ROC Graphs with Instance-Varying Costs

Tom Fawcett
Stanford Computational Learning Laboratory
Palo Alto, CA USA
tfawcett@acm.org

Abstract

Receiver Operating Characteristics (ROC) graphs are a useful technique for organizing classifiers and visualizing their performance. ROC graphs have been used in cost-sensitive learning because of the ease with which class skew and error cost information can be applied to them to yield cost-sensitive decisions. However, they have been criticized because of their inability to handle instance-varying costs; that is, domains in which error costs vary from one instance to another. This paper presents and investigates a technique for adapting ROC graphs for use with domains in which misclassification costs vary within the instance population.

Keywords: ROC analysis, cost-sensitive learning, classifier evaluation

1. Introduction

A Receiver Operating Characteristics (ROC) graph is a simple technique for visualizing, organizing and selecting classifiers based on their performance. ROC graphs have long been used in signal detection theory to depict the tradeoff between hit rates and false alarm rates of classifiers [2, 13]. ROC analysis has been extended for use in visualizing and analyzing the behavior of diagnostic systems [12, 4]. The medical decision making community has an extensive literature on the use of ROC graphs for diagnostic testing [17]. Swets, Dawes and Monahan recently brought ROC curves to the attention of the wider public with their *Scientific American* article [13].

Recent years have seen an increase in the use of ROC graphs in the machine learning and pattern recognition communitites. One advantage of ROC graphs is that they enable visualizing and organizing classifier performance without regard to class distributions or error costs [8]. This ability becomes very important when investigating learning with skewed distributions or cost-sensitive learning. A researcher can graph the performance of a set of classifiers, and that graph will remain invariant with respect to the operating conditions (class skew and error costs). As these conditions change, the region of interest of the graph may change, but the graph itself will not. In such cases the researcher calculates, at classification time, the approximate operating conditions under which the set of classifiers will be used, overlays it onto the ROC graph, and uses the information to choose which classifier(s) to use. As conditions change the ROC graph may be reconsulted, but the classifiers need not be re-evaluated. If one classifier is broadly superior, the researcher may decide to discard the others [10].

Unfortunately, such methods have an inherent limitation. ROC graphs plot true positive rate against false positive rate, treating all errors of a given type to be equivalent. In some domains this assumption does not hold: the cost of a particular kind of error is not constant throughout the population but varies by example. A typical such domain is that of credit card fraud detection, in which transactions must be evaluated in real time and judged fraudulent or legitimate. A $1000 transaction, if fraudulent, is much more costly than a fraudulent $10 transaction, and classification error costs for the transactions should be correspondingly different. Such costs have been given a variety of names: example-specific costs, instance-varying costs, and case-conditional error costs [14].

ROC graphs have been criticized because of their inability to handle example specific costs. In this paper we present a straightforward transformation of ROC graphs, called ROCIV graphs, that accomodate example specific costs. We show domains in which such graphs are useful; that is, in which standard ROC graphs may mislead as to classifier superiority. We prove that the area under the ROCIV curve has a natural interpretation related to the area under an ROC

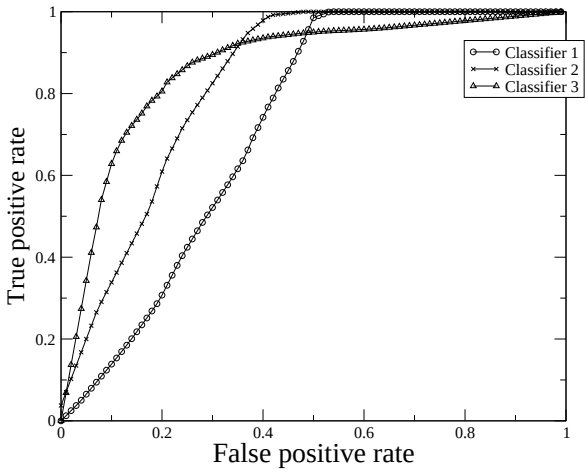


Figure 1. An ROC graph of three scoring classifiers

curve. Finally, we show several domains with instance varying costs, illustrate the ROCIV graph on each, and argue for its superiority over alternative methods for evaluating classifiers on the domain.

2. A Brief Review of ROC Graphs

This section will briefly review ROC graphs. Much more information is available in [4].

Let $\{\mathbf{p}, \mathbf{n}\}$ be the positive and negative instance classes, and let $\{\mathbf{Y}, \mathbf{N}\}$ be the classifications produced by a classifier. Let $p(\mathbf{p}|I)$ be the posterior probability that instance I is positive. The true positive rate, TP, of a classifier is:

$$tp\ rate = p(\mathbf{Y}|\mathbf{p}) \approx \frac{\text{positives correctly classified}}{\text{total positives}}$$

The false positive rate, FP, of a classifier is:

$$fp\ rate = p(\mathbf{Y}|\mathbf{n}) \approx \frac{\text{negatives incorrectly classified}}{\text{total negatives}}$$

We will use the term *ROC space* to denote the classifier performance space used for visualization in ROC analysis. On an ROC graph, *tp rate* is plotted on the Y axis and *fp rate* is plotted on the X axis. These statistics vary together as a threshold on a classifier's continuous output is varied between its extremes, and the resulting curve is called the ROC curve. An ROC curve illustrates the error tradeoffs available with a given classifier. Figure 1 shows a typical ROC plot of three classifiers.

