Data Min Knowl Disc (2008) 17:207–224
DOI 10.1007/s10618-008-0089-y

PRIE: a system for generating rulelists to maximize ROC performance

Tom Fawcett

Received: 21 December 2006 / Accepted: 16 January 2008 / Published online: 5 February 2008
Springer Science+Business Media, LLC 2008

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 classifier can perform poorly. This paper presents a method for learning rules directly from ROC space when the goal is to maximize the area under the ROC curve (AUC). Basic principles from rule learning and computational geometry are used to focus the search for promising rule combinations. The result is a system that can learn intelligible rulelists with good ROC performance.

Keywords Classification · ROC analysis · Rule learning · Cost-sensitive learning

1 Introduction

One concern of utility-based data mining is the ability to make cost-effective classification decisions. In order to make such decisions, a classifier should be able to produce, from a new instance, not just a hard classification (*i.e.* a class name) but also an instance score or probability that the instance belongs to the class. Given such an estimate, a classifier can use information about error costs to determine the most cost-effective classification. Previous work has shown that methods that can produce instances scores can be used to make cost-sensitive decisions, either directly by

Responsible editor: Bianca Zadrozny.

T. Fawcett (✉)
Center for the Study of Language and Information, Stanford University, Stanford, CA 94305, USA
e-mail: tfawcett@acm.org

converting them into proper probabilities using a calibration technique (Zadrozny and Elkan 2001; Niculescu-Mizil and Caruana 2005), or indirectly by using a technique such as the ROC convex hull (Provost and Fawcett 1998, 2001).

This goal calls for classification methods that are good at producing accurate instance scores; in other words, methods that maximize ROC performance. In the data mining community, researchers in classification are starting to look at the area under the ROC curve (AUC) instead of accuracy as an evaluation measure, and designing methods to maximize AUC. Such work has been pursued in data mining, but most of it has involved methods whose concepts are difficult to interpret, such as ensembles of classifiers, support vectors machines, etc. Probabilistic classification has been investigated with other model classes such as neural networks (Santini and Bimbo 1995) and ensembles of decision trees (Provost and Domingos 2001; Zadrozny and Elkan 2001). These models tend to be much more complex than rule sets. They attain good ROC performance but they do not have rules' appealing properties of modularity and intelligibility.[1]

Rules are commonly used in data mining because of several desirable properties: they are simple, intuitive, modular, and straightforward to generate from data. But 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. Others have pointed out that accuracy is a poor metric to optimize (Provost et al. 1998; Ling et al. 2003), so accuracy-maximizing methods may be a poor choice for producing scoring classifiers—that is, classifiers with good ROC performance.

Therefore, an open question in data mining is how to generate rules to produce reliable instance scores. Previous work (Fawcett 2001) showed how rules could be combined in various ways to maximize ROC performance, but the rules used in that work were generated by techniques that were not intended to maximize such performance. Most techniques attempt to maximize accuracy, and techniques that maximize accuracy do not always perform well in ROC space (Provost et al. 1998).

This paper may be seen as an extension to that prior work. We introduce a rule learning system called PRIE which is designed to maximize ROC performance. PRIE exploits the structure of ROC space to guide the generation of new rules. PRIE has several attractive features:

1. Because PRIE maximizes ROC performance, it naturally handles skewed datasets.
2. PRIE is able to handle multiple classes. It will attempt to optimize the combined AUC for any number of classes simultaneously.
3. PRIE's output is a single rulelist and thus is relatively intelligible and modular. To use this rulelist on a new unseen instance, the rules are evaluated sequentially and the first one matching determines the class and probability. Some data mining practitioners consider rulelists to be more intelligible than rulesets because only a single rule matches a new instance.

[1] *Modularity* means that each rule is local and a decision on an instance is made by only a single rule. Ensembles lack modularity because multiple classifiers may participate in deciding an instance's class, and their votes may combine in unintuitive ways. *Intelligibility* means that the classifier is reasonably understandable to the end users of the classification system.

4. Because PRIE uses a rulelist whose rules are ordered decreasing by class likelihood, the rulelist may be used naturally with the ROC convex hull (Provost and Fawcett 2001). In use, if operating conditions (class skew and relative error costs) are known, the rulelist can be truncated to eliminate rules that will never affect a classification decision.
5. PRIE handles numerical attributes naturally, using the ROC curve implicitly to identify promising discretizations. Other classification models may discretize variables in a preprocessing pass or may use techniques unrelated to model construction. PRIE considers every discretization of a continuous attribute to comprise a separate point in ROC space, and handles these the same as any other discrete attribute.
6. PRIE can handle set-valued attributes (Cohen 1996), in which an attribute of an instance may take on a set of discrete values instead of a single one. Such features are useful, for example, in text classification domains in which the set may represent the "bag of words" of a text document.

PRIE is unusual in that it uses basic principles from rule learning and computational geometry to focus the search for promising rule combinations. The result is a system that can learn rulelists with high AUC scores.

The remainder of the paper is organized as follows. Section 2 reviews the basics of ROC graphs and the ROC convex hull. These principles and equations form the basis of PRIE's operation. Section 3 describes PRIE and the principles behind its operation. Section 4 describes the sets of experiments, and their results, intended to validate PRIE's approach. The final section concludes and describes areas for future work.

2 ROC graphs

Prior to explaining the PRIE system we briefly review the theory of Receiver Operating Characteristics (ROC) graphs upon which PRIE is based. ROC graphs have long been used in signal detection theory to depict the trade-off between hit rates and false alarm rates of classifiers (Egan 1975; Swets et al. 2000). ROC analysis has been extended for use in visualizing and analyzing the behavior of diagnostic systems (Swets 1988).

