Using Rule Sets to Maximize ROC Performance

Tom Fawcett
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, California 94304
tfawcett@acm.org

Abstract

Rules are commonly used for classification because they are modular, intelligible and easy to learn. Existing work in classification rule learning assumes the goal is to produce categorical classifications to maximize classification accuracy. Recent work in machine learning has pointed out the limitations of classification accuracy: when class distributions are skewed, or error costs are unequal, an accuracy maximizing rule set can perform poorly. A more flexible use of a rule set is to produce instance scores indicating the likelihood that an instance belongs to a given class. With such an ability, we can apply rulesets effectively when distributions are skewed or error costs are unequal. This paper empirically investigates different strategies for evaluating rule sets when the goal is to maximize the scoring (ROC) performance.

1. Introduction

Rules are commonly used in data mining because of several desirable properties: they are simple, intuitive, modular, and straightforward to generate from data. Some work concentrates on association rule mining, in which individual rules are valued for the insight they can bring. Another area of work, termed *classification rule learning*, strives to generate rules that collectively have good classification performance [6]. It is the use of rules for classification with which this paper is concerned. Existing methods strive to optimize classification decisions, usually by maximizing accuracy (or equivalently, minimizing error rate) on a training set. They usually try to construct small, compact rule sets while achieving high accuracy.

Recent work in machine learning and data mining has demonstrated problems with using accuracy as a metric [11]. It can be irrelevant or misleading when classes are imbalanced or when misclassification costs are unequal. Ideally, costs should be taken into account and accuracy maximization should be replaced with cost minimization. If error costs and class distributions are known exactly, a cost minimizing problem can sometimes be transformed into an accuracy maximizing one. However, in many cases neither costs nor class distributions are known exactly, and both can change over time and over contexts [9, 5]. As conditions change, a high accuracy rule set may produce sub-optimal classifications. Data mining that uses rules for real world classification tasks will eventually face these problems.

One way to allow flexibility under uncertain conditions is to use probabilities. Instead of requiring that a classifier produce a hard (discrete) class decision for each instance, we can use the classification model to generate an estimated probability that an instance belongs to a specific class. Given such probabilities, simple decision theory can be used to generate thresholds under various assumptions of error costs and class distributions. This enables robust classification systems that can operate in uncertain and changing environments [10].

Probabilistic classification has been investigated with other model classes such as neural networks [14] and ensembles of decision trees [8, 21]. These models tend to be much more complex than rule sets and may not have rules' appealing properties of modularity and intelligibility. An open question in data mining is how to use a set of rules to produce reliable instance probabilities. This approach raises issues of how to generate scores from rules, how to combine scores from different rules, and how to select rules for inclusion in a rule set.

This paper explores these issues empirically. We employ the area under an ROC curve (commonly referred to as the AUC) to evaluate and compare strategies [2]. The remainder of this paper is structured as follows. We begin by discussing ROC curves and the AUC as a tool for measuring instance scoring performance. We discuss rules and rulesets, and how they can be used for classification. We then describe a number of experiments investigating the use of classification rules for scoring instances. We conclude by discussing future work and some related issues.

