Learning Decision Lists Using Homogeneous Rules

Richard Segal and **Oren Etzioni***
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{segal, etzioni}@cs.washington.edu

Appears in AAAI-94

Abstract

A decision list is an ordered list of conjunctive rules (Rivest 1987). Inductive algorithms such as AQ and CN2 learn decision lists incrementally, one rule at a time. Such algorithms face the *rule overlap problem* — the classification accuracy of the decision list depends on the overlap between the learned rules. Thus, even though the rules are learned in isolation, they can only be evaluated in concert. Existing algorithms solve this problem by adopting a greedy, iterative structure. Once a rule is learned, the training examples that match the rule are removed from the training set. We propose a novel solution to the problem: composing decision lists from *homogeneous* rules, rules whose classification accuracy does not change with their position in the decision list. We prove that the problem of finding a maximally accurate decision list can be reduced to the problem of finding maximally accurate homogeneous rules. We report on the performance of our algorithm on data sets from the UCI repository and on the MONK's problems.

Introduction

A decision list is an ordered list of conjunctive rules (Rivest 1987). A decision list classifies examples by assigning to each example the class associated with the first conjunctive rule that matches the example. The decision list induction problem is to identify, from a set of training examples, the decision list that will most accurately classify future examples. A learning algorithm requires some means for predicting how a decision list will perform on future examples. One solution is to use a heuristic scoring function that estimates the accuracy of the list on future examples based on its accuracy on training examples.[1] The overall induction problem can be decomposed into choosing an appropriate scoring function and finding a decision list that maximizes it.

*This research was funded in part by Office of Naval Research grant 92-J-1946 and by National Science Foundation grants IRI-9211045 and IRI-9357772. Richard Segal is supported, in part, by a GTE fellowship.

[1]To avoid overfitting, additional factors are often included such as the size of the list and the number of training examples covered.

A simple algorithm for finding a maximal decision list is to exhaustively search the space of decision lists and output the best one found. This algorithm is impractical because the number of decision lists is doubly-exponential in the number of attributes. Many existing algorithms (e.g., (Michalski 1969; Clark and Niblett 1989; Rivest 1987; Pagallo and Haussler 1990)) learn decision lists incrementally by searching the space of conjunctive rules for "good" rules and then combining the rules to form a decision list.

Such algorithms face the problem of *rule overlap* — the accuracy of a decision list is *not* a straightforward function of the accuracy of its constituent rules. To illustrate this point, consider the two rules r_1 and r_2, each having 80% accuracy and 50% coverage on the training examples. The rules may not overlap at all, which yields a two rule decision list with 80% accuracy and 100% coverage. However, the rules may have a 40% overlap in which case the accuracy of the decision list (r_1, r_2) could go down to 67% with a coverage of 60%. In general, any algorithm that forms a classifier by combining rules learned separately has to overcome the rule overlap problem.

Algorithms such as AQ and CN2 address the overlap problem by adopting an iterative structure. As each rule is learned, it is inserted into the decision list, and the examples covered by the rule are removed from the training set. The algorithm learns the next rule based on the reduced training set. The process is repeated until the training set is exhausted. The overlap problem is addressed by learning each successive rule from a training set where examples that match previously learned rules are filtered out. Note that this iterative approach is *greedy* — once the algorithm learns a rule, it is committed to keeping that rule in the decision list. All subsequent learning is based on this commitment.

While the greedy approach has proven to be effective in practice, it has several problems. First, as pointed out by Clark and Niblett (1989), the interpretation of each rule is dependent on the rules that precede it. This makes decision lists difficult to comprehend because the learned rules cannot be considered in iso-

lation. Second, on each iteration, fewer training examples are available for the learning algorithm, which hinders the algorithm's ability to learn. This is particularly important in situations where training data is scarce. Finally, poor rule choices at the beginning of the list can significantly reduce the accuracy of the decision list learned.