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:

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

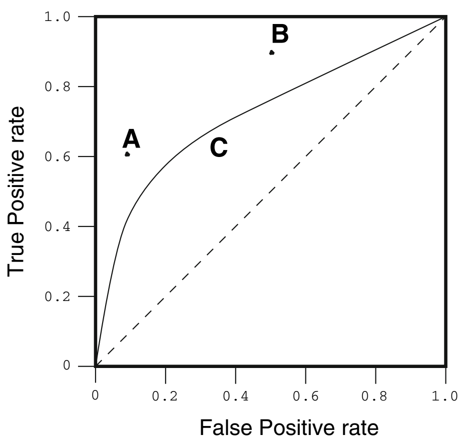
The **False Positive rate** (also called false alarm rate) of the classifier is:

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

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

A "hard" 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 Fig. 1 are hard classifiers.

Fig. 1 An ROC graph



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.

A *ranking* or *scoring* classifier may be thresholded 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 trade-offs available with a given classifier. Figure 1 shows the curve of a ranking classifier, C, in ROC space. Much further information on ROC graphs and ROC analysis is available elsewhere, such as Fawcett's (2006) overview article and tutorial notes by Peter Flach (2004).

2.1 Area under the ROC curve

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 (Bradley 1997; Hanley and McNeil 1982). 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.

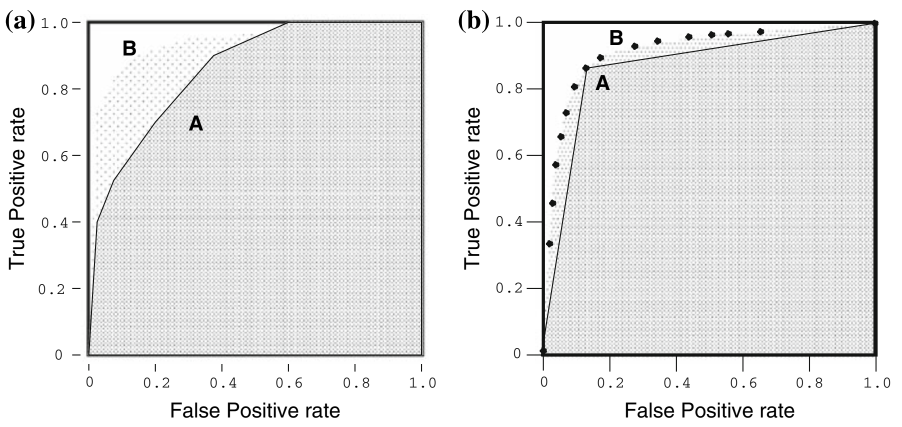


Fig. 2 The area under ROC curves (AUC)

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 (Hanley and McNeil 1982). Figure 2a shows the areas under two ROC curves, A and B. B has greater area and therefore better average performance, though A performs better than B in the upper right region of ROC space.

ROC analysis is commonly used for two classes, but it has been extended to multiple classes (Srinivasan 1999). Unfortunately, visualizing the result for more than two classes is non-intuitive. 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 value.

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:

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

2.2 The ROC convex hull

Provost and Fawcett (1998, 2001) have shown some important properties of the convex hull of a set of points in ROC space. The details are beyond the scope of this article, but a classifier is potentially optimal if and only if it lies on the convex hull of the set of points in ROC space. We call the convex hull of the set of points in ROC space the *ROC convex hull* (ROCCH) of the corresponding set of classifiers.

This ROCCH formulation has a number of useful implications. In use, since only the classifiers whose operating points are on the convex hull are potentially optimal, no other classifiers need to be retained. Furthermore, in rule learning, the convex hull can be used to efficiently keep track of the best individual rules yet found. This may in turn be used to guide generation of new rules, as Sect. 3.2 explains.

2.3 Rulelists and the ROCCH

PRIE constructs a rulelist: an ordered list of rules which, in use, will be tested one at a time against a new instance. The rules are tested in order until one of them matches or until the rulelist is exhausted. The score then emitted for the instance will be based on the rule's statistics.

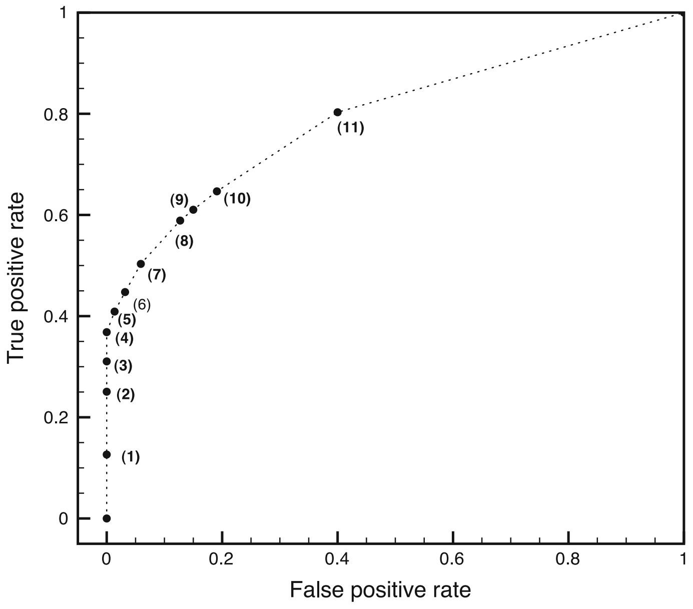
Each rule is a conjunction of conditions, the satisfaction of which implies membership in a class. For simplicity, consider a two-class problem with classes **p** and **n**. An example of a simple rule and some of its performance statistics is:

$$\begin{gathered}\mathbf{x_1} \wedge \mathbf{x_2} \wedge \mathbf{x_3} \longrightarrow \mathbf{p}\\ \text{TP} = 15,\ \text{P} = 100,\ \text{TPrate} = .15\\ \text{FP} = 2,\ \text{N} = 200,\ \text{FPrate} = .01\end{gathered}$$

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.

