Reduced Complexity Rule Induction

Sholom M. Weiss and Nitin Indurkhya
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903

Abstract

We present an architecture for rule induction that emphasizes compact, reduced-complexity rules. A new heuristic technique for finding a covering rule set of sample data is described. This technique refines a set of production rules by iteratively replacing a component of a rule with its single best replacement. A method for rule induction has been developed that combines this covering and refinement scheme with other techniques known to help reduce the complexity of rule sets, such as weakest-link pruning, resampling, and the judicious use of linear discriminants. Published results on several real-world datasets are reviewed where decision trees have performed relatively poorly. It is shown that far simpler decision rules can be found with predictive performance that exceeds those previously reported for various learning models, including neural nets and decision trees.

1 Introduction

Although many different models of induction, such as decision trees, neural nets, and linear discriminants, have been used for classification, their common goal is accuracy of prediction. A central issue in the design of most classifiers is the tradeoff of goodness of fit versus model complexity. While an increasingly complex classifier can usually be made to cover training samples better, its accuracy of predictions for new cases may be inferior to a simpler, less complex classifier. For example, a fully expanded decision tree may cover training samples completely, but a smaller tree, with a larger apparent error on the training cases, may be more accurate in its predictions for new cases.

Classifier complexity and prediction accuracy are highly related. Learning from sample data can be described as the estimation of parameters for a model. Finding the appropriate complexity fit of a model is a determination of the number of parameters that can be accurately estimated from the samples. From this characterization of learning it follows that given two classifiers that cover the sample data equally well, the simpler one is usually preferred because fewer parameters are estimated and therefore, predictions are likely to be more accurate. The theoretical foundation of this form of complexity fit analysis is presented in [Wallace and Freeman, 1987]. The related concept of minimum description length as a measure of the best complexity fit is found in [Rissanen, 1978, Rissanen, 1987].

In practice, designers of learning systems have implicitly recognized these principles, and many techniques for model simplification can be characterized as finding relatively compact solutions with an appropriate complexity fit. Examples from decision trees are quite numerous including the use of heuristic tree splitting functions to reduce expected tree size [Breiman, Friedman, Olshen, and Stone, 1984, Quinlan, 1986], tree pruning [Breiman, Friedman, Olshen, and Stone, 1984, Quinlan, 1987a], and the one-standard-error heuristic for selecting among pruned trees [Breiman, Friedman, Olshen, and Stone, 1984]. For parametric statistical linear discriminants, heuristic variable selection methods have been developed to reduce the number of features in the discriminant [James, 1985]. For rule induction, simplification has been achieved by pruning the rules implicit in decision trees [Quinlan, 1987a]. For single hidden layer back-propagation neural nets, the number of hidden units can be used as measure of complexity fit, and the apparent and true error rates will follow the classical statistical pattern [Weiss and Kapouleas, 1989].

Thus, there are strong theoretical reasons for developing learning methods that cover samples in the most efficient and compact manner. In this paper, we discuss a new approach to generating reduced complexity solutions for rule induction models. In a related work [Weiss, Galen, and Tadepalli, 1990], it was shown that short, disjunctive normal form expressions sometimes offer superior solutions. While that method empirically supports the reduced complexity approach, it is applicable only to problems with few attributes and classes. In this paper, we describe a new iterative rule method for inducing reduced complexity solutions in larger dimensions. We then describe a unified approach to finding the appropriate complexity fit among several competing rule sets.

2 Methods

2.1 The Rule-based Classification Model

We are given a set of sample cases, S, where each case is composed of observed features and the correct classification. The problem is to find the best rule set RS_{best} where the error rate on new cases, $Err_{true}(RS_{best})$, is minimum.

We examine solutions posed in disjunctive normal form (DNF), where each class is classified by a set of disjunctive productions (terms) . Each term is a conjunction of tests, p_i, where p_i is a proposition formed by evaluating the truth of a binary-valued feature or by thresholding any of the values a numerical feature assumes in the samples. One such model is the decision tree, where all the implicit productions are mutually exclusive. However, a general DNF model does not require mutual exclusivity of rules. With productions that are not mutually exclusive, rules for two classes can potentially be satisfied simultaneously. Such conflicts can be resolved by assigning a class (or rule) priority ordering, with the last class considered a default class. For example, the rule set in Figure 1 is a solution induced for the heart disease data discussed in Section 3.