Nevertheless, Rivest showed that a greedy, iterative algorithm can provably PAC learn the concept class *k-DL*, decision lists composed of rules of length at most k (Rivest 1987). However, Rivest's PAC guarantee presupposes there exist 100% accurate rules of length at most k that cover the training examples. This strong assumption neatly sidesteps the overlap problem because the accuracy of a 100% accurate rule remains unchanged regardless of the rules that precede it in the decision list. However, the assumption is often violated in practice. A full complement of 100% accurate rules of length at most k cannot be found when there is noise in the training data, when the concept to be learned is not in *k-DL* (relative to the algorithm's attribute language), or when the concept is probabilistic.

Our main contribution is a solution to the overlap problem that is both theoretically justified and practical. We borrow the notion of *homogeneity* from the philosophical literature (Salmon 1984) to solve the overlap problem in learning decision lists. Informally, a homogeneous rule is one whose accuracy does not change with its position in the decision list.

Formally, let E denote the universe of examples. Let T denote the set of tests within a domain and G the set of goal classes. Let DL denote the set of all decision lists. Let $c(e)$ denote the classification of example e, and $C(e,d)$ denote the classification that decision list d assigns to the example e. We write a rule as $A \rightarrow g$, where $A \subset T$ and $g \in G$. When an example e passes all the tests in A, we say $e \in A$. Let P be a probability distribution over examples. We define the accuracy of a decision list d with respect to P as follows:

$$\mathcal{A}(d) = \sum_{e \in E | c(e)=C(e,d)} P(e)$$

We define the accuracy of a rule to be its accuracy on the examples that it covers:

$$a(A \rightarrow g) = \sum_{e \in A | c(e)=g} P(e) \Big/ \sum_{e \in A} P(e)$$

A *homogeneous rule* is a rule for which all specializations of the rule have the same accuracy as the rule itself. Formally, a homogeneous rule is a rule $A \rightarrow g$ such that, for all $B \subset T$, the following holds:

$$a(A \wedge B \rightarrow g) = a(A \rightarrow g)$$

All 100% accurate rules are homogeneous, but homogeneous rules need not be 100% accurate. Thus, homogeneity can be viewed as a generalization of Rivest's solution to the overlap problem. This generalization is valuable in situations where concise 100% accurate rules do not exist.

Our algorithm for learning decision lists, BRUTEDL, searches the space of conjunctive rules for maximally accurate homogeneous rules and composes the rules found into a decision list. The remainder of the paper is organized as follows. The next section introduces the theory underlying BRUTEDL. The following section explains the approximations to the theory we use to make BRUTEDL practical. We then describe BRUTEDL's algorithm and discuss how we make our implementation efficient. Finally, we present empirical results that validate our approach and compare BRUTEDL with related algorithms.

Homogeneous Decision Lists

This section describes the theory underlying BRUTEDL. Our goal is to demonstrate that the problem of finding a maximally accurate decision list can be reduced to the problem of finding maximally accurate homogeneous rules.

A homogeneous decision list is a decision list composed exclusively of homogeneous rules. We use *HDL* to refer to the set of all homogeneous decision lists. BRUTEDL is restricted to learning homogeneous decision lists. Does this restriction mean that in some cases BRUTEDL will be forced to learn an inferior decision list? In other words, are there cases where the best homogeneous decision list is less accurate than the best decision list? The answer is no. We say that two decision lists d and d' are *logically equivalent* if they classify all examples identically. That is, $\forall\, e \in E,\ C(e,d) = C(e,d')$.

Theorem 1 *For every decision list, there exists a homogeneous decision list that is logically equivalent.*

Proof sketch:

Let d be a nonhomogeneous decision list and $r = A \rightarrow g$ be a nonhomogeneous rule in d. We can replace r with a set of equivalent homogeneous rules. By performing this replacement for all nonhomogeneous rules in d, a homogeneous decision list logically equivalent to d can be found. Let t be any test not in A. The rule r can be replaced with $A \wedge t \rightarrow g$ and $A \wedge \neg t \rightarrow g$ without changing how d classifies examples. If these rules are homogeneous, we are done. If not, we can repeat the procedure and replace the two rules with four rules without logically changing d. This procedure can be repeated until a set of homogeneous rules is found. A set of homogeneous rules is guaranteed to be found because there is a finite number of tests.□

