ROC Graphs: Notes and Practical Considerations for Data Mining Researchers

Tom Fawcett
MS 1143
HP Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
tom_fawcett@hp.com

January 2003

Abstract

Receiver Operating Characteristics (ROC) graphs are a useful technique for organizing classifiers and visualizing their performance. ROC graphs are commonly used in medical decision making, and in recent years have been increasingly adopted in the machine learning and data mining research communities. Although ROC graphs are apparently simple, there are some common misconceptions and pitfalls when using them in practice. This article serves both as a tutorial introduction to ROC graphs and as a practical guide for using them in research.

Keywords: Machine learning, classification, classifier evaluation, ROC, visualization

1 Introduction

An ROC graph is a 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 (Egan, 1975; Swets, Dawes, & Monahan, 2000a). ROC analysis has been extended for use in visualizing and analyzing the behavior of diagnostic systems (Swets, 1988). The medical decision making community has an extensive literature on the use of ROC graphs for diagnostic testing (Zou, 2002). Swets, Dawes and Monahan (2000a) recently brought ROC curves to the attention of the wider public with their *Scientific American* article.

One of the earliest adopters of ROC graphs in machine learning was Spackman (1989), who demonstrated the value of ROC curves in evaluating and comparing algorithms. Recent years have seen an increase in the use of ROC graphs in the machine learning community. In addition to being a generally useful performance graphing method, they have properties that make them especially useful for domains with skewed class distribution and unequal classification error costs. These characteristics of ROC graphs have become increasingly important as research continues into the areas of cost-sensitive learning and learning in the presence of unbalanced classes.

Most books on data mining and machine learning, if they mention ROC graphs at all, have only a brief description of the technique. ROC graphs are conceptually simple, but there are some non-obvious complexities that arise when they are used in research. There are also common misconceptions and pitfalls when using them in practice.

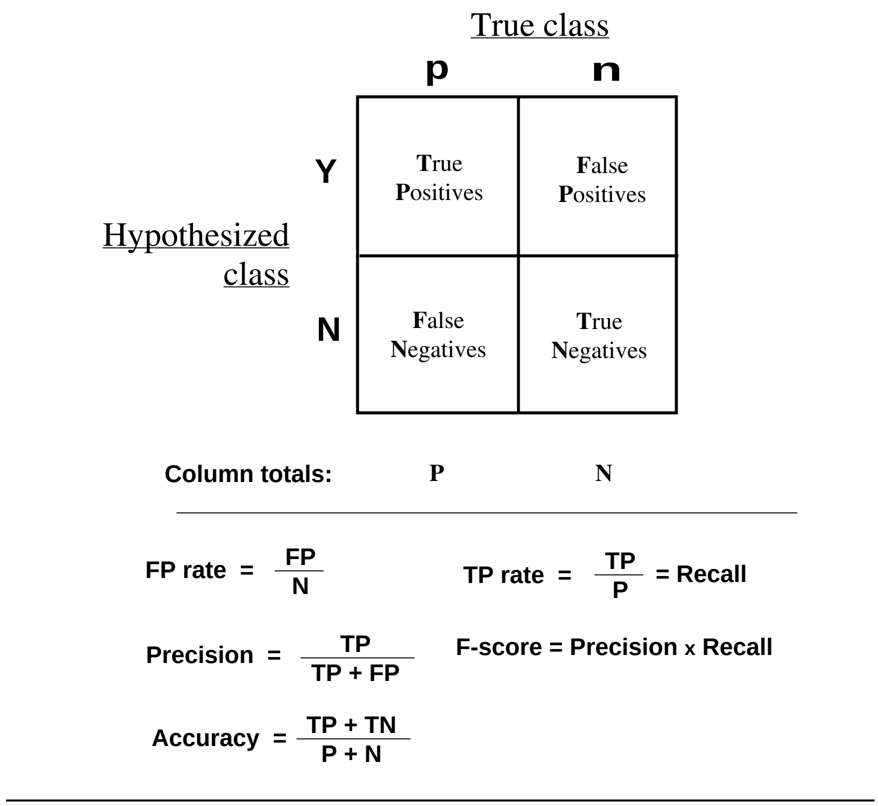


Figure 1: A confusion matrix and several common performance metrics that can be calculated from it

This article attempts to serve as a tutorial introduction to ROC graphs and as a practical guide for using them in research. It collects some important observations that are perhaps not obvious to many in the community. Some of these points have been made in previously published articles, but they were often buried in text and were subsidiary to the main points. Other notes are the result of information passed around in email between researchers, but left unpublished. The goal of this article is to advance general knowledge about ROC graphs so as to promote better evaluation practices in the field.