CA>0.5 & CP>3.5 →	Class=2
THAL>6.5 →	Class=2
[TRUE] →	Class=1

Figure 1: Example Rule Set

While tree induction remains the most widely applied rule-based learning system, other learning techniques have been developed with some success for non-mutually exclusive DNF rule induction. These systems include C4 [Quinlan, 1987b, Quinlan, 1987a] and the CN2 [Clark and Niblett, 1989] and GREEDY3 [Pagallo and Haussler, 1989] variants of the AQ family of rule induction systems [Michalski, Mozetic, Hong, and Lavrac, 1986].

Although the aforementioned rule induction methods may appear quite different, only a few key variations emerge. These rule induction methods can be characterized in terms of their covering schemes and their rule refinement techniques. The covering rule-set is induced by (a) decision tree; or (b) a single best rule is found, the cases covered by that rule are removed from the training sample, and the next rule is induced until no cases remain. The set of rules can be refined by pruning or by applying some statistical test.

2.2 A Swap-1 Covering Procedure

Both the tree covering and the single-best-rule covering methods that have been mentioned, look ahead one attribute and try to specialize the tree or rule. To this end, a heuristic mathematical function is used, such as an entropy or gini function [Breiman, Friedman, Olshen, and Stone, 1984]. For the tree covering solutions, these heuristics tend to work well on many problems, and the combinatorics of finding an optimal solution make alternative search procedures impractical.

Like the tree induction methods, current single-best-rule methods expand only a single rule at a time and add tests one by one until a rule has 100% predictive value, i.e. makes no errors on the training cases.[1] Although any single rule is relatively short, these single-best-rule procedures never look back, only look ahead for a single test.

In this section, we present a procedure, called Swap-1 that constantly looks back to see whether any improvement can be made before adding a new test. The following steps are taken to form the single best rule: (a) Make the single best swap for any rule component including deleting the component; (b) If no swap is found, add the single best component to the rule. Figure 2 formally describes the Swap-1 procedure. As in [Weiss, Galen, and Tadepalli, 1990], "best" is evaluated as predictive value, i.e. percentage correct decisions by the rule. For equal predictive values, maximum case coverage is a secondary criterion. Swapping and component addition terminate when 100% predictive value is reached.

The process of generating the single best rule can be seen in Figure 3, where an example rule is generated in 7 steps. Swap-1 tries to maximize predictive value. The initial rule is randomly assigned p3, which gets swapped out in favor of the single best test, p6. Then in step 3, p1 is the single best component that can be add to the rule. However, in step 4, p6 is swapped out for p4, which is found by refining previously selected rule components. In the final step, we see that p3, which was swapped out in the first step, gets swapped in again. Thus, it can be seen that if a test is swapped out, it does not necessarily stay out, but can be added back later on if doing so improves the predictive accuracy of the current rule. The completed rule is selected as the single best rule, and the method proceeds as usual with the removal of the covered cases, and the re-application of the single-best-rule construction procedure to the remaining cases.

As a pure covering procedure, Swap-1 has been compared to other rule-based methods for covering randomly generated expressions from uniformly distributed attributes [Indurkhya and Weiss, 1990, Indurkhya and Weiss, 1991]. Its performance exceeds that of other rule induction methods and matches the performance of FRINGE [Pagallo, 1989], an iterative tree induction technique. When applied to real-world data with numerical attributes, the Swap-1 procedure can lead to fragmentation of the data by covering with too many short rules. There may be longer rules that are still 100% predictive, but cover more cases. Therefore, once a 100% predictive rule, R_i, is obtained, a longer rule R_j is induced by swapping on R_i for minimum errors (not predictive value) to obtain R_k and then

Input: S a set of training cases

Initialize R_1 := empty set, k := 1, and C_1 := S

repeat
  create a rule B with a randomly chosen attribute as its LHS
  while (B is not 100% predictive) do
    make the single best swap for any component of B,
      including deleting the component, using cases in C_k
    if no swap is found, add the single best component to B
  endwhile
  P_k := rule B that is now 100% predictive
  E_k := cases in C_k that satisfy the single-best-rule P_k
  $R_{k+1} := R_k \cup \{P_k\}$
  $C_{k+1} := C_k - \{E_k\}$
  k := k+1
until (C_k is empty)
find rule *r* in R_k that can be deleted without affecting
  performance on cases in S
while (*r* can be found)
  $R_{k+1} := R_k - \{r\}$
  k := k+1
endwhile
output R_k and halt

Figure 2: The Swap-1 Procedure

Step	Predictive Value	Rule
1	31%	p3
2	36%	p6
3	48%	p6 & p1
4	49%	p4 & p1
5	69%	p4 & p1 & p2
6	80%	p4 & p1 & p2 & p5
7	100%	p3 & p1 & p2 & p5

Figure 3: Example of Swapping Rule Components

[1]Some variations such as CN2, may stop earlier, based on a statistical significance test.

re-initializing R_1 in the Swap-1 procedure with R_k. R_i is then compared for coverage with the longer rule R_j.