Rulelists have a natural correspondence to ROC points and to the ROCCH. Rules on a rulelist are naturally ordered by likelihood. Each rule in a rulelist corresponds to a point on an ROC curve. The probability that a given rule matches an instance corresponds to the conditional probability that the instance satisfies its conditions, given that it fails to satisfy all preceding rules on the rulelist. Figure 3 illustrates this relationship. At the bottom is a list of 11 rules learned from the UCI "car" domain with two classes, `acc` (positive class) and `unacc` (negative class). Each rule has two lines: its number and the text (antecedents and consequent) of the rule on the first line, and its local statistics on the second. The local statistics describe how many positive (P) and negative (N) examples matched against it in the training set, and from those how many were correct (TP) and incorrect (FP).

At the top of Fig. 3 is an ROC graph illustrating the rulelist's performance. Each point is labeled by the rule to which it corresponds. More precisely, each point corresponds to a *prefix* of the rulelist. For example, the point labeled 7 on the ROC graph represents the performance that would be derived if only the first seven rules were tested on an instance. It lies at (.06,0.5) in ROC space. The TP rate can be determined by adding the TPs of rules 1–7 (235) and dividing by the number of positives (P = 467). The FP rate can be determined by adding the FPs of rules 1–7 (13) and dividing by the number of negatives (N = 220). If an example were to be classified by this rulelist, and it failed to match rules 1–6 but matched rule 7, the example's score would be given as $235/(235 + 13) \approx 0.95$. If an example failed to match any rule in the rulelist, it would be given the score of $467/(220 + 467) \approx 0.68$, which is the prior probability of the `acc` class.



N	Conditions
1	(maint = med) AND (lug boot = big) → acc TP: 59 FP: 0 P: 467 N: 220
2	(maint = low) AND (lug boot = big) → acc TP: 58 FP: 0 P: 408 N: 220
3	(safety = high) AND (maint = low) AND (lug boot = med) → acc TP: 28 FP: 0 P: 350 N: 220
4	(maint = med) AND (lug boot = med) AND (safety = high) → acc TP: 27 FP: 0 P: 322 N: 220
5	(maint = low) AND (doors = 5more) → acc TP: 19 FP: 3 P: 295 N: 220
6	(maint = low) AND (doors = 4) → acc TP: 18 FP: 4 P: 276 N: 217
7	(maint = med) AND (lug boot = med) → acc TP: 26 FP: 6 P: 258 N: 213
8	(safety = high) AND (doors = 3) → acc TP: 40 FP: 15 P: 232 N: 207
9	(safety = high) AND (lug boot = big) AND (doors = 5more) → acc TP: 10 FP: 5 P: 192 N: 192
10	(safety = high) AND (lug boot = big) → acc TP: 17 FP: 9 P: 182 N: 187
11	(safety = high) → acc TP: 73 FP: 46 P: 165 N: 178

Fig. 3 A rulelist (bottom) for the “car” domain and the corresponding ROC curve (top) with respect to the acc class

This rulelist may thus be seen as a single classifier composed of 11 pieces. It can be “thresholded” by cutting off evaluation at any of these 11 pieces, and thus yields an ROC curve with 11 points, plus the two endpoints at (0,0) and (1,1). PRIE’s goal is to construct such a rulelist with the best possible ROC performance. The way it goes about generating rules and assembling a rulelist from them is described next.

3 PRIE

PRIE is a separate-and-conquer rule learner (Fürnkranz 1999; Fürnkranz and Flach 2005). It may be thought of as comprising an outer loop and an inner loop. The outer loop iteratively chooses the best existing rule and adds it to the end of the rulelist. As rules are added to the rulelist, the instances they match are removed from consideration (“covered”). The inner loop develops new rules for the outer loop to extract.

What makes PRIE different is that the outer loop covers examples based on ROC performance, and the inner loop uses ROC performance to suggest new promising rules.

Rule generation is accomplished by two interleaved processes. One process calculates the convex hull in ROC space, then repeatedly tries to generate rules that independently extend the hull. The outer process extracts the left-most rule of the convex hull, inserts the rule at the end of a rulelist, then removes those instances matched by the rule, and re-invokes the first process. The entire rule generation method may be seen as a process that "shrinks" ROC space iteratively then attempts to find the best (most conservative) individual rule in that space.

3.1 Rule selection and instance covering (outer loop)

The inner loop is discussed in more detail below. The outer loop comprises the higher level actions of the rulelist generation process. PRIE maintains a global rulelist, initially empty, and adds successive rules to the end of it. Algorithm 1 is a basic description of the outer loop process.

PRIE maintains one ROC space for each class under consideration. Each space is a *class reference* ROC (Fawcett 2006), *i.e.* if C is the set of classes, for every class $c_i \in C$ there is an ROC space ROC$_i$ such that the positive class $\mathbf{p} = c_i$ and the negative class $\mathbf{n} = \{c_j \in C | j \neq i\}$. Each of these ROC spaces is initialized to be simply a line connecting (0,0) and (1,1).