This article is divided into two parts. The first part, comprising sections 2 through 7, covers basic issues that will emerge in most research uses of ROC graphs. Each topic has a separate section and is treated in detail, usually including algorithms. Researchers intending to use ROC curves seriously in their work should be familiar with this material. The second part, in section 8, covers some related but ancillary topics. They are more esoteric and are discussed in less detail, but pointers to further reading are included. Finally, appendix A contains a few function definitions from computational geometry that are used in the algorithms.

Note: Implementations of the algorithms in this article, in the Perl language, are collected in an archive available from: `http://www.purl.org/NET/tfawcett/software/ROC_algs.tar.gz`

2 Classifier Performance

We begin by considering classification problems using only two classes. Formally, each instance I is mapped to one element of the set $\{\mathbf{p}, \mathbf{n}\}$ of positive and negative class labels. A *classification model* (or *classifier*) is a mapping from instances to predicted classes. Some classification models produce a continuous output (*e.g.*, an estimate of an instance's class membership probability) to which different thresholds may be applied to predict class membership. Other models produce a discrete class label indicating only the predicted class of the instance. To distinguish between the actual class and the predicted class we use the labels $\{\mathbf{Y}, \mathbf{N}\}$ for the class predictions produced by a model.

Given a classifier and an instance, there are four possible outcomes. If the instance is positive and it is classified as positive, it is counted as a *true positive*; if it is classified as negative, it is counted as a *false negative*. If the instance is negative and it is classified as negative, it is counted as a *true negative*; if it is classified as positive, it is counted as a *false positive*. Given a classifier and a set of instances (the test set), a two-by-two *confusion matrix* (also called a contingency table) can be constructed representing the dispositions of the set of instances. This matrix forms the basis for many common metrics.

Figure 1 shows a confusion matrix and equations of several common metrics that can be calculated from it. The numbers along the major diagonal represent the correct decisions made, and the numbers off this diagonal represent the errors—the confusion—between the various classes. The **True Positive rate** (also called *hit rate* and *recall*) of a classifier is estimated as:

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

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

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

Additional terms associated with ROC curves are:

$$\begin{aligned}
\text{Sensitivity} &= \text{Recall} \\
\text{Specificity} &= \frac{\text{True negatives}}{\text{False positives} + \text{True negatives}} \\
&= 1 - \text{FP rate} \\
\text{Positive predictive value} &= \text{Precision}
\end{aligned}$$

3 ROC Space

ROC graphs are two-dimensional graphs in which *TP rate* is plotted on the Y axis and *FP rate* is plotted on the X axis. An ROC graph depicts relative trade-offs between benefits (true positives) and costs (false positives). Figure 2 shows an ROC graph with five classifiers labeled A through E.

A *discrete* classifier is one that outputs only a class label. Each discrete classifier produces an *(FP rate, TP rate)* pair, which corresponds to a single point in ROC space. The classifiers in figure 2 are all discrete classifiers.

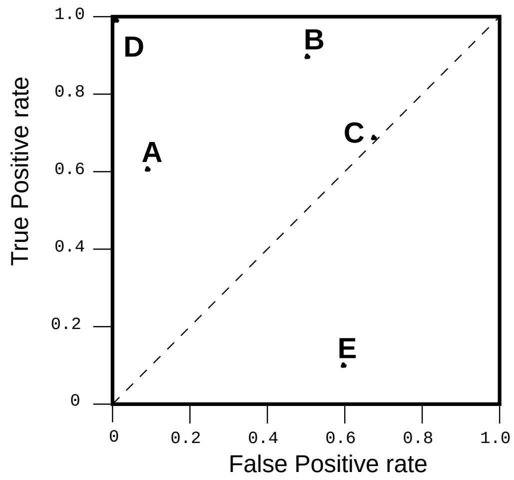


Figure 2: A basic ROC graph showing five discrete classifiers.