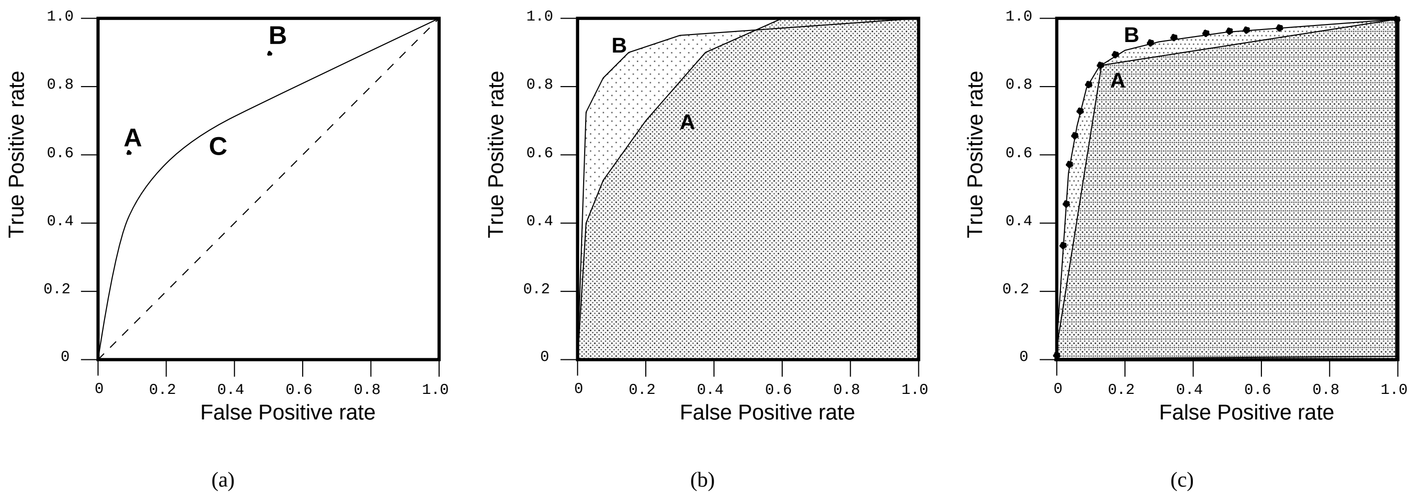

Figure 1. ROC graphs and area under ROC curves.

2. ROC graphs and the AUC

To evaluate probabilistic classifiers we adopt Receiver Operating Characteristics (ROC) analysis. ROC graphs have long been used in signal detection theory to depict the tradeoff between hit rates and false alarm rates of classifiers [4, 17]. ROC analysis has been extended for use in visualizing and analyzing the behavior of diagnostic systems [16].

A discrete classifier applied to a test set generates two important statistics. The **True Positive rate** (also called hit rate and recall) of a classifier is:

$$\text{TP rate} \approx \frac{\text{positives correctly classified}}{\text{total positives}}$$

The **False Positive rate** (also called false alarm rate) of the classifier is:

$$\text{FP rate} \approx \frac{\text{negatives incorrectly classified}}{\text{total negatives}}$$

On an ROC graph, TP rate is plotted on the Y axis and FP rate is plotted on the X axis.

A discrete classifier—one that outputs only a class label—produces an *(FP rate,TP rate)* pair, so it corresponds to a single point in ROC space. Classifiers A and B in Figure 1a are discrete classifiers.

Several points in ROC space are useful to note. The lower left point $(0, 0)$ represents the strategy of never issuing a positive classification; such a classifier commits no false positive errors but also gains no true positives. The opposite strategy, of unconditionally issuing positive classifications, is represented by the upper right point $(1, 1)$. Any classifier that randomly guesses the class will produce performance on the diagonal line $y = x$. The point $(0, 1)$ represents perfect classification. Informally, one point in ROC space is better than another if it is to the northwest (TP rate is higher, FP rate is lower, or both) of the first.

The diagonal line $y = x$ represents the strategy of randomly guessing a class, and any classifier that appears in the lower right triangle performs worse than random guessing. This triangle is therefore usually empty.

Classifiers can often be coerced into producing a probability estimate or numerical rank for each instance. Such a *ranking* or *scoring* classifier can be used with a threshold to produce a binary classifier: if the classifier output is above the threshold, the classifier produces a **Y**, else a **N**. Each threshold value produces a different point in ROC space, so varying the threshold from $-\infty$ to $+\infty$ produces a curve through ROC space. An ROC curve illustrates the error tradeoffs available with a given classifier. Figure 1a shows the curve of a probabilistic classifier, C, in ROC space. A more thorough discussion of ROC curves may be found in Provost and Fawcett's article [10].

An important point about ROC graphs is that they measure the ability of a classifier to produce good *relative* instance rankings. A classifier need not produce accurate, calibrated probability estimates; it need only produce relative accurate scores that serve to discriminate positive and negative instances. Thus, although this paper refers to rule sets used as probabilistic classifiers, these classifiers only need to produce good relative scores.

2.1. Area under an ROC curve (AUC)

An ROC curve is a two-dimensional depiction of classifier performance. To compare classifiers we often want to reduce ROC performance to a single number representing average expected performance. A common method is to calculate the area under the ROC curve, abbreviated **AUC** [2, 7]. Since the AUC is a portion of the area of the unit square, its value will always be between 0 and 1.0. However, because random guessing produces the diagonal line between $(0, 0)$ and $(1, 1)$, which has an area of 0.5, no realistic classifier should have an AUC less than 0.5.