PRIE then generates singleton rules for attributes. For each discrete-valued attribute a_j with a value $v_{j,k}$ it creates a rule:

$$(a_j = v_{j,k}) \rightarrow c_i$$

For each set-valued attribute a_j with a value $v_{j,k}$, PRIE creates a rule:

$$(v_{j,k} \in a_j) \rightarrow c_i$$

For each continuous-valued attribute a_j with a value $v_{j,k}$, PRIE creates two rules:

$$(a_j < v_{j,k}) \rightarrow c_i$$
$$(a_j \geq v_{j,k}) \rightarrow c_i$$

For all these singleton rules, PRIE calculates the rule statistics (TP and FP and their corresponding rates), then calculates their position in ROC$_i$ space and calculates the convex hull.[2]

PRIE then enters its outer loop. Given each of the ROC$_i$ spaces, it attempts to develop each space as described in Sect. 3.2. Once this is done, it extracts the single highest likelihood rule from all of the convex hull of each class. The highest likelihood rule corresponds to the endpoint of the leftmost segment of the hull. If there

[2] As an implementation note, PRIE keeps bit vectors with all rules, so calculating and updating rule statistics is very fast. Calculating and updating convex hulls is also efficient.

Algorithm 1 Rule generation algorithm

Given: C: Set of classes; I: Set of instances, each containing an instance vector and class assignment
Output: RL: rulelist

```
1: $RL \leftarrow \langle\rangle$
2: for $c \in$ Classes do
3:   Hull$_c \leftarrow \langle(0,0),(1,1)\rangle$
4: end for
5: Create initial rules for discrete and set-valued attributes
6: Create initial rules for continuous attributes
7: repeat
8:   for $c \in$ Classes do
9:     Develop_rules_for(ROC$_c$)
10:  end for
11:  $R \leftarrow$ Extract_best_rule
12:  Add $R$ to end of $RL$
13:  Remove $R$ from all ROC$_c$ spaces
14:  Recompute each ROC convex hull Hull$_c$
15: until all ROC convex hulls are diagonal lines from (0,0) to (1,1)
16: Output $RL$
```

are multiple vertical segments, the one with the highest TP rate is chosen. In Fig. 3, rule 1 would be considered the highest likelihood rule. This rule is added to the end of the global rulelist, and all of the ROC$_i$ spaces are adjusted accordingly. Finally, the convex hulls are recomputed.

Conceptually, this iterative extraction of rules may be thought of as excising successive lower left (L-shaped) portions of ROC space. Fürnkranz and Flach (2005) discuss separate-and-cover rule-learning algorithms and how the successive removal of examples implicitly shifts to successively nested PN spaces.[3] This is exactly what PRIE is doing explicitly as it extracts rules for its rulelist. Extracting a rule affects all the ROC spaces so they must be adjusted.

This process continues until all the ROC spaces are exhausted, *i.e.* they are reduced to a single line connecting (0,0) to (1,1). At this point, the global rulelist is output and PRIE terminates.

3.2 Constraining rule generation

PRIE's inner loop is responsible for generating new rules. It does this by combining existing rules, by conjoining their conditions. It operates independently over each of the ROC spaces it maintains.

One key issue for any separate-and-cover rule learning algorithm is how to generate promising new rules. Considering every possible combination of features is NP-complete so rule learning systems must use heuristics to guide their search. PRIE attempts to determine, for each of its ROC spaces, which combinations of existing rules are

[3] PN space is simply an unscaled version of ROC space in which the axes are labeled by the number of examples rather than the example proportions. For discussing evaluation metrics, PN space is often more natural than ROC space, though most of the properties of PN space also hold for ROC space.

likely to extend the convex hull. If the new rule would be unlikely to extend the ROC convex hull, it can be eliminated from consideration. In fact, geometric properties of ROC space can be used to eliminate large numbers of rules from consideration. Before explaining how PRIE's inner loop works in Sect. 3.3, we describe the principles underlying its process for deciding which rules to combine.

If we conjoin the conditions of two rules α and β to create a new rule γ, where will the new rule lie in ROC space? The answer depends on the intersections among the TP and FP sets of α and β; that is, upon their attribute interactions.

Let tpr_γ be the expected true positive rate of γ and let x be an instance in the true positive set of γ. Then:

$$\begin{aligned}\text{tpr}_\gamma &\approx p(x \in \text{TP}_\gamma)\\ &\approx p(x \in \text{TP}_\alpha \wedge x \in \text{TP}_\beta)\end{aligned}$$

A useful simplification is to assume that the rules are conditionally independent, so the probability of an instance matching one rule is independent of the probability of it matching the other. If we assume independence of α and β,

$$\begin{aligned}\text{tpr}_\gamma &\approx p(x \in \text{TP}_\alpha) \cdot p(x \in \text{TP}_\beta)\\ &\approx \frac{\mid \text{TP}_\alpha \mid}{\mid P \mid} \cdot \frac{\mid \text{TP}_\beta \mid}{\mid P \mid}\\ &\approx \text{tpr}_\alpha \cdot \text{tpr}_\beta\end{aligned}$$

A similar derivation can be done for the expected false positive rate fpr_γ. Thus, the conjunction of two rules α and β can be expected to lie in ROC space at:

$$\gamma = \left(\text{fpr}_\alpha \cdot \text{fpr}_\beta,\ \text{tpr}_\alpha \cdot \text{tpr}_\beta\right) \tag{1}$$

Figure 4 shows an ROC graph of two rules, α at (0.6, 0.85) and β at (0.35, 0.70). Rule γ is plotted at (0.21, 0.595) as calculated by Eq. 1.