Finding the optimal combination attributes and values for even a single fixed-size rule is a complex task. However, there are other optimization problems, such as the traveling salesman problem [Lin and Kernighan, 1973], where local swapping finds excellent solutions.

2.3 Finding the Right Complexity Rule Set

There have been some efforts to find the best rule set by devising measures of minimum description length which can give reasonable answers because complexity and predictive performance are highly related [Quinlan and Rivest, 1989]. However, the central objective remains to find a rule set that minimizes the true error rate, and this can be more directly measured by determining the true error rates of varying complexity rule sets.

Given a set of samples S, the objective is to select rule set RS_{best} from $\{RS_1,...RS_i,...RS_n\}$, a collection of varying complexity rule sets, such that RS_{best} will make the fewest errors on new cases T. In practice, the optimal solution can usually not be found because of incomplete samples and limitations on search time. Typically, there are insufficient cases to both train and accurately estimate the error rate of a rule set, $Err(RS_i)$. Moreover, it is not possible to search over all possible rule sets of complexity $Cx(RS_i)$, where Cx is some appropriate complexity fit measure, such as the number of components in the rule set.

Several thousand independent test cases are sufficient to give highly accurate estimates of the error rate of a classifier [Highleyman, 1962]. When fewer cases are available, resampling gives the most accurate estimates of the error rate. Cross-validation [Stone, 1974] is generally the procedure of choice [Breiman, Friedman, Olshen, and Stone, 1984], and 10-fold cross-validation[2] is usually quite accurate when the cases number in the hundreds. Because cross-validation techniques, such as 10-fold or leaving-one-out, average the estimates for many classifiers that are trained on approximately the same number of cases as the full sample, learning techniques have been developed that can train on all sample cases.

If the set $\{RS_1,...RS_i,...RS_n\}$ is ordered by some complexity measure $Cx(RS_i)$, then the best one is selected by $\min[Err(RS_i)]$.[3] Thus to solve this problem in practice, a method must induce and order $\{RS_i\}$ by $Cx(RS_i)$ and estimate each $Err(RS_i)$. Such methods have been developed for decision trees. Studies have shown that the minimization of a single complexity parameter, Cx, in addition to the error rate estimator, Err, adds a little bias to the estimates when used with resampling [Breiman, Friedman, Olshen, and Stone, 1984].

A variation of weakest-link pruning, also known as cost-complexity pruning [Breiman, Friedman, Olshen, and Stone, 1984], can be used to prune a rule set and form $\{RS_i\}$. Let the rule set RS_1 be the covering rule set. Each subsequent RS_{i+1} can be found by pruning RS_i at it weakest link. A rule set's weakest link can be defined as in equation 1, where $Errors(RS_i)$ is the number of errors RS_i makes on the training cases and $Size(RS_i)$ is the number of components in RS_i. The weakest link is the point at which the fewest new errors per deleted components are introduced. As in [Quinlan, 1987a], a rule set can be be pruned by deleting single rules or single components, where the weakest link is measured by WL_i. Repeated pruning of the least significant single component or rule to RS_i sequentially forms $\{RS_k\}$ and the global minimum is WL(i). The rule set at that point becomes the next RS_{i+1}. The process is repeated until the final RS_n is generated, where RS_n is single component rule of selecting the largest class.

$$WL_{i+1}=min\left[\frac{Err(RS_k)-Err(RS_i)}{Size(RS_i)-Size(RS_k)}\right] \quad \forall\ k\ prunes \qquad (1)$$

The application of weakest-link pruning results in an ordered series of decreasing complexity rule sets, $\{RS_i\}$. The complexity of RS_i can be measured in terms of WL_i or alternatively $Size(RS_i)$. With a large set of independent test cases and weakest-link pruning based on apparent error rate estimates, the true error rate of each RS_i can be estimated by $Err_{test}(RS_i)$, the error rate of RS_i on the test cases. With smaller samples, where thousands of test cases are not available, resampling is preferable and more accurate. As in [Breiman, Friedman, Olshen, and Stone, 1984], $\{RS_1,...RS_i,...RS_n\}$ are determined by weakest-link pruning on the complete training set. An n-fold cross-validation, typically 10-fold, is performed. In each fold k, an auxiliary rule set is induced using the training set of that fold. A new RS^k_i of complexity (approximately) equal to $Size(RS_i)$ is found by weakest link pruning. Test error-rates are obtained using the test cases corresponding to that fold. The average of the error rates over all the folds for each rule-set of $Size(RS_i)$, $Err_{cv}(RS_i)$, is the cross-validation estimate of the true error rate of RS_i.