Algorithm 1 Generating ROC points

Inputs: L, the set of test examples; $f(i)$, the probabilistic classifier's estimate that example i is positive; P and N, the number of positive and negative examples.
Outputs: R, a list of ROC points increasing by *fp rate*.
Require: $P > 0$ and $N > 0$

```
L_sorted ← L sorted decreasing by f scores
FP ← TP ← 0
R ← ⟨⟩
f_prev ← −∞
i ← 1
while i ≤ |L_sorted| do
    if f(i) ≠ f_prev then
        push (FP/N, TP/P) onto R
        f_prev ← f(i)
    if L_sorted[i] is a positive example then
        TP ← TP + 1
    else                          /* i is a negative example */
        FP ← FP + 1
    i ← i + 1
push (FP/N, TP/P) onto R          /* This is (1,1) */
end
```

Algorithm 1 is a basic algorithm for generating an ROC graph from a test set. It exploits the monotonicity of thresholded classifications: any instance that is classified as positive with respect to a given threshold will be classified as positive for all lower thresholds. This algorithm assumes that the classifier assigns scores to instances, proportional to the probability that a given instance is positive. The function $f(i)$ is the score assigned to instance i by the classifier. In this algorithm, TP and FP start at zero. For each positive instance we increment TP and for every negative instance we increment FP. We maintain a stack R of ROC points, pushing a new point onto R after each instance is processed. The final output is the stack R, which will contain points on the ROC curve.

2.1. Error costs

Let $c(hyp, class)$ be a two-place error cost function where *hyp* is the hypothesized class assigned to an instance by the classifier and *class* is the instance's real class. $c(\mathbf{Y}, \mathbf{n})$ is the cost of a false positive error and $c(\mathbf{N}, \mathbf{p})$ is the cost of a false negative error. In this paper we shall treat costs and benefits equivalently: a cost is simply a negative benefit. Similarly, it is possible to "roll up" benefits into costs, defining error costs to include benefits not realized.

If a classifier produces posterior probabilities, decision analysis gives us a way to produce cost-sensitive classifications from the classifier [15]. Classifier error frequencies can be used to approximate probabilities [7]. For an instance I, the decision to emit a positive classification is:

$$[1 - p(\mathbf{p}|I)] \cdot c(\mathbf{Y}, \mathbf{n}) \ < \ p(\mathbf{p}|I) \cdot c(\mathbf{N}, \mathbf{p})$$

Regardless of whether a classifier produces probabilistic or binary classifications, its expected cost on a test set can be estimated as:

$$\text{Cost} = FP \cdot c(\mathbf{Y}, \mathbf{n}) + FN \cdot c(\mathbf{N}, \mathbf{p})$$

Given a set of classifiers, a set of examples, and a precise cost function, most work on cost-sensitive classification uses an equation such as this to rank the classifiers according to cost and chooses the minimum. However, as discussed above, such analyses assume that the distributions are precise and static.

An advantage of ROC graphs is that they enable visualizing and organizing classifier performance without regard to class distributions or error costs. A researcher can graph the performance of a set of classifiers, and that graph will remain invariant with respect to the operating conditions (class skew and error costs). As these conditions change, the region of interest may change, but the graph itself will not. Identifying the region of interest is done using *iso-performance lines*.

2.2. Iso-performance lines

Provost and Fawcett [9, 10] show that a set of operating conditions may be transformed easily into a so-called *iso-performance line* in ROC space. Two points in ROC space, (FP_1,TP_1) and (FP_2,TP_2), have the same performance if

$$m = \frac{TP_2 - TP_1}{FP_2 - FP_1} = \frac{c(\mathbf{Y}, \mathbf{n})p(\mathbf{n})}{c(\mathbf{N}, \mathbf{p})p(\mathbf{p})} \qquad (1)$$

This equation defines the slope of an *iso-performance line*. All classifiers corresponding to points on a line of slope m have the same expected cost. Each set of class and cost distributions defines a family of iso-performance lines. Lines "more northwest" (having a larger TP-intercept) are better because they correspond to classifiers with lower expected cost.

The slope m is determined by external constraints: the class skew of the domain, represented by $p(\mathbf{n})/p(\mathbf{p})$, and the relative costs of false positive and false negative errors, represented by $c(\mathbf{Y}, \mathbf{n})/c(\mathbf{N}, \mathbf{p})$. All points (classifiers) along any such line will have equal expected cost. In essence, m represents the trade-off between the false positive error rate and the true positive rate that is acceptable after taking into account the inherent class skew of the domain. For example, if $m = 9$, this represents the condition that a 1% increase in false positive rate is worth a 9% increase in true positive benefit (or, equivalently, a 9% decrease in false negative rate). However, this formulation assumes that all errors of a given type are the same. In other words, that false positive costs are constant within the population of negative examples and all false negative costs are constant within the population of positive examples.

	fraudulent	legitimate
refuse	\$20	$-$\$20
approve	$-x$	$0.05x$

(a)

	fraudulent	legitimate
refuse	0	0
approve	$\$20 + x$	$0.05x + \$20$

(b)

Figure 2. Cost matrices for the credit approval domain. (a) original benefit matrix, (b) transformed cost-benefit matrix

3. Instance-varying Costs

In some domains the cost of a particular kind of error is not constant throughout the population but varies by example. Consider a simple credit card transaction domain used by Elkan [3] in which the task is to decide whether to approve or refuse a given transaction. Elkan describes a benefit matrix for the task, shown in figure 2a. This cost matrix is justified with the following explanation. A refused fraudulent transaction has a benefit of \$20 because it may prevent future fraud. Refusing a legitimate transaction has a negative benefit because it annoys a customer. Approving a fraudulent transaction has a negative benefit proporational to the transaction amount (x). Approving a legitimate transaction generates a small amount of income proportional to the transaction amount ($0.02x$).

In this credit approval example, costs of mistakes are known beforehand because the credit amount is known at the time a classification must be made. In other domains, costs can only be known *after* an action has been taken. An example of this is when soliciting for charitable donations. If a charity decides not to

solicit a prospective donor, the charity will likely never find out whether (or how much) the person would have donated.