Our solution to the overlap problem combines the notion of homogeneity with the intuition that the best rule to classify any given example is the most accurate rule that covers the example. We define a *maximal cover* as a set of rules containing, for each example, the most accurate homogeneous rule that covers it. Formally, let $hr(e)$ be the set of homogeneous rules that

match e. A maximal cover $M(E)$ of a universe of examples E is any set of homogeneous rules for which the following holds:

$$\forall\, e \in E,\ \exists\, r \in M(E) \text{ such that } a(r) = \max_{r' \in hr(e)} a(r')$$

We now show that the problem of finding the maximally accurate decision list can be reduced to the problem of finding a maximal cover for E.

Theorem 2 *Any homogeneous decision list d whose rules form a maximum cover of E and is sorted by decreasing accuracy will have:*

$$\mathcal{A}(d) = \max_{d' \in DL} \mathcal{A}(d')$$

Proof:

First we prove that d must be a maximally accurate homogeneous decision list. Assume that $\mathcal{A}(d) \neq \max_{d' \in HDL} \mathcal{A}(d')$. There must exist a homogeneous decision list f such that $\mathcal{A}(f) > \mathcal{A}(d)$. Let e denote an example that is classified by a rule f_i in f and d_j in d such that $a(f_i) > a(d_j)$. Since $\mathcal{A}(f) > \mathcal{A}(d)$, such an example must exist. The rules of d form a maximum cover; therefore, there exists a d_k such that $a(d_k) = \max_{r \in hr(e)} a(r)$. Furthermore, we have $a(d_k) \geq a(f_i)$ because $f_i \in hr(e)$. We have $k < j$ because $a(d_k) \geq a(f_i) > a(d_j)$ and d is sorted by decreasing accuracy. But if $k < j$, e should have been classified by d_k rather than d_j. This contradiction establishes that $\mathcal{A}(d) = \max_{d' \in HDL} \mathcal{A}(d')$. Let g be a decision list such that $\mathcal{A}(g) = \max_{d' \in DL} \mathcal{A}(d')$. Assume $\mathcal{A}(g) > \mathcal{A}(d)$. By Theorem 1, there exists an $h \in HDL$ that is logically equivalent to g. We have $\mathcal{A}(h) = \mathcal{A}(g)$ because h and g are logically equivalent. But since h is homogeneous, we have $\mathcal{A}(d) \geq \mathcal{A}(h) = \mathcal{A}(g)$ which contradicts the assumption. Thus, $\mathcal{A}(d) = \max_{d' \in DL} \mathcal{A}(d')$.□

Implications for BRUTEDL

We now consider the implications of Theorem 2 for BRUTEDL. If BRUTEDL had access to the probability distribution P and the set of homogeneous rules, it would be straightforward to build an algorithm based on Theorem 2. In practice, BRUTEDL is only given a set of training data from which it must approximate P and determine which rules are homogeneous.

BRUTEDL uses *LaplaceAccuracy* as an approximation of the actual accuracy of a rule (Niblett 1987). Let r be a rule that classifies r_p training examples correctly out of the r_n training examples it matches. Let $|G|$ denote the number of goal classes. The LaplaceAccuracy of r is calculated as follows:

$$LaplaceAccuracy(r) = \frac{r_p + 1}{r_n + |G|}$$

Once an estimate for the accuracy of individual rules has been defined, it is possible to check whether a rule is homogeneous. The accuracy of a homogeneous rule should not change when additional conjuncts are added. Therefore, homogeneity can be checked by comparing the LaplaceAccuracy of a rule with the LaplaceAccuracy of all the rule's specializations. Since LaplaceAccuracy is an approximation to the actual accuracy of a rule, a rule is considered homogeneous if all specializations have *roughly* the same LaplaceAccuracy. We check for statistically significant differences in LaplaceAccuracy using a χ^2 test.