Equation 1 may be used as a heuristic to constrain rule generation. Given two existing rules α and β, we can use this equation to estimate whether combining them would produce a new rule whose ROC position lies beyond the ROC convex hull (and so would extend it). If it would, we can proceed with the (presumably much more expensive) process of combining the rules performing instance matching to determine the actual performance of the new rule.

In fact, a further optimization is possible. It is not strictly necessary to examine all pairs of rules and test them with (1). For any given rule **a**, we can quickly eliminate from consideration many rules that would not likely extend the hull when combined with **a**.

Refer to Fig. 5. The ROC convex hull is made up of line segments connecting pairs of rules. One such hull segment is shown in the figure. Consider some existing rule **a**, shown in the figure. It is not necessary to test every other rule z using (1) to determine whether $a \wedge z$ would lie beyond this hull segment. We can do this by plugging **a**'s position into (1).

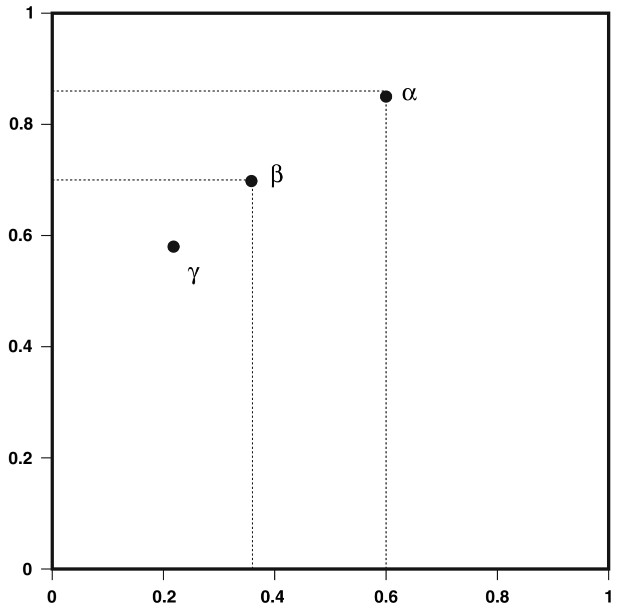


Fig. 4 Estimating the performance of a new rule. α and β are two rules conjoined to form γ. The coordinate of γ is the expected location of the new rule, assuming independence

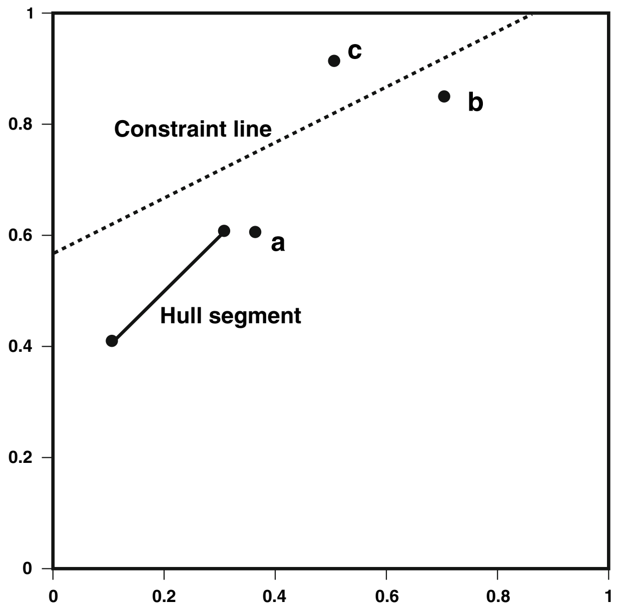


Fig. 5 A constraint line used to prune rule conjunction candidates

Given a rule α and a hull segment $h_i \in H$, we can create a corresponding boundary line w_i. If a second rule β does not lie above w_i, then $\alpha \wedge \beta$ will not lie above the hull line h_i, and the combination $\alpha \wedge \beta$ need not be considered. Given a hull line segment h_i expressed in the line equation $y = m_{hull} \cdot x + b_{hull}$, points lying above this line must satisfy the inequality:

$$y > m_{hull} \cdot x + b_{hull}$$

Let rule α have true positive rate tp_α and false positive rate fp_α. By (1), for a second rule r_β to produce a rule above this hull line when conjoined with α, it must satisfy the inequality:

$$y \cdot tp_\alpha > m_{hull} \cdot x \cdot \mathrm{fp}_\alpha + b_{hull} \tag{2}$$

$$\mathrm{tp}_\beta \cdot \mathrm{tp}_\alpha > m_{hull} \cdot \mathrm{fp}_\beta \cdot \mathrm{fp}_\alpha + b_{hull} \tag{3}$$

$$\mathrm{tp}_\beta > \frac{\mathrm{fp}_\alpha \cdot m_{hull}}{\mathrm{tp}_\alpha} \cdot \mathrm{fp}_\beta + \frac{b_{hull}}{\mathrm{tp}_\alpha} \tag{4}$$

This inequality yields a “constraint line” below which no rule should be considered for combining with α to extend segment h_i.