The AUC has an appealing statistical property: the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. This is equivalent to the Wilcoxon test of ranks [7]. It is possible for a high-AUC classifier to perform worse in a specific region of ROC space than a low-AUC classifier, but in practice the AUC performs very well and is often used when a general measure of predictiveness is desired.

Figure 1b shows the areas under two ROC curves, A and B. B has greater area and therefore better average performance. Figure 1c shows the area under the curve of a binary classifier A and a scoring classifier B. Classifier A represents the performance of B when B is used with a single, fixed threshold. Though the performance of the two is equal at the fixed point (B's threshold), B's performance becomes inferior to A further from this point.

2.2. AUC with multiple classes

The two ROC axes represent tradeoffs between false positives and true positives with two classes. ROC analysis has been extended to multiple classes [15], but the result in general is non-intuitive and computationally expensive. In practice, n classes are commonly handled by producing n different ROC graphs. Let C be the set of all classes. ROC graph i plots the classification performance using class c_i as the positive class and all other classes $c_{j \neq i} \in C$ as the negative class. Each such graph yields an AUC area.

For a single probabilistic classifier this produces n separate curves with n different AUC values. The AUC values can be combined into a single weighted sum where the weight of each class c_i is proportional to the class's prevalence in the training set:

$$AUC_{total} = \sum_{c_i \in C} AUC(c_i) \cdot p(c_i)$$

3. Rules and classification

A rule is a conjunction of conditions, the satisfaction of which implies membership in a class. For simplicity, consider a two-class problem with classes **p** and **n**. An example of a simple rule and some of its performance statistics is:

$$\mathbf{x_1} \wedge \mathbf{x_2} \wedge \mathbf{x_3} \longrightarrow \mathbf{p}$$

TP=15, P=100, TPrate=.15

FP=2, N=200, FPrate=.01

The second line specifies that within the dataset, 15 **p** examples satisfy $x_1 \wedge x_2 \wedge x_3$ (True Positives). There are 100 **p** examples altogether, yielding a true positive rate (TPrate) of .15. The rule matches two **n** examples (False Positives). There are 200 **n** examples altogether, yielding a false positive rate (FPrate) of .01.

3.1. Rule sets

For the purpose of classification, rules are usually aggregated into a rule set. When the set is ordered this is called a *decision list* [13]. To classify an instance, each rule in the list is tried in sequence, and the first rule whose conditions are satisfied determines the hypothesized class. A decision list is an "**if-then-elseif-...-else-**" formulation of a boolean concept. General cases usually appear toward the end of the list with more specific cases (exceptions) placed at the front. If no rule matches, the final classification is the most prevalent class.

Given a decision list and an instance set, a 2×2 confusion matrix can be generated, representing the classification performance:

		Hypothesized class	
		p	**n**
Actual	**p**	TP	FN
class	**n**	FP	TN

This adds the TN (True Negatives) and FN (False Negatives) statistics. From this matrix we can estimate the error rate of the classifier:

$$\begin{aligned} \text{Error rate} &= \frac{FP + FN}{TP + TN + FP + FN} \\ \text{Accuracy} &= 1 - \text{Error rate} \end{aligned}$$

Error rate weights FP and FN equally. If we have separate error cost functions $c(FP)$ and $c(FN)$, we instead want to measure expected cost:

$$\text{Cost} = FP \cdot c(FP) + FN \cdot c(FN)$$

Given such cost functions, simple decision analysis provides a way to determine each instance's optimal classification, if we can get an estimate of its class probability. For each instance I we should hypothesize the positive class **p** if:

$$[1 - p(\mathbf{p}|I)] \cdot c(FP) \; < \; p(\mathbf{p}|I) \cdot c(FN)$$

Note that the prior $p(\mathbf{p})$ is incorporated into the posterior estimate $p(\mathbf{p}|I)$. To minimize overall error cost, we thus need a way to estimate instance probabilities $p(\mathbf{p}|I)$. Given the statistics of a matching rule, we can generate a probability estimate simply as

$$p(\mathbf{p}|I) \approx \frac{\text{TP}}{\text{TP} + \text{FP}}$$

This measure is commonly called the *confidence* of the rule. This equation is often used in a Laplace corrected form:

$$p(\mathbf{p}|I) \approx \frac{\text{TP} + 1}{\text{TP} + \text{FP} + |C|}$$

Laplace correction smoothes probability estimates when the number of instances covered by a rule is small.