Although not required by Theorem 2, it is desirable that the rules learned by BRUTEDL do not contain irrelevant conjuncts. An irrelevant conjunct is any conjunct whose presence does not affect the accuracy of a rule. We will call any rule with only relevant conjuncts *minimal*. Restricting BRUTEDL to minimal rules does not affect the class of concepts it can learn because, for every nonminimal homogeneous rule, there is a minimal rule with identical accuracy and greater coverage that is formed using some subset of the original rule's conjuncts. We check whether a conjunct is relevant by checking if the accuracy of the rule changes when the conjunct is removed. Again, we use a χ^2 test to ensure that any differences in accuracy that are detected are significant.

Algorithm

The previous sections developed the basic framework behind BRUTEDL. We now describe how this framework is implemented. The core of BRUTEDL is a depth-first search to find, for each example, the best conjunctive rule that covers it. Rules that are neither homogeneous nor minimal are filtered out. Once a maximal cover has been found, the cover is sorted, and a default rule is appended. BRUTEDL limits its search to a fixed depth bound when it is too costly to search the entire space. A pseudo-code description of BRUTEDL appears in Table 1.

The search performed by BRUTEDL is systematic, it visits each rule exactly once. In a naive search of the space of conjunctive rules, the identical rules $A \wedge B \rightarrow g$ and $B \wedge A \rightarrow g$ would be visited separately: once while searching the children of $A \rightarrow g$ and once while searching the children of $B \rightarrow g$. BRUTEDL achieves systematicity by imposing a canonical order on the tests within a rule. BRUTEDL assigns a numeric rank to each test and only considers rules whose tests appear in order of increasing rank.

Homogeneity is checked by doing a systematic search of all specializations of a rule. If a specialization is found with a difference in accuracy that is considered statistically significant, the homogeneity check fails. If no such specialization is found, the rule is deemed homogeneous. BRUTEDL limits the cost of homogeneity checks by reducing their frequency. It is only necessary to check the homogeneity of a rule that is minimal and is the best rule seen thus far for some example. If a rule does not meet these two criteria, it cannot be

```
BruteDL()
  DecisionList := MakeEmptyDL()
  BruteSearch(MakeEmptyRule());
  Sort(DecisionList);
  AddDefaultRule(DecisionList);
END

BruteSearch(rule)
  IF Length(rule) >= MaxLength THEN EXIT;
  StartTest := FollowingTest(LastTest(rule));
  FOR test := StartTest to LastTest DO
    newrule := AddConjunct(rule, test);
    IF BestForSomeExample(newrule) AND
       IsMinimal(newrule) AND
       Homogeneous(newrule,newrule)
      THEN Insert(newrule,DecisionList);
    IF NOT PruneRule(newrule)
      THEN BruteSearch(newrule);
  END
END

Homogeneous(ParentRule, rule):boolean
  IF Length(rule) >= MaxLength THEN RETURN(True);
  IF rule = ParentRule
    THEN StartTest = 1;
    ELSE StartTest = FollowingTest(LastTest(rule));
  FOR test := StartTest to LastTest DO
    newrule := AddConjunct(rule,test);
    IF NOT SimilarAccuracy(ParentRule,newrule) THEN
      RETURN(False);
    ELSE IF NOT Homogeneous(ParentRule,newrule) THEN
      RETURN(False);
  END
  RETURN(True);
END

sMinimal(rule):boolean
  FOR test in Conjuncts(rule) DO
    ParentRule := DeleteConjunct(rule, test)
    IF SimilarAccuracy(ParentRule, rule)
      THEN RETURN(False);
  END
  RETURN(True);
END
```

Table Pseudo-code for BruteDL

part of the final decision list. Using this filter, a homogeneity check is required for only a small fraction of the rules searched. Furthermore, the homogeneity check for a nonhomogeneous rule is often inexpensive because the search is terminated once a specialization with a significant difference in accuracy is found.

Once a maximum cover has been found, BRUTEDL uses it to build a decision list. The final decision list is formed by sorting the maximum cover and appending a default rule. A default rule is necessary because the rules found by BRUTEDL, although required to cover the training examples, might not cover all the test examples. BRUTEDL appends a default rule that predicts the most frequent class in the training data.

(1) If Accuracy(r) = 100% then Prune(r).