To accomodate instance-varying costs, we extend the cost function. Costs may be expressed as three-place functions of the class, the hypothesized class and the instance x: $c(hyp, class, x)$

There are several common ways in which researchers deal with example-specific costs. The most common approach is simply to calculate the total expected cost of a classifier [3]. If error costs are known exactly, the expected cost of a classifier may be calculated as:

$$\sum_{x \in X^+} c(h(x), \mathbf{p}, x) + \sum_{x \in X^-} c(h(x), \mathbf{n}, x)$$

where $h(x)$ is the hypothesized class of instance x. This equation achieves an exact solution but it loses the advantage of ROC curves, which is to allow classifier performance to be visualized and compared over a range of performance conditions. However, this solution may be appropriate if costs and class distributions are known exactly and are known not to change.

Another method is to smooth out the costs: measure the average costs of false positives and false negatives and use these average values. Let X^+ and X^- be the set of positive and negative instances, respectively. Then:

$$c(\mathbf{N}, \mathbf{p}) \approx \sum_{x \in X^+} c(\mathbf{N}, \mathbf{p}, x)/|X^+| \quad (2)$$

$$c(\mathbf{Y}, \mathbf{n}) \approx \sum_{x \in X^-} c(\mathbf{Y}, \mathbf{n}, x)/|X^-| \quad (3)$$

These two-place cost functions may then be used in equation 1 so that iso-performance lines can be used with ROC curves. Previous work on fraud detection [5] took this approach, and it may be acceptable if cost variance is small.

Another approach is to divide costs into ranges and treat them as separate classes. For example, the positive class **p** could be split into three classes based on false negative costs: **p-high**, **p-medium** and **p-low**; and the instances assigned accordingly. The misclassification costs of instances within a bin could be approximated by the average cost. This would result in three separate negative error rates: FN_{high}, FN_{medium} and FN_{low}. A similar differentiation could be done for negative instances, resulting in a total of six classes. This allows finer differentiation among classification decisions whose costs may be different, and allows the classifier to choose a different threshold for each. Unfortunately, this approach has several disadvantages. Such a class differentiation is artificial. For example, medium cost instances may be very similar to high cost instances but this approach forces the classifier to regard them as different. Such partitioning fragments the instance space and reduces the number of training instances available for each class. It also proliferates the number of classes that must be dealt with. ROC analysis is natural for two classes, but for n classes there are $n^2 - n$ possible misclassifications, each of which is a separate dimension that must be dealt with. In short, increasing the number of classes greatly reduces the advantages (and attraction) of ROC anslysis.

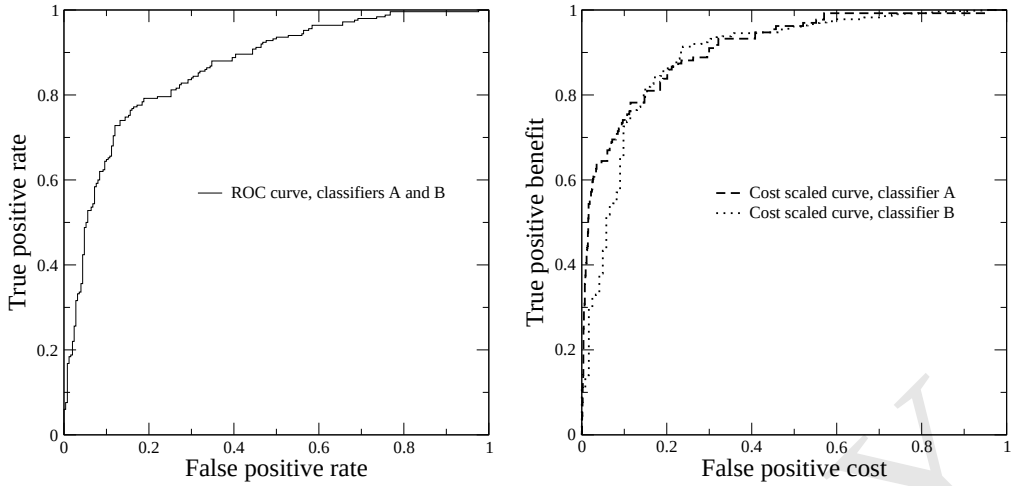
Algorithm 2 Generating ROC points from a dataset with example-specific costs

Inputs: L, the set of test examples; $f(i)$, the probabilistic classifier's estimate that example i is positive; P and N, the number of positive and negative examples; $c(\mathbf{Y}, class, i)$, the cost of judging instance i of class $class$ to be $\mathbf{Y}$.
Outputs: R, a list of ROC points increasing by *fp rate*.
Require: $P > 0$ and $N > 0$

```
for x ∈ L do
    if x is a positive example then
        P_total ← P_total + c(Y, p, x)
    else
        N_total ← N_total + c(Y, n, x)
L_sorted ← L sorted decreasing by f scores
FP_cost ← 0
TP_benefit ← 0
R ← ⟨⟩
f_prev ← −∞
i ← 1
while i ≤ |L_sorted| do
    if f(i) ≠ f_prev then
        push (FP_cost/N_total, TP_benefit/P_total) onto R
        f_prev ← f(i)
    if L_sorted[i] is a positive example then
        TP_benefit ← TP_benefit + c(Y, p, L_sorted[i])
    else                                  /* i is a negative example */
        FP_cost ← FP_cost + c(Y, n, L_sorted[i])
    i ← i + 1
push (FP_cost/N, TP_benefit/P) onto R    /* This is (1,1) */
end
```

4. ROC Graphs with instance-varying costs

Another approach is to use a straightforward transformation of ROC graphs, explained in this section. Consider again the simple credit card transaction domain whose cost matrix is shown in figure 2a. For this domain we assume that a **Y** decision corresponds to approving a transaction, and **N** means denying it. The default action will be to deny a transaction. To use a cost matrix for an ROC graph it must be transformed



(a) ROC curves of two classifiers (b) ROCIV curves of the same classifiers

Figure 3. Standard ROC curves vs ROCIV **curves**

into a cost-benefit matrix where the costs are relative only to $\mathbf{Y}$ decisions.