Figure 5 shows an example of this. Assume there is a hull segment from (.1, .4) to (.3,.6), shown as a solid line in the figure. We have a rule labeled **a** at (.35, .6), and we want to determine which rules, when combined **a**, will likely produce a new rule that lies beyond (the left of) this hull segment. Using the inequality in (4) we can derive the equation of a line:

$$y = 0.58 \cdot x + 0.5$$

which forms the dashed constraint line shown in Fig. 5. When considering rules to combine with **a**, we can immediately remove from consideration any rule lying to the right of the constraint line. Thus, rule **c** is a promising choice for conjoining with **a**, but **b** is not.

3.3 Rule generation (inner loop)

PRIE’s inner loop is responsible for generating new rules by conjoining the conditions of existing rules. PRIE operates independently over each of the ROC spaces it maintains, using the techniques of the last sections to develop rules for each space.

Each hull segment is examined in turn and rule pairs are considered whose conjunction might extend the hull; that is, whose conjunction is predicted to lie to the northwest of the segment, in the portion of ROC space as yet uncovered. PRIE uses the principles in Sect. 3.2 to constrain its search for combinations.

Prior to generating the new rule, the antecedents of the constituent rules are checked for tests that might render the conjunction useless. For example, if one rule contains a test $(a_i = v)$ and the other contains a test $(a_i = w)$ for two distinct values v and w, the two would not be combined. Another such test is for conditions like $a_i < v$ and

$a_i > v$ being combined in the conjoined rule. In addition, duplicate tests within the antecedents are removed.

4 Empirical validation

In this section, we compare PRIE's performance against various other induction methods. The domains for evaluation are described and summarized in Table 1. These results are based on 10-fold cross-validation, and all values are given as means and standard deviations.

4.1 PRIE versus randomness

As a baseline, we compare PRIE's rule generation method against a method that evaluates rules randomly when considering whether they should be combined. Specifically, when considering whether a new rule conjunction is likely to extend the ROC hull, this random method generates a random number between -1 and 1 as an estimate of how far to the left of a hull segment the new rule will be. We would expect such random estimates not to perform very well, and in fact the results shown in Table 2 shows they do not. A Wilcoxon Matched-pair Signed-Ranks Test confirms this.

4.2 PRIE versus error minimization methods

Next we compare PRIE against error minimization methods. Previous work (Fawcett 2001) showed that rules used for scoring performed better on maximizing the AUC than the same rules with just their hard classifications. This is not a surprising result, since hard classifications produce a single ROC point, whereas scores can produce a set of points (a curve) in ROC space. Figure 2b illustrates this. However, it remains to be seen whether a system like PRIE, designed to maximize AUC performance, can perform better than error minimizing rules interpreted probabilistically.

Previous work (Fawcett 2001) evaluated the performance of rule sets for maximizing the AUC, but there are two significant differences with that work. The goal of that work was to use standard error minimizing (*i.e.* accuracy maximizing) rule generation techniques to see how they could best be used to maximize AUC. In addition, that work experimented with a variety of rule resolution techniques—methods for combining the responses of multiple matching rules—to determine which was most effective in maximizing AUC. That paper concluded that a weighted voting scheme (called WVOTE) performed best with the error-minimizing rules.

In this section we compare the best results from that paper with PRIE's results. Table 3 shows the AUC results of PRIE and the AUC results of C4.5rules using the WVOTE protocol. PRIE's performance is comparable to the weighted voting of accuracy-maximizing rulesets.

4.3 PRIE and other rule-learning methods

ROCCER (Prati and Flach 2005) is an AUC-maximizing rule learning method. As such, it is a competitor with PRIE. ROCCER uses an association rule learning method

Table 1 Datasets and their characteristics

Name	Instances	Classes		Attributes	
		N	Proportions	N	Type
Breast	683	2	65/34	9	0+9
Breast-wisc	683	2	65/34	9	0+9
Breast-wpbc	198	2	76/23	32	0+32
Bupa	345	2	57/42	34	0+34
Car	1728	2	70/29	34	6+28
Cmc	1473	3	42/34/22	34	7+27
Covtype	5000	7	24/18/18/12/11/9/3	34	4+30
Crx	653	2	54/45	34	9+25
Dermatology-BB[a]	366	2	69/30	34	33+1
Ecoli	336	2	89/10	34	26+8
Flag	194	8	35/24/20/8/4/4/1/0	33	32+1
Flag-Prati	194	2	91/8	33	22+11
German	1000	2	70/30	33	23+10
Glass	214	2	92/7	32	16+16
Glass-RS	205	5	37/34/14/8/6	32	16+16
Haberman	306	2	73/26	33	16+17
Heart	270	2	55/44	33	15+18
Image	2310	7	14/14/14/14/14/14/14	33	11+22
Ionosphere	351	2	64/35	34	0+34
Kr-vs-kp	3196	2	52/47	36	36+0
Letter-a	20000	2	96/3	36	20+16
Mushroom	5644	2	61/38	36	36+0
New-thyroid	215	2	86/13	36	31+5
Nursery	12960	2	97/2	36	36+0
Optdigits	5620	10	10/10/10/10/10/9/9/9/9/9	64	0+64
Page-blocks	5473	5	89/6/2/1/0	64	0+64
Pima	768	2	65/34	64	0+64
Promoters	106	2	50/50	64	57+7
Sonar	208	2	53/46	64	0+64
Splice	3190	3	51/24/24	64	60+4
Vehicle	846	2	76/23	64	42+22
Yeast	1484	10	31/28/16/10/3/2/2/2/1/0	63	42+21

Attribute type in the form “d+c” is the number of discrete and continuous attributes

[a] This is the UCI dermatology domain as converted by Barakat and Bradley (2006): a two-class domain comprising the original class 1, and class 2 comprising the original classes 2 through 6

to generate rules. It attempts to insert each rule into a rulelist, testing the resulting rulelist’s ROC classification performance. If the new rule improves the performance, it is retained, else it is discarded.