As mentioned above, we do not need accurate instance probabilities $p(\mathbf{p}|I)$ to make the classification decision. We only need good relative instance scores. Given a test set and knowledge of performance conditions, we can derive a suitable threshold [10].

3.2. Resolving rules

If rules were mutually exclusive, each instance would match at most one rule and a probability estimate could be taken directly from the rule's confidence. However, rules can overlap and multiple rules may "claim" an instance, resulting in potentially conflicting classifications and instance scores. Resolving these into a single class and score is called the *resolution problem*. Given the information available from each rule, a number of strategies can be enumerated:

1. **Random selection (RAND)**. A random matching rule is chosen, and its class and confidence are used as the winning class and score. RAND provides a baseline against which other strategies may be compared.

2. **First matching rule (FIRST)**. Each rule in a list is tested in turn and the first matching rule wins. The rule's class and confidence become the winning class and score. This strategy is appropriate when order is imposed, as in decision lists [13]. When rules are sorted by decreasing confidence, this method selects the highest confidence rule.

3. **Equal voting (VOTE)**. Each rule is tested, and every matching rule contributes a single vote for its class. The majority class wins. The score assigned to the instance is the fraction of votes won by the majority.

4. **Weighted voting (WVOTE)**. This is similar to VOTE, but each matching rule votes with a strength of its confidence. The class with the highest summed confidence wins, and the score is the average confidence. The rationale for weighted voting is that high confidence rules should have more influence than lower confidence ones.

5. **Lowest false positive rate (LFPR)**. Among matching rules, the rule with the lowest false positive rate is selected. Its class and confidence are used as the class and score. The rationale for LFPR is to choose the matching rule that has the least chance of committing a false positive error.

With each method, if no rule fires on an instance, the majority class is used and the majority class prevalence is used as the instance score.

4. Experiments

Given the discussion above, we investigate the following questions:

1. Is instance scoring really necessary for good AUC performance? Perhaps rule sets already produce good classification performance throughout ROC space. Some prior work has found this not to be true with other model classes [11], but this hypothesis is worth testing this hypothesis with rule set classifiers.

2. What is the effect of various rule set resolution strategies on a rule set's instance scores?

3. How well do rules perform relative to other methods for scoring instances?

4.1. Rule induction methods

Two rule induction methods were used in the experiments below. They are fairly different in operation and in results.

C4.5rules [12] is a companion program to C4.5 which creates rule sets by post-processing decision trees. C4.5rules begins by constructing a rule from each path to a leaf node, with each attribute test in the path becoming a conjunct in the rule. This results in potentially a large number of rules, but the initial set of rules is mutually exclusive. C4.5rules then examines the rules, testing each conjunct to determine whether it is necessary; if rule accuracy is unaffected, the conjunct is deleted. After deleting conjuncts, the resulting rule set is no longer mutually exclusive and exhaustive, so C4.5rules performs several final steps for improving the rule set. Finally it groups the rules by class, based on the number of false positive errors committed by each class subgroup. In the experiments reported here, both C4.5 and C4.5rules were used with their default settings.

	Resolution strategies					Number of rules	
Dataset	**RAND**	**FIRST**	**LFPR**	**VOTE**	**WVOTE**	**Generated**	**Fired**
Breast-wisc	69.9 ± 6.8	50.1 ± 0.5	97.6 ± 1.3	95.1 ± 2.6	94.7 ± 3.4	306.5	112.3
Car	74.0 ± 2.7	61.7 ± 7.0	92.3 ± 1.7	71.2 ± 4.5	94.3 ± 1.4	107.6	6.6
Cmc	63.2 ± 3.8	63.7 ± 4.1	63.7 ± 4.2	61.9 ± 4.3	63.9 ± 4.0	196.6	3.2
Covtype	69.0 ± 1.9	64.4 ± 3.8	72.9 ± 2.1	66.6 ± 1.8	73.3 ± 1.5	1416.6	32.2
Crx	70.0 ± 7.5	75.0 ± 4.3	83.9 ± 5.1	88.4 ± 5.0	90.2 ± 4.2	758.5	69.8
German	58.2 ± 4.7	67.1 ± 4.9	66.2 ± 5.3	62.3 ± 5.6	71.9 ± 4.9	807.5	36.0
Glass	68.4 ± 10.1	70.1 ± 9.1	71.2 ± 10.0	74.4 ± 10.0	71.7 ± 10.5	183.7	35.0
Image	88.8 ± 2.1	86.0 ± 4.1	93.3 ± 1.4	80.9 ± 2.4	92.3 ± 1.6	811.4	66.2
Kr-vs-kp	66.2 ± 3.0	52.9 ± 3.9	92.6 ± 1.5	84.8 ± 3.3	88.8 ± 2.4	2328.3	340.0
Mushroom	90.1 ± 1.8	53.4 ± 2.5	100.0 ± 0.0	99.4 ± 0.1	99.9 ± 0.1	2362.2	131.7
Nursery	90.2 ± 0.6	93.7 ± 0.4	97.1 ± 0.2	93.6 ± 0.6	96.0 ± 0.3	606.6	12.1
Promoters	51.8 ± 12.6	47.1 ± 20.6	72.6 ± 13.3	76.8 ± 14.4	83.5 ± 16.2	7432.2	334.0
Sonar	48.3 ± 11.2	57.0 ± 11.0	59.4 ± 13.7	63.5 ± 12.4	65.8 ± 12.8	10075.7	1869.2
Splice	57.8 ± 2.3	45.9 ± 2.5	74.6 ± 2.2	70.6 ± 2.8	87.3 ± 1.6	8406.8	214.9

