* algorithm. When .3 classification noise prevails, the confidence which we have for the Naive strategy's superiority is only .66 and the confidence for the superiority of *Rank* over the Sophisticated strategy has dropped down to .82. The decreasing confidences are due to the fact that the concept represented by the training examples approaches a random one. Thus, no data model can predict the class of an instance. The same can also be observed from the dropping average accuracies of all three strategies; they all gradually approach the random guess' accuracy .5.

In Schaffer's experiment the number of discrepant cases also increased along with the error rate. No such trend appears in the results of our experiment. In our experiment the noise rate as such has no bearing on the relative strengths of the learning strategies, none of them can be said to cope with classification noise better than the other strategies do.

3.5 Further experiments

Schaffer [20] further explored the effects of, e.g., changing the representation of the concepts, adding different kinds of noise to the training examples, and on the number of instances belonging to different classes. We, however, leave performing similar tests as future work. It is evident that the representation changes would have similar effects on our test strategies as they had on those of Schaffer.

4 Discussion

The above experiments confirmed what Schaffer [20] already reported: In learning syntactically simple concepts, pruning leads to more accurate decision trees than leaving the tree be as it is after growing it. There are concepts that require a syntactically more complex description. In learning such concepts it is better to do as accurate fitting of the decision tree to the training examples as possible and not prune the tree. Most importantly, the concepts requiring a complex decision tree description are the ones that dominate the space of all possible concepts.

The last point is crucial for the practical applicability of decision trees. However, the empirical evidence overwhelmingly supports the hypothesis that most real-world learning domains have a syntactically simple decision tree representation [11, 20]. There also exists some analysis to back up this claim [18].

An interesting difference in the results of the above comparison and those of Schaffer is that as the noise rate approaches the maximal .5 level, both the Naive strategy and the *Rank* algorithm start to lose the edge that they have over the Sophisticated strategy. This effect is intuitive; as the noise rate increases, the class associated with the instances tends towards a random assignment. Thus, it is impossible for any learning approach to predict the class of an instance. As Schaffer [20, pp. 163–165] analyses, exact fitting is the best prediction policy when all concepts are equiprobable. Therefore, the Naive strategy maintains some edge over the Sophisticated one even with as high error rate as .3.

Pre-pruning is not as aggressive as post pruning, which explains a substantial part of the differences observed in the performance of the *Rank* algorithm and the other two learning strategies. From these experiments it is hard to discern to what amount can we attribute these differences to the incomparable search procedures. However, in some trials the backtracking search clearly benefited and in others hampered the performance of decision tree learning.

5 Related research

In a subsequent study Schaffer [21] generalized the main observations of the overfitting avoidance bias into the conservation law of generalization performance, which essentially observes that there cannot be a universally superior bias. More or less the same has been expressed in Wolpert's [24] "no free lunch" theorems. The relevance of these results for practical inductive learning can be questioned, since experience has shown that many of the learning domains encountered in practice have an extremely simple representation [11] and, hence, the bias of heavy pruning would suit practical learning tasks better than other biases.

Holder [10] has shown that the *intermediate decision trees*—subtrees pruned in a breadth-first order from the full trees grown by C4.5—perform better than the full and pruned trees in a large corpus of UCI data sets. Intermediate decision trees correspond to pre-prunings of the full tree. Holder's experiments also support our earlier observation: pre-pruning, if given a slight advantage by careful parameter adjusting, may be a competitive alternative to post pruning of decision trees on real-world domains.

Domingos [4] has considered the correctness of different interpretations of Occam's Razor. He compiles a substantial amount of analytical and empirical evidence against interpreting that the Law of Parsimony would somewhat qualify or justify pruning as an inherently beneficial technique in concept learning. No evidence supports assuming that lower syntactic complexity of a concept description would somehow transform into lower generalization error.

6 Conclusions

In this study we wanted to set into perspective of Schaffer's [20] results the bias induced by pre-pruning of decision trees and to compare the bias of the rank-minimization to that of the standard top-down induction of decision trees. The performance of pre-pruning on the scale of the complexity of concept representation settles down in between the Naive and the Sophisticated strategies. The bias resulting from rank-minimization was visible in some trials, but on the average level we cannot conclude much about it on the basis of these experiments.