First we subtract the first row from both rows in the matrix. Conceptually the resulting matrix corresponds to a baseline situation where all transactions are refused, so all fraud is denied and all legitimate customers are annoyed. We then negate the approve-fraudulent cell to turn it into a cost. This yields the cost-benefit matrix of figure 2b which forms the definition of the cost function $c(\mathbf{Y}, \mathbf{p}, x)$ and $c(\mathbf{Y}, \mathbf{n}, x)$.

In standard ROC graphs the x axis represents the fraction of total FP mistakes possible. In the instance-varying cost formulation it will represent the fraction of *total FP cost* possible, so the denominator will now be

$$\sum_{x \in X^+} \$20 + x$$

Similarly the y axis will be the fraction of *total TP benefits* so its denominator will be

$$\sum_{x \in X^+} 0.02x + \$20$$

Instead of incrementing TP and FP instance counts as in algorithm 1, we increment *TP_benefit* and *FP_cost* by the cost (benefit) of each negative (positive) instance as it is processed. The ROCIV points are the fractions of total benefits and costs, respectively. Conceptually this transformation corresponds to replicating instances in the instance set in proportion to their cost, though this transformation has the advantage that no actual replication is performed and non-integer costs are easily accommodated. The final algorithm is algorithm 2.

We shall call these transformed ROC curves ROCIV curves. Figure 3 illustrates the importance of this transformation. Figure 3a shows the ROC curves of two classifiers, A and B, with identical performance (their ROC curves are identical). This ROC curve shows raw classification performance with respect to positive and negative examples. Figure 3 shows the ROCIV curves of A and B when example-specific costs are taken into account. Note that performance differs significantly, and each has regions of dominance. The ROC curve in figure 3b is a poor representation of the peformance of either, and using it to select a low-cost classifier might be misleading.

Section 2.2 discussed how operating conditions could be transformed into an iso-performance line and used with ROC curves to choose the best performing classifier for those conditions. With ROCIV graphs the iso-performance line slope is calculated the same way as with ROC curves, but the error costs $c(\mathbf{N}, \mathbf{p})$ and $c(\mathbf{Y}, \mathbf{n})$ are *averages* as calculated in equations 2 and 3. Thus, when all costs of instances within a given class are equal, a ROCIV graph is identical to a ROC graph.

5. Area under a ROCIV curve

The area under a conventional ROC curve (AUC) has an important 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 [6]. In his Ph.D. thesis, Sing [11] presents an elegant and intuitive proof of this. In this section we show that the area un-

der the ROCIV(which we shall call AUCIV) is very similar: it is equivalent to the same probability, with the stipulation that *instances are chosen in proportion to their costs*. The following proof borrows heavily from the notation and structure of Sing's (indeed, this proof may be seen as a variant of his).

Definitions: Let n^+ and n^- denote the number of positive and negative examples, respectively. Denote the positive training examples by $x_1^+, \cdots, x_{n^+}^+$ and the negative training examples by $x_1^-, \cdots, x_{n^-}^-$. Let $cost(P) = \sum_{i=1}^{n^+} cost(x_i^+)$ and $cost(N) = \sum_{j=1}^{n^-} cost(x_j^-)$. We define two cost-weighted probability functions:

$$p_{\text{esc}}(x_i^+) = cost(x_i^+)/cost(P)$$
$$p_{\text{esc}}(x_j^-) = cost(x_j^-)/cost(N)$$

Theorem: *The area under an ROCIV curve of a classifier h is equivalent to the probability that h will rank a randomly chosen positive instance higher than a randomly chosen negative instance, assuming that the probability of an instance being chosen is proportional to its cost. Formally:*

$$\text{AUCIV} = \sum_{j=1}^{n^-} \sum_{i=1}^{n^+} p_{\text{esc}}(x_i^+) \cdot p_{\text{esc}}(x_j^-) \cdot \mathbf{1}_{h(x_i^+) > h(x_j^-)}$$

Proof: In order to simplify the proof we assume without loss of generality that h assigns unique scores to each of the instances.

Let $A = \{a_1, \cdots, a_n\}$ and $B = \{b_1, \cdots, b_m\}$, $a_i, b_j \in R$. The *cost weighted rank* of a sample b_j with respect to A is the summed costs of the elements of A with higher value:

$$\text{rank}_c(b_j|A) = \sum_{i=1}^{n} cost(a_i) \cdot \mathbf{1}_{a_i > b_j}$$

Assume that positive and negative samples are labeled in descending order so that $h(x_1^+) > \cdots > h(x_{n^+}^+)$ and $h(x_1^-) > \cdots > h(x_{n^-}^-)$. Figure 4 shows an unscaled ROCIV curve with the predictions arranged in descending h order. The curve steps upward with a positive example (marked by an empty circle) and to the right with a negative example (filled circle). The actual ROCIV curve would have the x axis scaled by $1/cost(N)$ and the y axis scaled by $1/cost(P)$. The area A under the curve (shaded in the figure) is related to the true AUCIV by:

$$\text{AUCIV}(h) = \frac{A}{cost(P) \cdot cost(N)}$$

Each vertical column can be thought of as belonging to a negative example, marked by the filled circle at the top of its column. The area of the column of an example is equal to the sum of the costs of the positive examples that scored higher, which is the cost-weighted rank of the negative example with respect to the positive examples. Therefore,