Table 1. Mean and standard deviation of AUCs using RL rules with various resolution strategies. The final two columns give the average number of rules generated and the average number of rules fired on each instance.

RL [3] is a MetaDENDRAL-style rule learner that performs a general-to-specific search of the space of conjunctive rules. This type of rule-space search is described in detail by Webb [18]. RL uses a beam search for rules whose coverage and confidence are above user-defined thresholds. In the experiments reported here, a beamsize of 100 was used along with rule constraints of confidence greater than 0.60, coverage greater than two instances, and no more than four conjuncts per rule. A Laplace corrected version of the confidence equation was used, to compensate for small sample sizes.

With its default settings, RL only finds rules that cover examples not previously covered by other rules. In these experiments, RL was allowed to generate redundant rules in order to experiment with the effects of rule overlap. It is important to note that, unlike C4.5rules, RL does not try to produce a rule set that maximizes classification accuracy.

4.2. Datasets

Fourteen data sets were selected from the UCI Repository [1]. In general, datasets were avoided that had extreme class skews since the purpose of this paper is not to experiment with learning under skewed distributions. Datasets were chosen that could produce reasonable performance with standard rule learners.

Each experiment reported below was performed using 10-fold cross-validation on the datasets. Means and standard deviations of the experimental results are given. For readability, AUCs are reported as percentages of the total possible so they range from 0 to 100 instead of 0 to 1.

4.3. The effect of resolution strategy

Tables 1 and 2 show the effect of different rule resolution strategies using rules from RL and C4.5rules, respectively. Several observations can be made from these results. From the last two columns in each table we can see that RL generated far more rules for each dataset than C4.5rules did, in some cases by one or two orders of magnitude. Also, the number of rules fired on average per instance is far greater for RL than for C4.5rules, indicating that rule contention is considerably higher for the rule sets created by RL. Neither result is surprising. The RL parameters were chosen so that it would generate a large number of overlapping rules, resulting in high contention. On the other hand, C4.5rules begins with a set of mutually exclusive rules; although rules can overlap after conjunct deletion, they will not otherwise conflict.

The results of this contention are seen in the effect of the resolution strategy. There is little difference between resolution strategies with the C4.5rules results because there is little contention to resolve. The RL results in Table 1 show much greater variability.

The differences between resolution strategies are evaluated more carefully in Table 3, in which they are compared in pairs using the mean AUC for each dataset shown in Table 1. Results are given for both the Sign Test (which ignores difference magnitudes) and the Wilcoxon Matched-Pair Signed-Ranks Test (which takes magnitude into account). Each test result is the probability that the first strategy is indistinguishable from the second in performance. RAND and FIRST both perform poorly, and are indeed virtually indistinguishable in performance. Uni-