Consistent with the minimum length description approach, each rule covers the original cases with only the weakest rules and components removed. As given, pruning a rule set is less stable and accurate than tree pruning because coverage of the pruned set is highly variable. While pruning a subtree retains full coverage of the data set, pruning rules can leave gaps in the coverage. Moreover, for RS_i of complexity $Size(RS_i)$, there may be a better RS_i' of $Size(RS_i)$. Unlike the decision tree induction where the nodes are fixed, RS_i can be refined by the swapping single components to minimize the apparent training error $Err_{app}(RS_i)$. The process of refining any rule set RS_i into RS_i' can be described as modifying RS_i such that $Err_{app}(RS_i')\leq Err_{app}(RS_i)$ and $Cx(RS_i')\leq Cx(RS_i)$. The rules are iteratively checked for the best component deletion, rule deletion, or component swap. Here the definition of "best" is minimum errors. Thus a rule set can be refined so that the refined rule set is smaller or equal in size to the original rule set and makes fewer or an equal number of errors than the original.

The net result of this process is an error rate estimate for varying complexity rule sets. A typical result is illustrated in Figure 4, where the results of resampling for the rheumatic disease application in Section 3 are summarized. For each rule set RS_i, Figure 4 lists the number of rules, the number

[2]The average results of 10 runs using 90% training and 10% testing cases, with 10 mutually exclusive test partitions.

[3]Or as in [Breiman, Friedman, Olshen, and Stone, 1984], the $\min\{Cx(RS_i)\}$ that is close (within one standard error) to $\min Err(RS_i)$.

RS_i i=	Rules	Cx	Err_{app}	Err_{test}	Test SE	Mean(Cx)	WL_i
1	11	18	.0000	.1074	.0282	17.9	.0
2	10	15	.0083	.0909	.0261	14.9	.3
3	9	13	.0165	.0909	.0261	13.0	.5
4*	6	7	.0661	.0744	.0239	7.0	1.0
5	6	6	.0826	.1322	.0308	6.0	2.0
6	4	4	.1322	.1322	.0308	4.0	3.0
7	3	3	.2975	.2975	.0416	3.0	20.0
8	2	2	.5372	.5620	.0451	2.0	29.0
9	1	1	.6529	.6529	.0433	1.0	14.0

Figure 4: Example of Summary Table

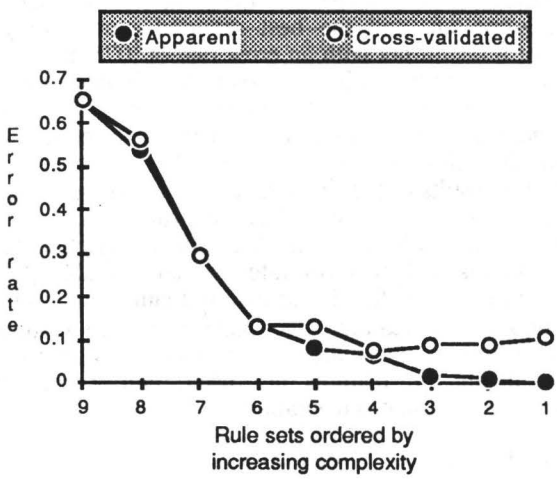

Figure 5: Error rates vs. Model Complexity

of rule components, the apparent error rate on the training cases, the test error rate (by cross-validation), the standard error of the test error, the average number of components over all cross-validated test sets, and the complexity measured by weakest-link pruning. Plotting the error rates for increasingly complex RS_i, as in Figure 5, illustrates the classical pattern of behavior for the apparent error rate on training cases versus the estimated true error on test cases. As model complexity increases, the apparent error rate decreases, but the true error rate flattens out and eventually increases.

2.4 Mixed Models for Reduced Complexity

While complexity within a given model can be measured in some common units, complexity measures for different models are not readily comparable. For example, we have adopted the number of rules components as the unit of complexity measure, where components are simple logical propositions. While a tree can be directly decomposed into a set of rules, where a path to a terminal node is a rule, these rules would share common components, and a complexity unit of tree nodes seems more appropriate. Complexity units measured in terms of weights are appropriate for linear discriminants and neural nets.

Our approach has been to minimize the complexity of rule sets. Although difficult to express in terms of single units, the true complexity of a solution could be further reduced by allowing for mixed models. For example, it is well-known that in (two dimensional) geometric terms, rule-based models find solutions that are parallel to the axes and cannot efficiently fit even simple linear functions between features such as x>y. This has led to a number of hybrid methods that incrementally embed alternative models such as piecewise linear discriminants [Breiman, Friedman, Olshen, and Stone, 1984] or perceptrons [Utgoff, 1988] within decision trees.