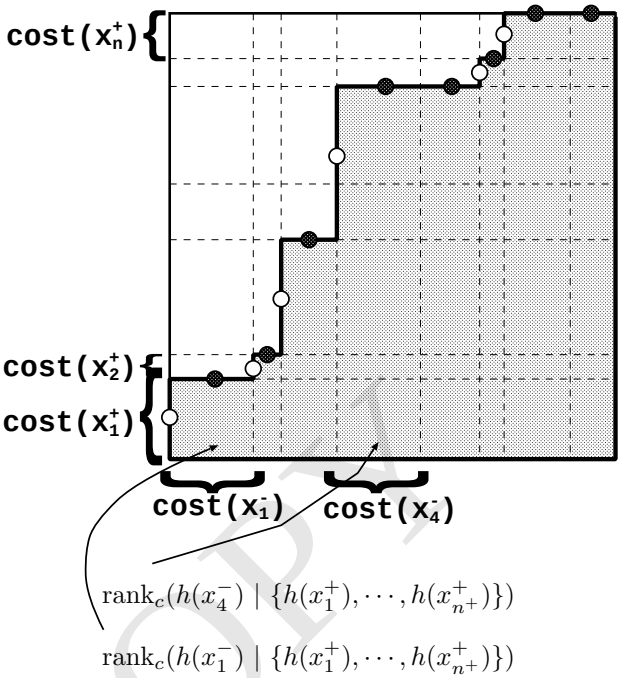


Figure 4. A cost-scaled ROCIV **plot**

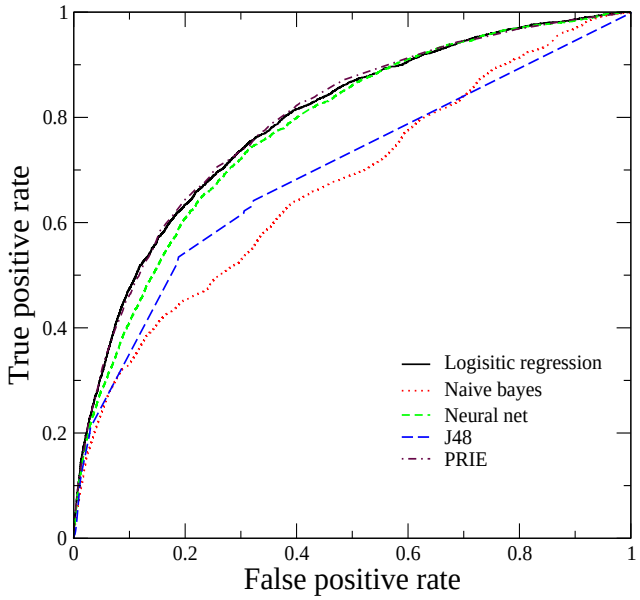
$$A = \sum_{j=1}^{n^-} \text{rank}_c(x_j^- | \{x_1^+, \cdots, x_{n^+}^+\})$$

Rewriting this in terms of the AUCIV we have:

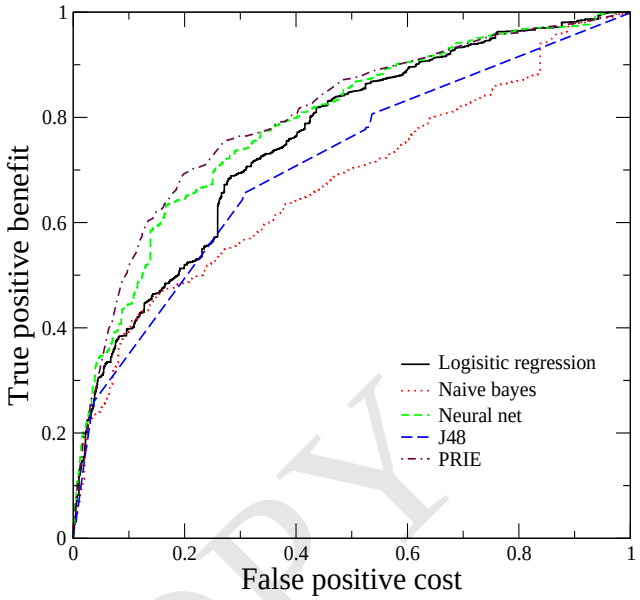
$$\begin{aligned} A &= \frac{\sum_{j=1}^{n^-} \text{rank}_c(x_j^- | \{x_1^+, \cdots, x_{n^+}^+\})}{cost(P) \cdot cost(N)} \\ &= \frac{\sum_{j=1}^{n^-} \sum_{i=1}^{n^+} cost(x_i^+) \cdot cost(x_j^-) \cdot \mathbf{1}_{h(x_i^+) > h(x_j^-)}}{cost(P) \cdot cost(N)} \\ &= \sum_{j=1}^{n^-} \sum_{i=1}^{n^+} \frac{cost(x_i^+)}{cost(P)} \cdot \frac{cost(x_j^-)}{cost(N)} \cdot \mathbf{1}_{h(x_i^+) > h(x_j^-)} \\ &= \sum_{j=1}^{n^-} \sum_{i=1}^{n^+} p_{\text{esc}}(x_i^+) \cdot p_{\text{esc}}(x_j^-) \cdot \mathbf{1}_{h(x_i^+) > h(x_j^-)} \end{aligned}$$

■

6. Empirical Demonstrations



(a) ROC curves



(b) ROCIV curves

Figure 5. Classifier performance, charitable donation data

To illustrate the importance of the ROCIV transformation, we present an empirical demonstration of its use for evaluation classifier performance on several domains with example-specific costs.