(2) If MatchedPositives(r) < MinPositives then Prune(r).

(3) If MatchedNegatives(A$\wedge\neg$c) < MinSimNegatives $\wedge$ MatchedPositives(A$\wedge\neg$c) < MinSimPositives then Prune(A$\wedge$c).

Table 2: Pruning axioms used by BRUTEDL. Prune(r) indicates the children of r should not be searched. The axioms are sound because the rules they prune cannot be part of the final decision list.

Efficiency

For BRUTEDL to be a practical algorithm, it is important that it be as efficient as possible. The efficiency of BRUTEDL is determined by two factors: the efficiency of processing each rule and the number of rules processed. We address the first element of BRUTEDL's efficiency by carefully implementing BRUTEDL in C. BRUTEDL can process approximately 100,000 rules per second when running on a SPARC-10 processor with a data set of 500 examples. BRUTEDL's running time grows linearly with the number of examples. Significant improvements in program efficiency are not expected because BRUTEDL is currently within an order of magnitude of the machine's clock rate. However, further improvements in rule processing speed will occur as faster machines become available.

The second element of BRUTEDL's efficiency is the number of rules it processes. BRUTEDL can reduce the number of rules it has to process by pruning away rules *guaranteed* not to be part of the final decision list. BRUTEDL uses the axioms in Table 2 to determine the portions of the search space it can ignore. BRUTEDL's pruning axioms significantly reduce the number of rules it has to process. For the test domains presented later, the pruning axioms reduced the search space by as much as a factor of 1,000. The remainder of this section describes BRUTEDL's pruning axioms in detail.

The first axiom prunes descendants of 100% accurate rules because they cannot be minimal. The second axiom prunes all specializations of a rule that do not cover a minimum number of positive examples. During its search, BRUTEDL keeps track of the worst rule that is the best for some example. For a rule to appear in the final output, it is necessary for it to be better than this rule. It is possible to show that for LaplaceAccuracy and many other functions, there is a minimum number of positive examples a rule must cover for it to achieve a particular score. By setting MinPositives to the number of positives required to improve upon the worst rule, we can prune rules that are guaranteed not to appear in the final decision list.

The third pruning axiom avoids exploring portions of the search space that are guaranteed not to be minimal. If a rule contains a conjunct that does little to affect the rule's accuracy, then any specialization of that rule

will not be minimal. Consider the rule $r = A \wedge c \rightarrow g$ where the conjunct c has little influence on the accuracy of the rule. We will determine the conditions for which any specialization of r is guaranteed to be nonminimal. Let $s = A \wedge c \wedge B \rightarrow g$ be any specialization of r. For s to be a minimal rule, it is necessary that its accuracy be significantly different from the accuracy of $p = A \wedge B \rightarrow g$. For s to be minimal, it must therefore be more accurate or less accurate than p. The maximum possible value of χ^2 for a specialization of p that is more accurate than p is obtained by a rule that matches all the positives of p and none of its negatives. If p contains too few negative examples, then the maximal value of χ^2 will be lower than the threshold required to be judged significant. Thus, no specialization of p that is more accurate than p can be significant unless p contains a minimum number of negative examples. A similar argument demonstrates that no specialization of p that is less accurate than p can be significant unless p contains a minimum number of positive examples. The negative and positive examples in p that do not appear in r are those for which $A \wedge \neg c \wedge B$ holds. If $A \wedge \neg c \wedge B$ has too few positive examples and too few negative examples, then s cannot be minimal. Furthermore, if the set $A \wedge \neg c$ contains too few positive examples and too few negative examples, then $A \wedge \neg c \wedge B$ must also not contain enough examples. Therefore, by pruning specializations of a rule $A \wedge c \rightarrow g$ when $A \wedge \neg c$ does not have the required number of positive and negative examples, we remove from consideration only rules that are guaranteed not to be minimal.