In contrast to those nonparametric, incremental methods, we have used the standard parametric linear discriminant[4] where the solution is posed as in equation 2, and for each Class i, $f_i(e)$, a linear function of the set of attributes e, is derived. An unknown pattern is classified by applying the functions and choosing the class whose linear magnitude is greatest. In addition, heuristic stepwise feature selection can be used to find a reduced complexity linear solution by reducing the original set of features $\{e_k\}$ to $\{e_j'\}$ where j<k, i.e a subset of the given features, reducing the number of weights that are estimated [James, 1985].

$$f_i(e) + ln(P(C_i) > f_j(e) + ln(P(C_j) \quad \forall\, i \neq j \tag{2}$$

The parametric stepwise linear discriminant is highly developed and has a strong tendency to produce comparable results on both the training and test cases. In our design, the discriminant function is completely derived prior to rule induction, and the results (on the training cases) are used to create artificial features. One binary higher-order feature per class is specified, where each feature is simply whether or not that class is selected during training by the linear discriminant. To the rule induction system, each of these higher-order features is no different than any of the original features. Unlike previous approaches where the numerical result of applying a single linear function is used, here the encoding is simplified to a binary feature that preserves the classification result of the discriminant. This results in an interesting interplay between the linear discriminant and an induced rule set. For example in Figure 6, Class1 is chosen when both LD1 and RBPS>109 are true, where LD1 is the selection of Class1 by the linear discriminant, and RBPS is a test result.

LD1 & RBPS>109	→	Class1
RBPS>151 & AGE<62	→	Class1
[TRUE]	→	Class2

Figure 6: Heart Disease Example of Mixed Model Rules

3 Results

To evaluate the efficacy of reduced complexity learning, four applications were considered. These applications are real-world problems with significant published results. Following the original publications, studies were performed and published by other researchers comparing different learning models, usually back-propagation neural nets compared with decision trees. Of particular note, decision trees performed relatively poorly in these published comparisons. Figure 7 summarizes the characteristics of the data sets for each application, and the train-and-test variations, such as leaving-one-out (Lv-1), used in the

[4]This is the standard linear discriminant found in statistical packages under the assumption of normality and equal covariance matrices.

original studies to estimate the true error rate.

Figure 8 summarizes the reported results of previous studies and our new results for a reduced complexity approach. Cx, the complexity measure, is measured in units of weights for neural nets or linear discriminants, propositions (rule components) for rules, and nodes for trees.[5] We review the results for each application.

Text-to-Speech Recognition

The NETTALK text-to-speech application was originally reported in [Sejnowski and Rosenberg, 1987] and a rigorous comparative study of ID3 decision trees and back-propagation neural nets was reported in [Diettrich, Hild, and Bakri, 1990]. While the larger goal is the recognition of spoken words, in the previous analyses, the problem is formulated as the recognition of "letters" or classes of phoneme-stress pairs. Although the problem fits the traditional classification mold of mutually exclusive classes, previous studies have used a relatively complex non-mutually exclusive encoding for the classes. This was done based on knowledge of the domain and its suitability for a neural net representation.[6] In our analysis, we return to the traditional classification approach. There are 47 phoneme classes and 4 stress classes but only 99 phoneme-stress pairs with more than 1 case in the training set. Despite over 7000 training cases, relatively few cases appear for many classes. The rule sets $\{RS_i\}$ are induced from the training set, and given the large number of test cases, the appropriate complexity fit, $Cx(RS_i)$, can be determined by the test cases.[7] The classes were ordered by decreasing predictive values of their rule sets.[8]

The previously reported best result was for a neural net with an error rate of .291 and over 27,000 weights. A reduced complexity rule-based solution was found with an error rate of .260 and 663 rule components.[9] The previously reported error rate for a tree-based solution was a relatively weak .344. However, with the 99-class representation, the error rate of an induced tree is the same as the Swap-1 rule-based solution although with far greater complexity.