It should be emphasized that the point of the demonstrations here is to show classifier performance on realistic cost-varying domains. No claim is made that these are the best learning algorithms for these domains, or that the relative performance reported is typical for these algorithms.

6.1. Charitable donations

The first domain is a proprietary dataset used at Hewlett-Packard, provided by a third party. The data comprise about 60,000 records of solicitation response data for a charity. The independent variables measure, for each person, a basic history comprising recency and frequency of donations to the charity, and some features capturing periodicity of donations. The dependent (response) variable is the donation amount of a single mailing.

The classification problem is to determine whether a given person will donate. From the donation amount and the mailing costs, misclassification costs may be derived. The cost of a false positive is the cost of mailing a solicitation for which no donation is received. For this study the mailing amount was assumed to be \$1. The benefit of a true positive is the donation amount x minus the \$1 mailing cost. For this mailing, when a donation is given the amounts varied greatly, from \$1 to \$1500 (mean \$24 $\pm$ \$49).

Several classification models were trained on this data, including Logistic regression, Naive bayes, a Neural network, J48 (a decision tree learner similar to C4.5) and PRIE. PRIE is a rule learning system designed to maximize ROC performance[1].

Figure 5 shows resulting ROC and ROCIV curves for this domain. Note that in the ROC curves the performance of PRIE and logistic regression are nearly indistinguishable, and the performance of the neural net is very close. In the ROCIV curves the relationships are somewhat different. The greatest difference is that logistic regression performs noticeably worse, dropping substantially below that of PRIE over most of the false positive range. The ROC curve of logistic regression is nearly monotonic, whereas its ROCIV curve has prominent concavities.

This differing performance may also be explained by looking at the costs of instances as they are ranked by each algorithm. These costs are shown in figure 10. The Y axis shows the cost of each instance, with the instances ordered from highest assigned score (at the left) to lowest (at the right). There are many instances of cost -1 (the non-responders) which are difficult to see because of the scale. The ROCIV curves are essentially showing the simultaneous integrals of these instance costs: the true positive benefit is the integral

[1]PRIE's algorithm has not been published previously but the details of its operation are not germane to the investigation here.

	fraudulent	legitimate
refuse	\$20	$-\$20$
approve	$-x$	$0.05x$

(a)

	fraudulent	legitimate
refuse	0	0
approve	$\$20 + x$	$0.05x + \$20$

(b)

Figure 7. Matrices for the credit approval domain. (a) original benefit matrix, (b) transformed cost-benefit matrix

	fraudulent	legitimate
no alarm	$-\$1 - \$0.20x$	$\$0.10x$
alarm	\$1	$-\$2$

(a)

	fraudulent	legitimate
no alarm	0	0
alarm	$\$2 + \$0.20x$	$-\$2 - \$0.10x$

(b)

Figure 8. Matrices for the cell phone fraud detection domain. (a) original benefit matrix, (b) transformed cost-benefit matrix

of instance benefits above the line, and the false positive cost is the integral of instance costs below the line. An ideal classifier would rank all positive instances first, decreasing by benefit, followed by all negative instances, increasing by cost. One can see informally that the "cost mass" of the instances ranked by PRIE is slightly to the left of the cost mass of Logistic regression. This difference is reflected in the resulting ROCIV curves.

6.2. Credit scoring

The "German" domain is a dataset of German credit information provided by Dr. Hans Hofmann and donated to the UCI Machine Learning Repository [1]. Each record contains information about a person requesting a loan, including various demographic information, a summary of the credit history, and the amount of the loan.

We trained classifiers on this domain using the cost matrix in figure 7b, generated from 7a. Three classifier model types were induced using the Weka package [16]: a multilayer perceptron, denoted "Neural net"; a naive bayes learner, and a simple logistic regression model. These model types were chosen because they were reasonably good at producing instance scores rather than simply assigning a class label to each instance.

Half of the instances were used for training, half for testing. The ROC curves on the test set are shown in figure 6a. Looking at the convex hull of the classifiers, each has a region of superiority: the neural net is most conservative ($0 \leq FP \leq 0.10$), followed by simple logistic regression ($.10 < FP \leq 0.78$), followed by Naive bayes ($0.78 < FP \leq 1.0$).

The corresponding ROCIV curves, shown in figure 6b, tell a different story. When considering individual instances' costs and benefits, the regions of superiority do not corresponding to those of figure 6a. Naive bayes is superior over a much larger region, and the Neural net is virtually dominated completely.