Experimental Results

We ran BRUTEDL on several data sets from the UCI repository (Murphy 1994)[2] and on all the data sets from the MONK's competition (Thrun *et al.* 1991). The results are shown in Table 3. For comparison, the results for the IND (Buntine and Caruana 1991) implementation of C4 (Quinlan 1986) are also included. The results for the UCI data sets are averaged over 10 iterations. Each iteration randomly splits the available data into 70% for training and 30% for testing. The MONK's problems specify both the training set and test set to use for each problem.

BRUTEDL performed as well as C4 on many of the UCI data sets and better than C4 on the lymphography data set. However, BRUTEDL performed relatively poorly on the glass and voting data sets. BRUTEDL is a clear winner on the MONK #1 data set, and performed at least as well as C4 on the other two MONK's problems. The target concept in the MONK #1 data set is an XOR, which is known to be difficult for decision tree algorithms. In contrast, XOR is easy for BRUTEDL since it merely has to find a homogeneous rule corresponding to each disjunct.

All of the UCI data sets were too large for a complete search. In each of the data sets, a depth bound was used to restrict the search to consider rules only up to a certain length. Table 4 shows the execution times and depth bounds for each data set. BRUTEDL is fast, taking only a few CPU seconds on some data sets and no more than 17 CPU minutes on the slowest one.

[2]The breast cancer, lymphography, and primary tumor domains were obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.

	BRUTEDL		C4	
Domain	Acc.	σ	Acc.	σ
Breast cancer	68.7	4.3	69.8	3.2
Chess endgame	98.6	0.4	99.2	0.3
Glass	62.0	5.3	69.2	5.5
Hepatitis	80.6	7.9	80.0	7.9
Iris	93.1	4.5	94.2	2.7
Lymphography	82.0	3.4	69.6	3.4
Mushroom	100.0	0.1	100.0	0.0
Primary tumor	39.9	3.0	39.1	4.9
Voting records	93.0	3.2	94.6	1.5
MONK #1	100.0	N/A	80.6	N/A
MONK #2	68.1	N/A	64.8	N/A
MONK #3	97.2	N/A	97.2	N/A

Table 3: The results of BRUTEDL and C4 on several data sets. All results except for the MONK data sets are averaged over 10 iterations. The MONK data sets come with a single training and test set.

Domain	CPU Time min:sec	Search depth
Breast cancer	0:31	4
Chess endgame	16:34	5
Glass	6:29	3
Hepatitis	0:53	3
Iris	0:37	5
Lymphography	1:13	5
Mushroom	2:09	3
Primary tumor	0:08	4
Voting records	0:04	5
MONK #1	0:01	N
MONK #2	0:04	N
MONK #3	0:01	N

Table 4: Running times and search depths for BRUTEDL. CPU time is for a SPARC-10 workstation. A search depth of N indicates a complete search.

Critique

Ideally, BRUTEDL's massive search would result in significant improvements when compared with a greedy algorithm such as C4. Our experiments do not demonstrate this improvement for several reasons. First, on some data sets (e.g., mushroom) we observe a ceiling effect — C4 is performing about as well as possible, given the data set and attribute language. Second, in some cases, BRUTEDL overlooks homogeneous rules.

BRUTEDL discards a rule as nonhomogeneous when it has a specialization that differs significantly in accuracy from the rule itself. BRUTEDL performs a χ^2 test at $p = .005$ on each specialization of the rule to determine if its accuracy is significantly different from that of the rule. However, it is not the case that the probability that BRUTEDL incorrectly judges a rule to be nonhomogeneous is .005. Although the probability that a single error is .005, the probability that at least one of N judgments is in error is $1 - .995^N$. Thus, the more specializations a rule has, the more likely it is to be judged incorrectly as nonhomogeneous.

On both the voting and breast cancer data sets, BRUTEDL incorrectly judged several key rules to be nonhomogeneous. We can reduce the likelihood BRUTEDL will incorrectly judge a rule as nonhomogeneous by using a lower p value for the χ^2 tests. By using $p = .00001$, BRUTEDL improves its performance to 94.4% accuracy on the voting data and to 72.2% accuracy on the breast cancer data. However, simply increasing the confidence in individual χ^2 tests can cause BRUTEDL to treat a nonhomogeneous rule as homogeneous. For instance, accuracy on the primary tumor data set decreases to 34.8% when we change the confidence level to $p = .00001$. It is clear that a more stable method of checking homogeneity is needed.