Heart Disease Classification

An application to heart disease was originally reported in [Detrano et al., 1989] and a comparative study of (ID3) decision trees and (back-propagation) neural nets was reported in [Shavlik Mooney and Towell, 1991]. In the comparative study, the average test results for ten 70% training and 30% testing sets were used. There, the best result was for a neural net with an error rate of .194 and 86 weights. The result of a 10-fold (or a 3-fold) cross-validation for Swap-1 yields a simple 4-rule, 6-component solution with an estimated error rate of .215, as compared with the reported .288 for the decision tree. Unlike the results in [Shavlik Mooney and Towell, 1991], results were improved by pruning the rule sets and not using binary encodings of numerical variables. When the linear discriminant is added with the original features, the rule set RS_{best} is simply the linear discriminant which has an estimated error rate of .176 with 16 weights.[10]

DNA Pattern Analysis

In [Towell, Shavlik, and Noordeweir, 1990] several learning methods are compared on a DNA pattern recognition task. In addition, a human expert's "theory," described in terms of a grammar, is reformulated as a neural network and refined -- yielding somewhat better results than pure empirical learning systems.[11] Error rates are estimated by leaving-one-out, and the best reported pure learning result is for a neural net with an error rate of .075 and over 3600 weights. The tree solution does relatively poorly with an error rate of .179, and a reduced complexity 3-rule solution with 5 components is better with an error rate of .132. Still the rule-based solution is not fully competitive. Including the results of the linear discriminant in the feature set, results in an induced rule set that is pruned back to the linear discriminant alone. This discriminant function has only 12 weights and an error rate estimate of .047. The reduced complexity of this linear discriminant is due to the success of stepwise feature selection.[12] With 228 features and only 106 cases, there are far too many weights to be estimated if all the features are used, and a reduced complexity discriminant is desirable.

Rheumatic Disease Diagnosis

A very preliminary version of a rule-based expert system for diagnosing rheumatic diseases was described and evaluated in [Lindberg,et al., 1980]. The knowledge base and data were later used in heuristic refinement systems [Ginsberg, Weiss, and Politakis, 1988] that could modify a theory, i.e. an expert-specified rule base, to improve its performance. These systems were restricted to minor refinements that would assure the preservation of the expert's knowledge base close to its original form. A more radical "theory revision" approach uses the theory, i.e the expert knowledge base, as a starting point and allows massive changes to the rule set. Such an approach was taken in [Ginsberg, 1990] where these same data were used to evaluate a theory revision method (RTLS). There, it was reported that that the leaving-one-out estimate of the error rate for the five class problem is zero. Because this new rule set is radically different from the expert's knowledge base, it is worthwhile considering how well a pure empirical learning system might do.

---

[5]Because the tree complexities were not published, the complexity of the trees in Figure 8 was recomputed by running CART.

[6]Further details of the application and the class encodings can be found in [Diettrich, Hild, and Bakri, 1990]. There, the results for the neural net or decision tree were further processed by a nearest neighbor method. We used the exact data sets of [Diettrich, Hild, and Bakri, 1990].

[7]The error rate in Figure 8, is the minimum complexity rule set whose error rate is within 1 standard error of the minimum error rate. Because the test set is so large and the rule sets are pruned without looking at the test cases, these estimates are reliable. Skeptics will get the same answer by estimating the appropriate complexity fit solely from the training set using a 50% training subsample and a 50% testing subsample. For all the NETTALK variations we tried, this was a pruning complexity of 1 case per pruned component.

[8]These predictive values were estimated by 2-fold cross-validation of the rule sets induced for each class C_i vs. not C_i.

[9]With 99 classes, we were unable to get a useful linear discriminant.

[10]The estimate of .176 is the average of three 3-fold cross-validations. The estimate for a 10-fold cross-validation is .168.

[11]In [Towell, Shavlik, and Noordeweir, 1990] a human assisted solution was estimated at an error-rate of 0.038.

[12]During leaving-one-out estimation, feature selection was repeated (at a 10 significance level) for each of the 106 train-and-test cycles.

Dataset	Number of Training Cases	Number of Test Cases	Feature-type	Number of Features	Number of Classes in Training Set
NETTALK	7229	7242	Boolean	203	99
HEART	297	(10) 70% / 30%	Numerical	13	2
DNA	106	Lv-1	Boolean	228	2
RHEUM	121	Lv-1	Boolean and Numerical	140	5

Figure 7: Dataset Characteristics

Dataset	Previously Reported Best Results			Tree Results		Swap-1 Results						
						Rules only			Rules with Discriminants			
	Err	Type	Cx	Err	Cx	Err	Rules	Cx	Err	Rules	Cx-R	Cx-LD
NETTALK	0.291	NN	27626	0.344	3477	0.260	253	663	-	-	-	-
HEART	0.194	NN	86	0.288	109	0.215	4	6	0.176	2	2	16
DNA	0.075	NN	3698	0.179	25	0.132	3	5	0.047	2	2	12
RHEUM	> 0.074	rule	3000	0.372	63	0.074	6	7	Not Applicable			

Figure 8: Comparison of Best Reported, Decision Tree and Swap-1 Results

From a purely empirical learning perspective, the previous (RTLS) solution is far too complex with 600 rules to provide a good complexity fit to the data.[13] The true error rate of RTLS is also higher than cited because error rates were measured in a non-traditional manner. Some ties were scored as correct, and answers were marked correct when they agreed with the output of a second refinement program, SEEK2. However, the output of SEEK2 had a leaving-one-out error rate estimate of .074, so the true error rate of RTLS is >.074.