Table 2 PRIE compared to a random rule generation method

Dataset	AUC using random	PRIE
Bupa	68.3 ± 9.1	69.9 ± 7.4
Car	93.5 ± 3.0	94.0 ± 1.7
Covtype	79.6 ± 0.7	87.8 ± 1.3
Haberman	67.0 ± 11.5	67.2 ± 16.3
Promoters	79.2 ± 13.7	83.3 ± 13.9
Sonar	62.9 ± 12.3	65.3 ± 10.9
Splice	83.8 ± 3.3	93.0 ± 1.7
Vehicle	97.2 ± 1.9	98.3 ± 1.3
Yeast	54.9 ± 2.5	56.5 ± 2.1

Table 3 PRIE compared to weighted voting (WVOTE) of accuracy-maximizing rules

Dataset	AUC using WVOTE	PRIE
Breast-wisc	97.3 ± 3.6	98.1 ± 1.8
Car	98.3 ± 0.7	94.0 ± 1.7
Cmc	66.4 ± 4.9	66.1 ± 3.2
Covtype	82.2 ± 1.4	87.8 ± 1.3
Crx	90.2 ± 3.4	91.0 ± 4.4
German	68.4 ± 9.9	74.2 ± 5.0
Glass	75.5 ± 6.0	82.5 ± 12.5
Image	99.0 ± 0.5	99.0 ± 0.4
Kr-vs-kp	99.7 ± 0.2	98.5 ± 0.6
Mushroom	100.0 ± 0.0	100.0 ± 0.0
Nursery	99.8 ± .1	99.0 ± 0.3
Promoters	88.9 ± 13	83.3 ± 13.9
Sonar	76.9 ± 14	65.3 ± 10.9
Splice	97.2 ± 0.7	93.0 ± 1.7

Table 4 shows ROCCER's performance results compared to PRIE's on fifteen domains[4] reported by Prati and Flach (2005). The table shows that PRIE performs on par on these domains with ROCCER. A Wilcoxon Matched-pair Signed-Ranks Test confirmed that there was no statistical difference.

Prati and Flach also reported the number of rules selected by their system, and compared these numbers to the number of rules selected by CN2 (Clark and Niblett 1989; Clark and Boswell 1991), a competing rule learning system that attempts to maximize accuracy. Their results are reported in Table 5 along with PRIE's results from the same domains. It is difficult to draw conclusions from only three domains, but on this sample PRIE's rulelists contain fewer rules than CN2 does.

[4] Satimage is not included because the contributors of that domain specified that it should not be used in cross-validation, upon which these results are based.

Table 4 PRIE compared to ROCCER

Dataset	ROCCER	PRIE
Breast	98.63 ± 1.88	98.1 ± 1.8
Bupa	65.30 ± 7.93	69.9 ± 7.4
E-coli-prati	90.31 ± 11.56	94.1 ± 7.5
Flag	61.83 ± 24.14	68.2 ± 16.1
German	72.08 ± 6.02	74.2 ± 5.0
Glass	79.45 ± 12.98	82.5 ± 12.5
Haberman	66.41 ± 11.54	67.2 ± 16.3
Heart	85.78 ± 8.43	80.3 ± 6.8
Ionosphere	94.18 ± 4.49	93.7 ± 5.4
Kr-vs-kp	99.35 ± 0.36	98.5 ± 0.6
Letter-a	96.08 ± 0.52	99.4 ± 0.4
New-thyroid	98.40 ± 1.70	96.1 ± 7.8
Nursery	97.85 ± 0.44	99.0 ± 0.3
Pima	70.68 ± 5.09	78.9 ± 6.2
Vehicle	96.42 ± 1.47	98.3 ± 1.3

Table 5 Number of rules selected by CN2, ROCCER and PRIE

Dataset	CN2	ROCCER	PRIE
German	139.8 ± 5.9	29.9 ± 2.88	57.6 ± 5.2
Pima	158.5 ± 12.3	11.8 ± 1.32	44 ± 1.7
Sonar	31.6 ± 1.71	62.5 ± 3.59	21.5 ± 1.8

Table 6 PRIE compared to the SVM rule extraction method of Barakat and Bradley

Dataset	SVM	SVM + Rules	PRIE
Pima	82 ± 3	94 ± 2	78.9 ± 6.2
Breast cancer	97 ± 1	96 ± 2	98.1 ± 1.8
Heart	89 ± 4	81 ± 5.1	80.3 ± 6.8
Dermatology	1.00 ± 0	98.4 ± 1.1	99.5 ± 0.5

Values given are AUC and standard deviation

As another comparison, we consider the work of Barakat and Bradley (2006). They point out that, because support vector machines are "black box" classifiers, some work has been done on generating rulesets from SVMs. The goal is to obtain a classifier that retains much of the performance of the SVM with the intelligibility of a rule set. Barakat and Bradley specifically investigated the AUC performance of the SVMs and the rulesets derived from them. Though it requires two steps—generating an SVM from data, then generating a ruleset from the SVM—it achieves the same ends and thus is a competing approach to PRIE. It is worth comparing their results to PRIE's.

Barakat and Bradley (2006) compared their approach on only four domains. Their results are shown in Table 6. After the domain name is the AUC value obtained from the original SVM trained on the data, followed by the AUC obtained from the ruleset generated from the SVM. The fourth column shows the results of PRIE on the domain. These data show that PRIE is comparable to the rules extracted from an SVM on these domains.