Finally, BRUTEDL's performance is limited in domains where it is not able to search to sufficient depth to find accurate rules. For instance, the rules found by BRUTEDL at depth 3 in the glass domain are not as accurate as those found by C4 at depths 5 and 6. Heuristic search techniques (e.g., beam search) can be used when a pure depth-bounded search to the desired depth is too costly. The basic ideas behind BRUTEDL apply equally well to heuristic search.

Related Work

BRUTEDL builds on our previous work on BRUTE (Riddle *et al.* 1994). BRUTE uses a depth-bounded search of the space of conjunctive rules to find accurate predictive rules. We tested BRUTE on two data sets from a Boeing manufacturing domain. The first data set has 1,075 examples with 48 attributes, and the second has 519 examples with 1,652 attributes. In the first data set, the predictive rules found by BRUTE were 20% more accurate on average than those found by C4. In the second data set, the predictive rules found by BRUTE were 44% accurate on average, while C4 was unable to find any rules. The results demonstrate the effectiveness of depth-bounded search on a complex real-world domain. BRUTE's running time on the two data sets was less than 3 CPU minutes on a SPARC-10 workstation.[3]

nSeveral other systems have used depth-bounded search. ITRULE (Smyth and Goodman 1991), like BRUTE, uses depth-bounded search to find accurate predictive rules. Schlimmer (1993) uses depth-bounded search to find determinations. However, none of these systems attempt to build a classifier from the rules they find.

Rivest (1987) describes an algorithm for PAC learning the concept class *k-DL*, decision lists composed of rules of length at most k. Rivest's *k-DL* algorithm conducts a depth-bounded search of the space of conjunctive rules to find 100% accurate rules. This depth-bounded search is repeated n times where n is the number of rules in the learned decision list. We can improve upon the *k-DL* algorithm by restricting BRUTEDL to consider only 100% accurate rules. The homogeneity check can be dropped because 100% accurate rules are necessarily homogeneous. This restricted version of BRUTEDL will PAC learn *k-DL* using a *single* depth-bounded search of the space of conjunctive rules. The time complexity of the restricted BRUTEDL is asymptotically faster than that of the *k-DL* algorithm by a factor of n. Furthermore, the unrestricted BRUTEDL is more general. It can be used on noisy domains, probabilistic concepts, and concepts not in *k-DL*.

Rivest's algorithm is very similar to the AQ line of inductive algorithms (e.g., (Michalski 1969; Clark and Niblett 1989)). These algorithms share Rivest's iterative structure but use a beam search to find the best rule according to a scoring function. The OPUS system (Webb 1993) extends CN2 to use depth-bounded search but retains the same iterative structure. As a result, poor rule choices at the beginning of the list can significantly reduce the accuracy of the decision list learned. Furthermore, the greedy structure introduces dependencies among the decision list's rules that can make the decision list difficult to interpret. BRUTEDL's solution to the overlap problem avoids both these pitfalls by learning each rule in the decision list independently.

The PVM system (Weiss *et al.* 1990) does a massive search of the space of classifiers. PVM's search is not exhaustive because it uses several heuristics to reduce the search space.[4] Even with heuristics, the doubly-exponential search space searched by PVM limits it to considering classifiers that are significantly smaller than those considered by BRUTEDL. Finally, Murphy and Pazzani (1994) used a depth-bounded search of the space of decision trees to analyze the relationship between the smallest decision tree and classification accuracy. A massively parallel Maspar computer and small domains were used to make a limited search of this doubly-exponential space possible. Our theory of homogeneity and sound pruning axioms significantly reduce the cost of depth-bounded search and make it practical in many domains.

Many of BRUTEDL's features help to improve the

[3] We previously reported BRUTE's running time as 33 CPU minutes. We have since added additional pruning axioms that significantly improve BRUTE's efficiency.