Most empirical learning systems cannot handle this dataset because almost half the feature values are missing. Using CART to induce a decision tree, with its surrogate strategy for handling missing values, yields a high error rate of .405. Yet with Swap-1, a 6-rule, 7-component solution can be found that has a leaving-one out estimate of .074. Even with the missing values, the covering of the 121 cases is quite compact. With so many missing values, the advantage of the rule-based solution over the decision tree can be traced to the non-mutual exclusivity of the rules.[14]

4 Discussion

We have presented better solutions than those previously reported for four significant classification applications. These solutions are superior in terms of estimated error rates, but equally important are far less complex than the previously reported solutions. We were particularly interested in these applications because decision trees performed relatively poorly compared to competitive learning models. The reduced complexity rule induction approach was readily able to to exceed the performance of the decision trees. However, in two instances, the pure rule-based solution was still not competitive with alternative methods. By combining the results of the stepwise parametric linear discriminant with the original feature set, we were able to exceed previous results, at the expense of a mixed model solution. Interestingly, in two applications the method indicated to use solely the linear discriminant. In other applications, the rules can exhibit much interplay between the discriminant functions and the original features. We showed how a traditional classifier model can be incorporated into a rule-based solution, and a method was presented that arbitrates between the two models. There are many applications were pure rule-based solutions are a priori preferred by humans, and some applications where linear classifiers do not yield good results [Weiss and Kapouleas, 1989].

The central theme of the learning methods we have described is the reduction of rule complexity. We have presented an architecture for minimizing complexity of rules by reducing the number of rules and components and maximizing rule coverage of cases by iteratively swapping rule components. The goal of minimizing rule complexity is entirely consistent with the goal of maximizing predictive accuracy. To accomplish this task we have also borrowed many techniques, including weakest link pruning and resampling, that have proven effective in other applied learning systems.

While our approach is directed towards a rule-based solution, the notion of minimizing complexity is not restricted to any particular model. Neural net solutions that are compact and minimize the number of weights are also likely to increase predictive accuracy. For the applications we cited, similar simplifications to other models may yield improved results. For example, the previously used NETTALK neural net model might benefit from a reduced number of weights. The limiting factor is not only the specific model, but also the effectiveness of the learning technique and computational time. While the Swap-1 approach may appear computationally overwhelming, it is highly bounded. For instance, when the number of errors for a candidate swap exceeds those for the current best swap,

[13]The rule set had 600 rules. We approximated 5 components per rule and 3000 total components. In the ground-form, with no intermediate hypotheses, there were 34,522 rule components.

[14]Although the results are quite good, caution is still warranted. With missing values, there is always the question of whether the prediction is made based on a feature value or on the event that a value is missing.

that candidate can be immediately dropped. For all applications except NETTALK, a 10-fold cross-validation required only several minutes of Sparcstation-1 time. The NETTALK application required on the order of 5 days. Still, with steadily increasing available computational power, we can look forward to more computationally intensive attempts to extract the maximum amount of information from sample data.

Acknowledgments

This research was supported in part by NIH Grant P41-RR02230. We thank Kevin Kern for his programming support and our colleagues for supplying the data.

References

[Breiman, Friedman, Olshen, and Stone, 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. *Classification and Regression Tress.* Monterrey, Ca.: Wadsworth, 1984.

[Clark and Niblett, 1989] Clark, P. and Niblett, T. "The CN2 Induction Algorithm." *Machine Learning.* 3 (1989) 261-283.

[Detrano et al., 1989] Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S. and Froelicher, V. "International application of a new probability algorithm for the diagnosis of coronary artery disease." *American Journal of Cardiology.* 64 (1989) 304-310.

[Diettrich, Hild, and Bakri, 1990] Diettrich, T., Hild, H. and Bakri, G. "A comparative study of ID3 and back-propagation for english text-to-speech mapping." In *Proceedings of the seventh International Conference on Machine Learning.* 1990, 24-31".

[Ginsberg, 1990] Ginsberg, A. "Theory reduction, theory revision and retranslation." In *Proceedings of AAAI-90.* Boston, MA., 1990, 777-782.

[Ginsberg, Weiss, and Politakis, 1988] Ginsberg, A., Weiss, S., and Politakis, P. "Automatic Knowledge Base Refinement for Classification Systems." *Artificial Intelligence.* 35 (1988) 197-226.