Dataset	Resolution strategies RAND	FIRST	LFPR	VOTE	WVOTE	Number of rules Generated	Fired
Breast-wisc	96.6 ± 3.2	97.6 ± 3.0	97.5 ± 2.9	95.8 ± 2.9	97.4 ± 3.6	8.2	2.2
Car	97.6 ± 0.7	98.4 ± 0.7	98.4 ± 0.7	96.3 ± 0.8	98.3 ± 0.7	78.6	1.3
Cmc	66.1 ± 5.4	66.8 ± 5.0	66.6 ± 5.2	67.0 ± 2.6	66.5 ± 4.9	39.1	1.2
Covtype	81.5 ± 1.2	83.1 ± 1.5	82.6 ± 1.4	80.3 ± 1.2	82.2 ± 1.4	63.5	1.6
Crx	89.1 ± 3.7	90.9 ± 2.6	89.2 ± 4.2	84.7 ± 3.3	90.1 ± 3.4	12.9	1.9
German	68.2 ± 9.0	68.1 ± 9.7	67.6 ± 10.1	67.7 ± 8.4	67.9 ± 9.6	23.4	1.1
Glass	76.9 ± 5.0	77.1 ± 5.2	77.1 ± 5.2	75.8 ± 8.7	75.7 ± 5.9	12.2	1.0
Image	99.0 ± 0.5	99.1 ± 0.5	99.1 ± 0.5	98.0 ± 0.5	99.0 ± 0.5	28.6	1.5
Kr-vs-kp	99.7 ± 0.2	99.9 ± 0.1	99.9 ± 0.1	99.5 ± 0.4	99.7 ± 0.2	26.3	1.9
Mushroom	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	99.8 ± 0.0	100.0 ± 0.0	11.5	1.2
Nursery	99.7 ± 0.1	99.8 ± 0.1	99.8 ± 0.0	99.6 ± 0.1	99.8 ± 0.1	336.8	1.6
Promoters	89.2 ± 10.0	89.4 ± 10.6	89.9 ± 11.2	88.9 ± 9.9	88.4 ± 12.8	8.0	1.2
Sonar	77.7 ± 11.6	78.2 ± 12.3	77.7 ± 10.9	76.4 ± 11.2	77.8 ± 13.7	9.1	1.3
Splice	97.1 ± 0.7	97.5 ± 0.6	97.4 ± 0.6	97.2 ± 0.8	97.2 ± 0.7	76.2	2.3

Table 2. Mean and standard deviation of AUCs using C4.5rules with various resolution strategies. The final two columns give the average number of rules generated and the average number of rules fired on each instance.

form unweighted voting (VOTE) performs better. The two measures that take rule statistics into account, LFPR and WVOTE, perform best of all. WVOTE appears to have an advantage over LFPR in these domains but the difference is statistically inconclusive.

Pair	Wins-Ties-Losses	Sign test	WMPSR test
RAND vs FIRST	8-0-6	0.791	0.194
RAND vs LFPR	0-0-14	0.000	0.000
RAND vs VOTE	4-0-10	0.180	0.011
RAND vs WVOTE	0-0-14	0.000	0.000
FIRST vs LFPR	1-0-13	0.002	0.000
FIRST vs VOTE	4-0-10	0.180	0.013
FIRST vs WVOTE	0-0-14	0.000	0.000
LFPR vs VOTE	10-0-4	0.180	0.194
LFPR vs WVOTE	5-0-9	0.424	0.153
VOTE vs WVOTE	2-0-12	0.013	0.003

Table 3. Pairwise comparisons of the results of resolution strategies from Table 1.

4.4. Benefits of instance scoring

Another question is whether instance scoring is useful. Perhaps the classification done by C4.5rules is already sufficient to provide good performance over the entire ROC space. To evaluate this hypothesis, the AUC performance of the C4.5rules was measured as if the rules were evaluated directly for classification; that is, as if they were interpreted by the `consultr` program that comes with C4.5. This evaluation produces a single "accuracy point" in ROC space. This constitutes an ROC curve whose area can be measured. If probabilistic interpretation of rules has benefits for predictiveness, we might expect a situation as shown in Figure 1c, where classifier B (probabilistic) has a larger area than classifier A (discrete).

Figure 2 shows ROC curves from a C4.5rules rule set on the Covtype domain. Discrete classification yields an FP rate of .13 and a TP rate of .70. Connecting this point to $(0, 0)$ and $(1, 1)$ yields an ROC "curve" with flat sides. If the same rules are used for instance scoring, a curve of greater area can be produced, as shown in the figure. At the accuracy point the two strategies are close in performance, but elsewhere in ROC space the instance scoring strategy exhibits a substantial predictive advantage. This can be seen in the "bowing out" of the scoring curve, which has greater area than the discrete classification curve. This means that as conditions change away from the accuracy point, *e.g.*, the class distribution becomes skewed or one type of error becomes more costly, instance scoring will have a definite advantage.