[4] Unlike BRUTEDL's pruning axioms, PVM's heuristics are not sound and can cause it to overlook accurate classifiers

human readability of its decision lists. As pointed out by Clark and Niblett (1989), the readability of a decision list suffers because the interpretation of each rule is dependent on the rules that precede it. BRUTEDL avoids this problem by finding only homogeneous decision lists. Homogeneous decision lists are easier to understand because the interpretation of each rule is not dependent on its position. Furthermore, BRUTEDL attempts to include all relevant conjuncts within each rule while leaving out any irrelevant conjuncts.

Another unique aspect of BRUTEDL is that it does not use a postpruning phase to avoid overfitting. Postpruning does not make sense for any algorithm that is trying to maximize a scoring function because it would prune the maximal classifier found into some classifier that would be nonmaximal according to its scoring function. Instead, BRUTEDL uses its heuristic scoring function to avoid overfitting by assigning a low score to any rule that covers too few training examples. Overfitting is also avoided by requiring that every conjunct in a rule be relevant.

Conclusion

This paper introduced BRUTEDL, a novel algorithm for learning decision lists. Unlike algorithms such as AQ or CN2, BRUTEDL conducts a single search for accurate homogeneous rules, which contain no redundant conjuncts, and builds a decision list from the rules it finds. We show that, in the limit, the problem of learning maximally accurate decision lists can be reduced to the problem of learning maximally accurate homogeneous rules. BRUTEDL introduces a number of approximations to this theory but, as our empirical results demonstrate, BRUTEDL is effective in practice. BRUTEDL outperforms C4 in several cases and runs in less than a minute on most benchmark data sets. In future work we plan to compare BRUTEDL with CN2 and to demonstrate that decision lists, based on homogeneous rules, are easier to comprehend than standard decision lists.

Acknowledgments

We are grateful to Wray Buntine for distributing the IND package. IND's reimplementation of C4 greatly facilitated our research. Thanks are due to Ruth Etzioni for her expert advice on statistical testing, Patricia Riddle for many helpful discussions, Dan Weld for commenting on an earlier version of this paper, and Omid Madani for help testing BRUTEDL.

References

W. Buntine and R. Caruana. Introduction to IND and recursive partitioning. NASA Ames Research Center, Mail Stop 269-2 Moffet Field, CA 94035, 1991.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, March 1989.

R. S. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the Fifth International Symposium on Information Processing*, pages 125–128, Bled, Yugoslavia, 1969.

Patrick M. Murphy and Michael J. Pazzani. Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction. Submitted to *Artificial Intelligence Research*, 1994.

Patrick M. Murphy. UCI repository of machine learning databases. [Machine-readable data repository]. Irvine, CA. University of California, Department of Information and Computer Science., 1994.

T. Niblett. Constructing decision trees in noisy domains. In *Progress in Machine Learning (Proceedings of the 2nd European Working Session on Learning)*, pages 67–78, Wilmslow, UK, 1987.

G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–100, March 1990.

J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1986.

Patricia Riddle, Richard Segal, and Oren Etzioni. Representation design and brute-force induction in a Boeing manufacturing domain. *Applied Artificial Intelligence*, 8:125–147, 1994. Available via anonymous FTP from `/pub/ai` at cs.washington.edu.

R. Rivest. Learning decision trees. *Machine Learning*, 2:229–246, 1987.

Wesley C. Salmon. *Scientific Explanation and the Causal Structure of the World.* Princeton University Press, Princeton, NJ, 1984.

J. C. Schlimmer. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA, June 1993.

P. Smyth and R. M. Goodman. Rule induction using information theory. In *Knowledge Discovery in Databases*, pages 159–176. MIT Press, Cambridge, MA, 1991.

S.B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik J. Cheng, K. De Jong, S. Dzeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang. The MONK's problems - A performance comparison of different learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University, 1991.

Geoffrey I. Webb. Systematic search for categorical attribute-value data-driven machine learning. In N. Foo and C. Rowles, editors, *AI '93*. World Scientific, Singapore, 1993.

S. M. Weiss, R. S. Galen, and P. V. Tadepalli. Maximizing the predictive value of production rules. *Artificial Intelligence*, 45:47–71, September 1990.