5 Conclusions and future work

PRIE is an induction technique that is able to generate rules directly from ROC space when the goal is to maximize the area under the ROC curve. PRIE uses basic principles from rule learning and computational geometry to focus the search for promising rule combinations. The result is a system that can learn intelligible rulelists with good ROC performance. Empirical results show that PRIE's rules compare favorably with those of other similar induction techniques.

As mentioned in Sect. 3.1, PRIE discretizes continuous attributes by generating discrete singleton rules from every cut-point. It then "drops" these rules into ROC space and lets them compete with others to form new rules. Thus PRIE implicitly uses ROC space to determine effective cut-points; the best discretizations will end up on the convex hull. To the best of our knowledge, this is the first time ROC space has been used as a discretization technique in data mining, though it is used here in conjunction with rule learning. An area of future work is to investigate this discretization method independently, against other discretization techniques, to determine the extent to which the attribute discretization is a factor in rule-learning performance.

In terms of system control, PRIE may be seen as complementary to the ROCCER system (Prati and Flach 2005). ROCCER takes rules generated from an association rule learner and examines their ROC performance when placed into a rulelist. Thus ROCCER may be seen as generating from the rule lattice space and filtering in ROC space. On the other hand, PRIE generates rules directly from ROC space, but it must employ post-processing tests (described in the last paragraph of Sect. 3.3) to filter obviously poor combinations. In a sense, these tests are incorporating information about the structure of rule space that an association rule learner employs directly. One interesting area of future work is to attempt a more concerted integration of these spaces, so that both rule space information and ROC performance expectations are used in parallel to maximum effect.

Acknowledgements Ronaldo Prati provided much help with the evaluation domains of ROCCER. Comments by the anonymous reviewers were very useful. 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.

References

Barakat N, Bradley A (2006) Rule extraction from support vector machines: measuring the explanation capability using the area under the ROC curve. In: ICPR 2006. 18th international conference on pattern recognition, vol 2, IEEE Press, pp 812–815

Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159

Clark P, Boswell R (1991) Rule induction with CN2: some recent improvements. In: Kodratoff Y (ed) Machine learning—proceedings of the fifth European conference, pp 151–163

Clark P, Niblett T (1989) The CN2 induction algorithm. Mach Learn 3:261–283

Cohen WW (1996) Learning trees and rules with set-valued features. In: AAAI/IAAI, vol. 1, pp 709–716

Egan JP (1975) Signal detection theory and ROC analysis. Series in cognitition and perception. Academic Press, New York

Fawcett T (2001) Using rule sets to maximize ROC performance. In: Proceedings of the IEEE international conference on data mining (ICDM-2001), pp 131–138

Fawcett T (2006) An introduction to ROC analysis. Pattern Recognit Lett 27(8):882–891

Flach P (2004) The many faces of ROC analysis in machine learning ICML-04 Tutorial. Notes available from http://www.cs.bris.ac.uk/~flach/ICML04tutorial/index.html

Fürnkranz J (1999) Separate-and-conquer rule learning. Artif Intell Rev 13(1):3–54

Fürnkranz J, Flach PA (2005) Roc 'n' rule learning—towards a better understanding of covering algorithms. Mach Learn 58(1):39–77

Hanley JA, McNeil BJ (1982) The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143:29–36

Ling CX, Huang J, Zhang H (2003) Auc: a better measure than accuracy in comparing learning algorithms. In: Advances in artificial intelligence: 16th conference of the canadian society for computational studies of intelligence, Springer, pp 329–341

Niculescu-Mizil A, Caruana R (2005) Predicting good probabilities with supervised learning. In: Raedt LD, Wrobel S (eds) Proceedings of the twenty-second international conference on machine learning (ICML'05), ACM Press, pp 625–632

Prati R, Flach P (2005) Roccer: an algorithm for rule learning based on ROC analysis. In: IJCAI 2005, pp 823–828

Provost F, Domingos P (2001) Well-trained PETs: improving probability estimation trees. CeDER Working Paper #IS-00-04, Stern School of Business, New York University, NY, NY 10012

Provost F, Fawcett T (1998) Robust classification systems for imprecise environments. In: Proceedings of AAAI-98. AAAI Press, Menlo Park, CA, pp 706–713

Provost F, Fawcett T (2001) Robust classification for imprecise environments. Mach Learn 42(3):203–231

Provost F, Fawcett T, Kohavi R (1998) The case against accuracy estimation for comparing induction algorithms. In: Shavlik J (ed) Proceedings of ICML-98. Morgan Kaufmann, San Francisco, CA, pp 445–453. Available: http://www.purl.org/NET/tfawcett/papers/ICML98-final.ps.gz

Santini S, Bimbo DA (1995) Recurrent neural networks can be trained to be maximum a posteriori probability classifiers. Neural Netw 8(1):25–29

Srinivasan A (1999) Note on the location of optimal classifiers in n-dimensional ROC space. Technical Report PRG-TR-2-99, Oxford University Computing Laboratory, Oxford, England. Available: http://citeseer.nj.nec.com/srinivasan99note.html

Swets J (1988) Measuring the accuracy of diagnostic systems. Science 240:1285–1293

Swets JA, Dawes RM, Monahan J (2000) Better decisions through science. Sci Am 283:82–87

Zadrozny B, Elkan C (2001) Obtaining calibrated probability estimates from decision trees and naive bayesian classiers. In: Proceedings of the eighteenth international conference on machine learning, pp 609–616