Table 4 compares these AUCs across the UCI domains. "AUC using consultr" is the AUC from discrete classification and "AUC using WVOTE" is the AUC using instance scoring with WVOTE. On nearly every dataset the WVOTE AUC exceeds the corresponding AUC that would result from direct classification. The results pass both the Sign and Wilcoxon tests at $p < .05$. This demonstrates that using rules to score instances results in a measurable predictive benefit over what would be realized from interpreting them as direct classification rules.

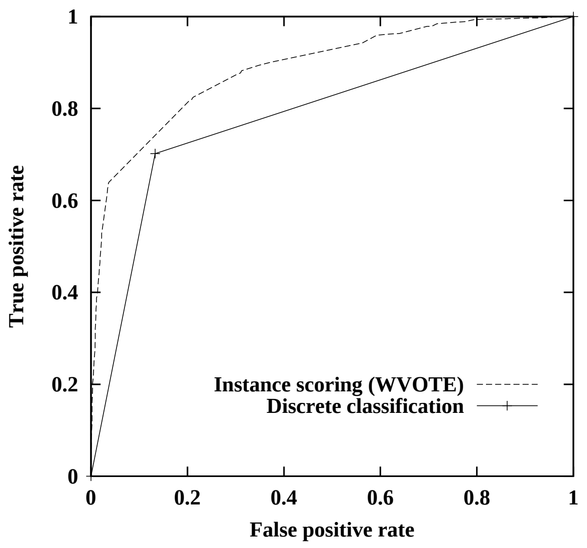

Figure 2. ROC curves of the Covtype domain

Dataset	AUC using `consultr`	AUC using WVOTE
Breast-wisc	95.6 ± 3.4	97.3 ± 3.6
Car	95.3 ± 2.2	98.3 ± 0.7
Cmc	65.1 ± 2.7	66.4 ± 4.9
Covtype	76.0 ± 2.7	82.2 ± 1.4
Crx	85.6 ± 4.8	90.2 ± 3.4
German	64.4 ± 3.4	68.4 ± 9.9
Glass	74.4 ± 7.5	75.5 ± 6.0
Image	97.7 ± 0.7	99.0 ± 0.5
Kr-vs-kp	99.6 ± 0.4	99.7 ± 0.2
Mushroom	100.0 ± 0.0	100.0 ± 0.0
Nursery	98.8 ± 0.3	$99.8 \pm .1$
Promoters	90.1 ± 11.0	88.9 ± 13
Sonar	73.9 ± 6.8	76.9 ± 14
Splice	94.8 ± 1.3	97.2 ± 0.7

Table 4. Comparisons of AUCs under direct classification (“AUC using consultr”) and under instance scoring (“AUC using WVOTE”).

4.5. Comparison with Naive Bayes

Finally, how well do rules perform against other probability estimation techniques? Rules have various desirable qualities such as modularity and intelligibility, but these qualities are shared by other model classes as well, such as Naive Bayes and linear threshold units. How do rules compare?

Table 5 shows ROC performance of Naive Bayes[1] and rules from C4.5rules using WVOTE resolution. In general, rules appear to perform better than simple Naive Bayes. This observation passes a Sign test ($p < 0.05$), though it does not pass the Wilcoxon test at an acceptable level.

It is possible that more complex model classes such as decision tree ensembles [8, 21] or neural networks [14] would produce better probability estimates than rules do. However, these model classes are more complex and expensive, and lack some of the appealing characteristics of classification rules.

5. Discussion and Future Work

This paper has demonstrated that rules, commonly used for direct classification, can also be used effectively for probability estimation. In fact, when used this way their predictive performance increases. In general, they are competitive with a simple probability estimation method such as Naive Bayes. It is likely that further experimentation with the rule generation methods would result in better absolute performance from rules.

[1]Because Naive Bayes is representationally equivalent to a linear threshold unit, no separate test was done.

Several important issues have been left unaddressed in this paper.

Even when rule generation is efficient and effective, classification performance can benefit from rule selection, and this is an area of ongoing work. Wilkins and Ma [19] proved that optimal rule selection is NP-hard because overlapping rules can have “sociopathic” interactions; therefore, practical rule selection techniques must be heuristic. The field of machine learning would benefit from a systematic study of heuristic rule selection methods.