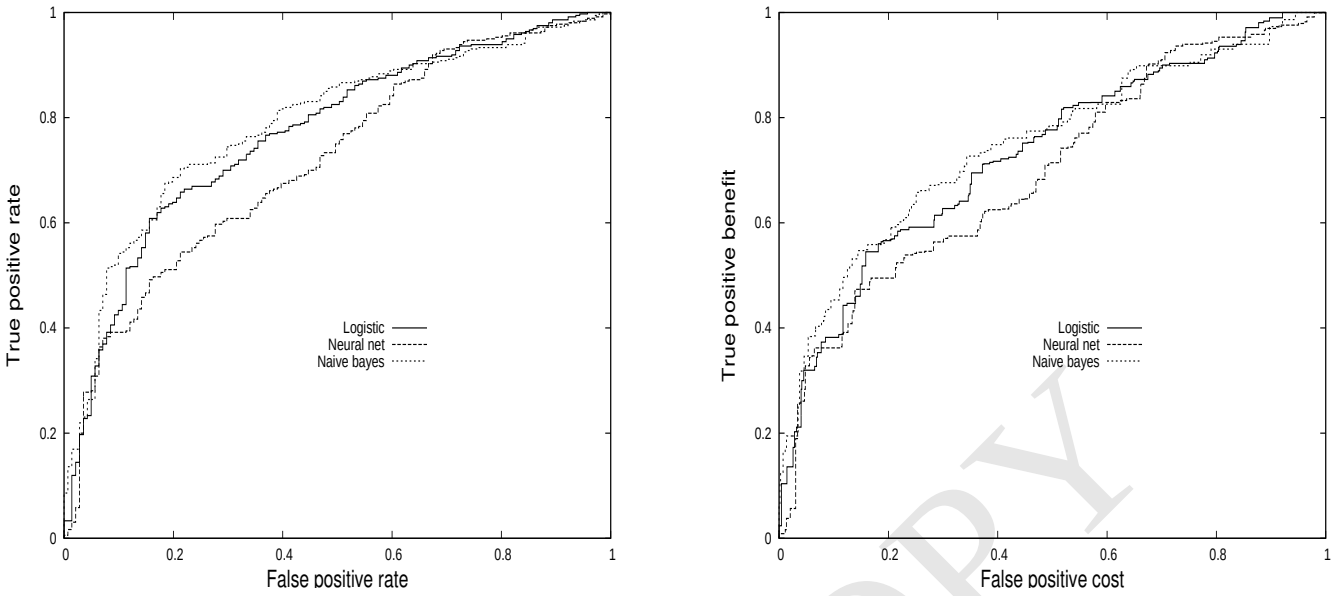
6.3. Fraudulant phone call detection

As another demonstration of instance-varying costs we examined a simplified form of cell phone fraud detection. An historical dataset of 50,000 cell phone calls was used: 25,000 for training and 25,000 for testing. Fawcett and Provost [5] previously described the domain in detail. The version used here is simplified in that the goal is to classify individual calls rather than to profile user behavior over time and to classify account-days. The classifiers learned here are similar to the fraudulent call classifiers described by Fawcett and Provost [5] in section 6.3.

The independent variables in this domain are attributes of a given cell phone call, such as its originating and terminating locations, the length, the carrier used, time of day, and so on. The dependent variable is simply a binary flag indicating whether the call was fraudulent.

Figure 8 shows the benefit and cost-benefit matrices for this domain. In this domain, the default action will be to do nothing (the account is not suspected of fraud). An alarm consists of flagging the account and temporarily disabling it. The top matrix is justified as follows[2]. Alarming on a fraudulent call serves to inhibit some future fraud, a benefit valued at approximately \$1. Issuing an alarm on a legitimate customer temporarily shuts down the account and irritates the

[2]While the costs and benefits used here are plausible, they are fabricated and should not be interpreted as actual values assigned by Bell Atlantic Mobile.



(a) ROC curves (b) ROCIV curves

Figure 6. Classifier performance, credit scoring domain

customer, a benefit of -$2. Missing a fraudulent call costs a certain amount in overhead and toll costs (long distance and international charges are billed to the carrier) and eventual customer irritation. Finally, allowing a legitimate call generates a certain amount of revenue.

Following the transformation described in section 4, the cost-benefit matrix in figure 8b is derived. The results are shown in figure 9. The difference between the ROC and the ROCIV curves in this domain are very slight, perhaps because the relative performance of the classifiers are so distinct in this domain. The only substantial change is that Naive Bayes declines noticeably in the ROCIV curves, and no longer dominates JRip.