[Highleyman, 1962] Highleyman, W. "The Design and Analysis of Pattern Recognition Experiments." *Bell System Technical Journal.* 41 (1962) 723-744.

[Indurkhya and Weiss, 1990] Indurkhya, N. and Weiss, S. "Iterative Rule Induction Procedures." In *Proceedings of PRICAI-90: First Pacific Rim International Conference on Artificial Intelligence 1990.* Japanese Society for Artificial Intelligence, Japan, 1990, 649-654.

[Indurkhya and Weiss, 1991] Indurkhya, N. and Weiss, S. "Iterative Rule Induction Methods." *Applied Intelligence.* (1991).

[James, 1985] James, M. *Classification Algorithms.* New York: Wiley, 1985.

[Lin and Kernighan, 1973] Lin, S. and Kernighan, B. "An Efficient Heuristic for the Traveling Salesman Problem." *Operations Research.* 21:2 (1973) 498-516.

[Lindberg, et al., 1980] Lindberg, D., Sharp, G., Kingsland, L., Weiss, S., Hayes, S., Ueno, H. and Hazelwood, S. "Computer-based rheumatology consultant." In *MEDINFO-80: Proceedings of the third world conference on medical informatics 1980.* 1980, 1311-1315.

[Michalski, Mozetic, Hong, and Lavrac, 1986] Michalski, R., Mozetic, I., Hong, J., and Lavrac, N. "The Multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains." In *Proceedings of AAAI-86.* Philadelphia, Pa., 1986, 1041-1045.

[Pagallo, 1989] Pagallo, G. "Learning DNF by Decision Trees." In *Proceedings of IJCAI-89.* Detroit, Michigan, 1989, 639-644.

[Pagallo and Haussler, 1989] Pagallo, G. and Haussler, D. "Two Algorithms That Learn DNF By Discovering Relevant Features." In *International Workshop on Machine Learning.* Ithaca, NY, 1989, 119-123.

[Quinlan, 1986] Quinlan, J. "Induction of Decision Trees." *Machine Learning.* 1 (1986) 81-106.

[Quinlan, 1987a] Quinlan, J. "Simplifying Decision Trees." *International Journal of Man-Machine Studies.* 27 (1987) 221-234.

[Quinlan, 1987b] Quinlan, J. "Generating Production Rules from Decision Trees." In *Proceedings of IJCAI-87.* Milan, Italy, 1987, 304-307.

[Quinlan and Rivest, 1989] Quinlan, J. and Rivest, R. "Inferring Decision Trees Using the Minimum Description Length Principle." *Information and Computation.* 80 (1989) 227-248.

[Rissanen, 1978] Rissanen, J. "Modeling by Shortest Data Description." *Automatica.* 14 (1978) 465-471.

[Rissanen, 1987] Rissanen, J. "Stochastic Complexity." *Journal of Royal Statistical Society, B.* 49:3 (1987) 223-239.

[Sejnowski and Rosenberg, 1987] Sejnowski, T. and Rosenberg, R. "Parallel networks that learn to pronounce english text." *Complex Systems.* 1 (1987) 145-168.

[Shavlik Mooney and Towell, 1991] Shavlik, J., Mooney, R. and Towell, G. "Symbolic and Neural learning Algorithms: An experimental comparison." *Machine Learning.* 6 (1991).

[Stone, 1974] Stone, M. "Cross-Validatory Choice and Assessment of Statistical Predictions." *Journal of the Royal Statistical Society.* 36 (1974) 111-147.

[Towell, Shavlik, and Noordeweir, 1990] Towell, G., Shavlik, J. and Noordeweir, M. "Refinement of approximate domain theories by knowledge-based neural networks." In *Proceedings ofAAAI-90.* Boston, MA., 1990, 861-866.

[Utgoff, 1988] Utgoff, P. "Perceptron Trees: A Case Study in Hybrid Concept Representation." In *Proceedings of AAAI-88.* Minneapolis, 1988, 601-606.

[Wallace and Freeman, 1987] Wallace, C. and Freeman, P. "Estimation and Inference by Compact Encoding." *J. Royal Statistical Society B.* 49B:3 (1987) 240-265.

[Weiss and Kapouleas, 1989] Weiss, S. and Kapouleas, I. "An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods." In *Proceedings of IJCAI-89,* Michigan, 1989, 781-787.

[Weiss, Galen, and Tadepalli, 1990] Weiss, S., Galen, R., and Tadepalli, P. "Maximizing the Predictive Value of Production Rules." *Artificial Intelligence.* 45 (1990) 47-71.