The experiments in this paper employed two standard rule learning methods, C4.5rules and RL, but neither was designed to maximize AUC performance. An open question is how rule *generation* should be altered to produce rule with good AUC performance. Various ideas have been proposed, such as starting with a high confidence bias and dynamically adjusting it based on feedback from AUC performance. To our knowledge, no such research has been seriously pursued.

Finally, a related issue is how best to learn rules when one or more classes is rare. Induction under skewed distributions is an important open issue in machine learning that has received attention recently. The datasets used in this study were chosen to be fairly balanced in order to sidestep this issue so that off-the-shelf rule induction methods could be used. Research on other model classes with skewed data sets should provide valuable insights on rule induction as well.

Dataset	AUC (Naive Bayes)	AUC (C4.5rules)
Breast-wisc	93.1 ± 5.5	97.3 ± 3.6
Car	92.3 ± 2.2	98.3 ± 0.7
Cmc	64.1 ± 5.8	66.4 ± 4.9
Covtype	81.5 ± 2.4	82.2 ± 1.4
Crx	87.6 ± 4.3	90.2 ± 3.4
German	77.1 ± 4.5	68.4 ± 9.9
Glass	74.0 ± 8.7	75.5 ± 6.0
Image	95.6 ± 0.9	99.0 ± 0.5
Kr-vs-kp	95.1 ± 0.8	99.7 ± 0.2
Mushroom	99.8 ± 0.1	100.0 ± 0.0
Nursery	98.0 ± 0.2	$99.8 \pm .1$
Promoters	97.7 ± 4.0	88.9 ± 13
Sonar	76.1 ± 13.0	76.9 ± 14.3
Splice	99.2 ± 0.6	$97.2 \pm .7$

Table 5. Probability estimation: Naive Bayes versus rules from C4.5rules using WVOTE

6. Acknowledgments

Discussions with Foster Provost were helpful in clarifying some rule learning issues. Two anonymous reviewers provided valuable comments. I thank Ross Quinlan for making C4.5 and C4.5rules available for non-commercial use; I thank Foster Provost for making the RL program available; and I thank the Weka project [20] for making their software available.

Much open source software was used in this work. I wish to thank the authors and maintainers of XEmacs, TEX, LATEX, Perl and its many user-contributed packages, and the Free Software Foundation's GNU Project. Scripts for the Sign and Wilcoxon tests were written by Rob van Son.

References

[1] C. Blake and C. Merz. UCI repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[2] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[3] S. Clearwater and F. Provost. RL4: A tool for knowledge-based induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, pages 24–30. IEEE CS Press, 1990.

[4] J. P. Egan. *Signal Detection Theory and ROC Analysis*. Series in Cognitition and Perception. Academic Press, New York, 1975.

[5] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In Chaudhuri and Madigan, editors, *Proceedings on the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53–62, San Diego, CA, Aug. 1999.

[6] A. Freitas. Understanding the crucial differences between classification and discovery of association rules – a position paper. *KDD Explorations*, 2(1):65–69, June 2000.

[7] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

[8] F. Provost and P. Domingos. Well-trained PETs: Improving probability estimation trees. CeDER Working Paper #IS-00-04, Stern School of Business, New York University, NY, NY 10012, 2001.

[9] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pages 43–48, Menlo Park, CA, 1997. AAAI Press.

[10] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, Mar. 2001.

[11] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453, San Francisco, CA, 1998. Morgan Kaufmann.

[12] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.

[13] R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[14] S. Santini and D. A. Bimbo. Recurrent neural networks can be trained to be maximum a posteriori probability classifiers. *Neural Networks*, 8(1):25–29, 1995.

[15] A. Srinivasan. Note on the location of optimal classifiers in n-dimensional ROC space. Technical Report PRG-TR-2-99, Oxford University Computing Laboratory, Oxford, England, 1999.

[16] J. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240:1285–1293, 1988.

[17] J. A. Swets, R. M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, 283:82–87, October 2000.

[18] G. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:383–417, 1995.

[19] D. C. Wilkins and Y. Ma. The refinement of probabilistic rule sets: sociopathic interactions. *Artificial Intelligence*, 70:1–32, 1994.

[20] I. Witten and E. Frank. *Data mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, San Francisco, 2000. Software available from `http://www.cs.waikato.ac.nz/~ml/weka/`.

[21] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of KDD-2001*, pages 204–213, Aug. 2001.