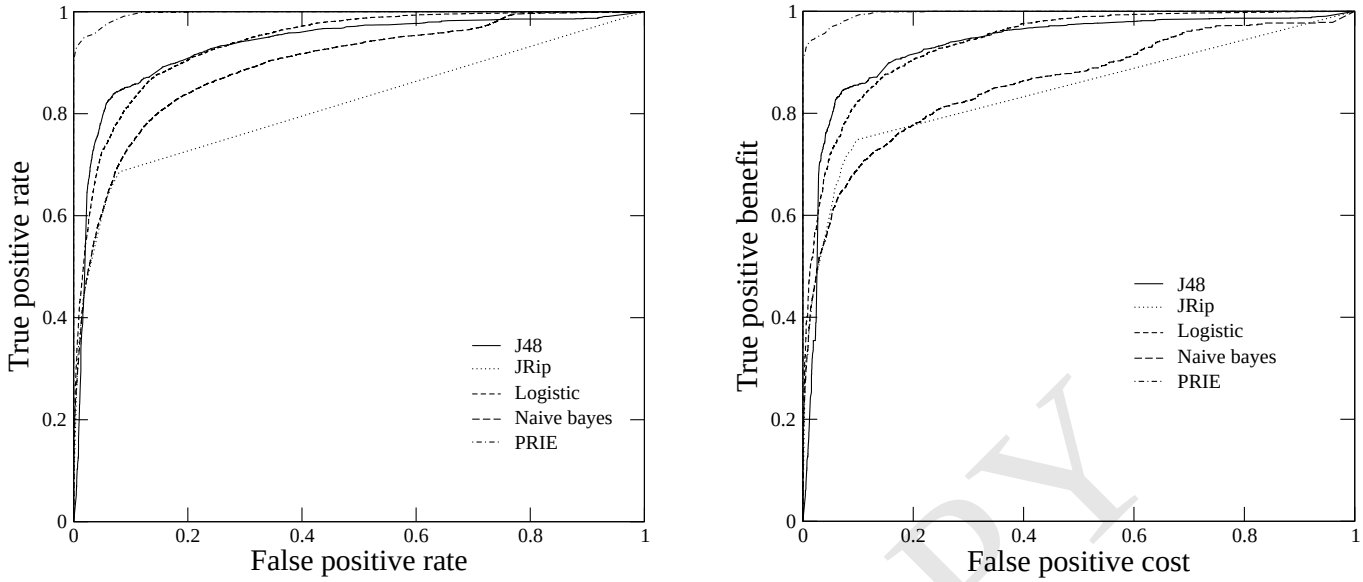
7. Discussion

This paper has presented a straightforward transformation of ROC graphs, called ROCIV graphs, that accomodate instance varying costs. The new curves have a intuitive interpretation, in that the axes are now scaled by instance costs within each class. The area under the ROCIV curves has a straightforward interpretation: it is equivalent to the probability that a randomly chosen positive instance will be ranked more highly than a randomly chosen negative instance, given that each is chosen in proportion to their costs. Finally, we have demonstrated the transformation on three domains and have seen cases in which the ROCIV curves show considerably different regions of classifier superiority than the corresponding ROC curves.

It is important to mention two caveats in adopting this transformation. First, while example costs may vary, ROC analysis requires that costs always be negative and benefits always be positive. For example, if a cost function were defined as $c(\mathbf{Y}, \mathbf{p}, x) = x - \$20$, with example x values ranging in $[0, 40]$, this would be violated for x in $[0, 20]$.

Second, incorporating error costs into the ROC graph in this way introduces an additional assumption into a researcher's testing environment. Traditional ROC graphs assume that the *fp rate* and *tp rate* metrics of the test population will be similar to those of the training population; in particular that a classifier's performance on random samples will be similar. This new formulation adds the assumption that the example *costs* will be similar as well. In other words, ROCIV curves assume that not only will the classifier continue to score instances similarly between the training and testing sets, but the costs and benefits of those instances will be similar between the sets too.

Adding this assumption partially violates the cost-insensitivity of ROC curves. A standard ROC curve is insensitive to changes in both *intra*-class and *inter*-class error costs. A ROCIV curve remains insensitive to *intra*-class error cost variations but will be sensitive to *inter*-class cost variations. As such, a researcher using ROCIV curves should check inter-class error cost distributions in the training and testing environments to ensure that they are stable.



(a) ROC curves (b) ROCIV curves

Figure 9. Classifier performance, fraudulent call classification

References

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

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

[3] C. Elkan. The foundations of cost-sensitive learning. In *IJCAI-01*, pages 973–978, 2001.

[4] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. Tech Report HPL-2003-4, HP Laboratories, 2003. Available: `http://www.purl.org/NET/tfawcett/papers/ROC101.pdf`.

[5] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[6] 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.

[7] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing misclassification costs. In *Proc. 11th International Conference on Machine Learning*, pages 217–225. Morgan Kaufmann, 1994.

[8] 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.

[9] F. Provost and T. Fawcett. Robust classification systems for imprecise environments. In *Proceedings of AAAI-98*, pages 706–713, Menlo Park, CA, 1998. AAAI Press. Available: `http://www.purl.org/NET/tfawcett/papers/aaai98-dist.ps.gz`.

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

[11] T. Sing. *Learning Localized Rule Mixtures by Maximizing the Area under the ROC Curve, with an Application to the Prediction of HIV-1 Coreceptor Usage*. PhD thesis, Max-Planck-Institut für Informatik Saarbrücken, Mar 2004.

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

[13] J. A. Swets, R. M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, 283:82–87, October 2000. Available: `http://www.psychologicalscience.org/newsresearch/publications/journals/%siam.pdf`.

[14] P. D. Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, 2000.

[15] M. C. Weinstein and H. V. Fineberg. *Clinical Decision Analysis*. Philadelphia, PA: W. B. Saunders Company, 1980.

[16] 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/`.

[17] K. H. Zou. Receiver operating characteristic (ROC) literature research. On-line bibliography available from `http://splweb.bwh.harvard.edu:8000/pages/ppl/zou/roc.html`, 2002.

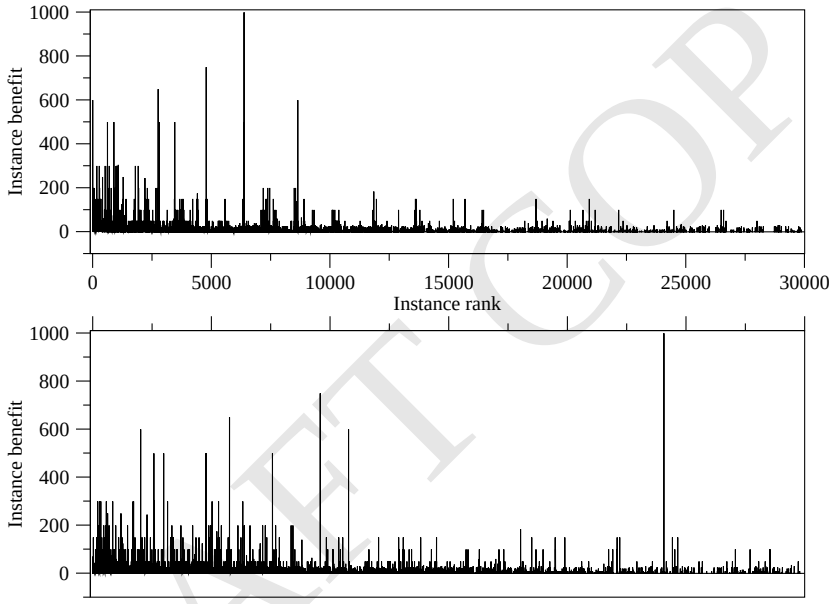


Figure 10. Charitable donation data: Costs of instances as ordered by PRIE (top) and Logistic regression (bottom).