The main findings of our experiments are similar to those that were reported by Schaffer [20]: Sophisticated strategy is the better choice in domains with a simple description, while the Naive one is the better choice otherwise. The differences that were observed in learning random Boolean concepts when classification noise prevails are interesting and worth attention.

In future work one could experiment with forcing a maximum rank bound for the decision trees—in the spirit of learning decision trees with stringent syntactic restrictions [11]—and observing such restriction's effects on the prediction accuracy of the resulting tree. Further experimentation with the effects of different noise generating schemes and other tests performed by Schaffer should be carried out in order to complete the study initiated in this paper.

References

1. Angluin, D., Laird, P.: Learning from noisy examples. Mach. Learn. **2** (1988) 343–370
2. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth, Pacific Grove, CA (1984)
3. Domingos, P.: A process-oriented heuristic for model selection. In: Shavlik, J. (ed.): Machine Learning: Proceedings of the Fifteenth International Conference. Morgan Kaufmann, San Francisco, CA (1998) 127–135
4. Domingos, P.: Occam's two razors: the sharp and the blunt. In: Agrawal, R., Stolorz, P., Piatetsky-Shapiro, G. (eds.): Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, CA (1998) 37–43

5. Domingos, P.: Process-oriented estimation of generalization error. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, CA (to appear)
6. Ehrenfeucht A., Haussler, D.: Learning decision trees from random examples. Inf. Comput. **82** (1989) 231–246
7. Elomaa, T.: Tools and techniques for decision tree learning. Report A-1996-2, Department of Computer Science, University of Helsinki (1996)
8. Elomaa, T., Kivinen, J.: Learning decision trees from noisy examples, Report A-1991-3, Department of Computer Science, University of Helsinki (1991)
9. Hancock, T., Jiang, T., Li, M., Tromp, J.: Lower bounds on learning decision lists and trees. Inf. Comput. **126** (1996) 114–122
10. Holder, L. B.: Intermediate decision trees. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, CA (1995) 1056–1061
11. Holte, R. C.: Very simple classification rules perform well on most commonly used data sets. Mach. Learn. **11** (1993) 63–90
12. Murthy S. K., Salzberg, S.: Lookahead and pathology in decision tree induction. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, CA (1995) 1025–1031
13. Oates, T., Jensen, D.: The effects of training set size on decision tree complexity. In: Fisher, D. H. (ed.): Machine Learning: Proceedings of the Fourteenth International Conference, Morgan Kaufmann, San Francisco, CA (1997) 254–261
14. Quinlan, J. R.: Learning efficient classification procedures and their application to chess end games. In: Michalski, R., Carbonell, J., Mitchell, T. (eds.): Machine Learning: An Artificial Intelligence Approach. Tioga, Palo Alto, CA (1983) 391–411
15. Quinlan, J. R.: Induction of decision trees. Mach. Learn. **1** (1986) 81–106
16. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1993)
17. Quinlan, J. R.: Improved use of continuous attributes in C4.5. J. Artif. Intell. Res. **4** (1996) 77–90
18. Rao, R. B., Gordon, D. F., Spears, W. M.: For every generalization action, is there really an equal and opposite reaction? Analysis of the conservation law for generalization performance. In: Prieditis, A., Russell, S. (eds.): Machine Learning: Proceedings of the Twelfth International Conference. Morgan Kaufmann, San Francisco, CA (1995) 471–479
19. Sakakibara, Y.: Noise-tolerant Occam algorithms and their applications to learning decision trees. Mach. Learn. **11** (1993) 37–62
20. Schaffer, C.: Overfitting avoidance as bias. Mach. Learn. **10** (1993) 153–178
21. Schaffer, C.: A conservation law for generalization performance. In: Cohen, W. W., Hirsh, H. (eds.): Machine Learning: Proceedings of the Eleventh International Conference. Morgan Kaufmann, San Francisco, CA (1994) 259–265
22. Valiant, L. G.: A theory of the learnable. Commun. ACM **27** (1984) 1134–1142
23. Wang, C., Venkatesh, S. S., Judd, J. S.: Optimal stopping and effective machine complexity in learning. In: Cowan, J. D., Tesauro, G., Alspector, J. (eds.): Advances in Neural Information Processing Systems, Vol. 6. Morgan Kaufmann, San Francisco, CA (1994) 303–310
24. Wolpert, D. H.: The lack of a priori distinctions between learning algorithms. Neural Comput. **8** (1996) 1341–1390