Several points in ROC space are important 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 $(1, 1)$.

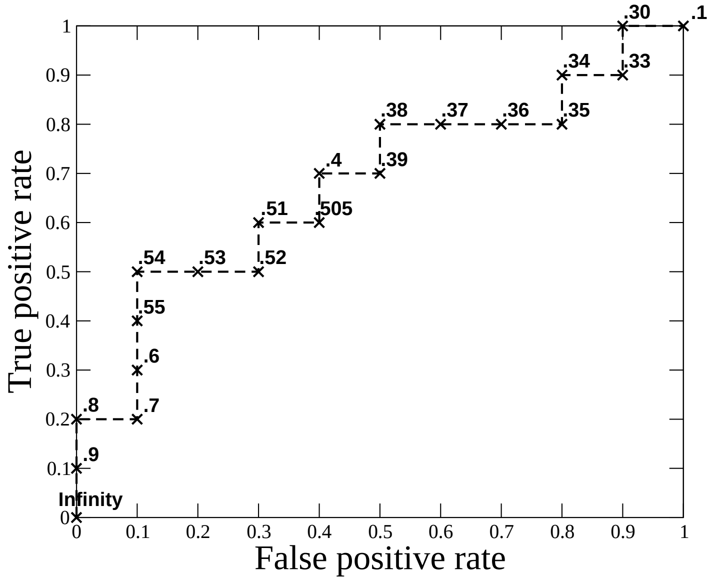
The point $(0, 1)$ represents perfect classification. D's performance is perfect as shown.

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. Classifiers appearing on the left hand-side of an ROC graph, near the X axis, may be thought of as "conservative": they make positive classifications only with strong evidence so they make few false positive errors, but they often have low true positive rates as well. Classifiers on the upper right-hand side of an ROC graph may be thought of as "liberal": they make positive classifications with weak evidence so they classify nearly all positives correctly, but they often have high false positive rates. In figure 2, A is more conservative than B. Many real world domains are dominated by large numbers of negative instances, so performance in the far left-hand side of the ROC graph becomes more interesting.

3.1 Random Performance

The diagonal line $y = x$ represents the strategy of randomly guessing a class. For example, if a classifier randomly guesses the positive class half the time, it can be expected to get half the positives and half the negatives correct; this yields the point $(0.5, 0.5)$ in ROC space. If it guesses the positive class 90% of the time, it can be expected to get 90% of the positives correct but its false positive rate will increase to 90% as well, yielding $(0.9, 0.9)$ in ROC space. Thus a random classifier will produce a ROC point that "slides" back and forth on the diagonal based on the frequency with which it guesses the positive class. In order to get away from this diagonal into the upper triangular region, the classifier must exploit some information in the data. In figure 2, C's performance is virtually random. At $(0.7, 0.7)$, C may be said to be guessing the positive class 70% of the time,

Any classifier that appears in the lower right triangle performs worse than random guessing. This triangle is therefore usually empty in ROC graphs. However, note that the decision space is symmetrical about the diagonal



Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

Figure 3: The ROC "curve" created by thresholding a test set. The table at right shows twenty data and the score assigned to each by a scoring classifier. The graph at left shows the corresponding ROC curve with each point labeled by the threshold that produces it.

separating the two triangles. If we negate a classifier—that is, reverse its classification decisions on every instance—its true positive classifications become false positive mistakes, and its false positives become true positives. Therefore, any classifier that produces a point in the lower right triangle can be negated to produce a point in the upper left triangle. In figure 2, E performs much worse than random, and is in fact the negation of A.

Given an ROC graph in which a classifier's performance appears to be slightly better than random, it is natural to ask: "is this classifier's performance truly significant or is it only better than random by chance?". There is no conclusive test for this, but Forman (2002) has shown a methodology that addresses this question with ROC curves.

4 Curves in ROC space

Many classifiers, such as decision trees or rule sets, are designed to produce only a class decision, *i.e.*, a **Y** or **N** on each instance. When such a discrete classifier is applied to a test set, it yields a single confusion matrix, which in turn corresponds to one ROC point. Thus, a discrete classifier produces only a single point in ROC space.

Some classifiers, such as a Naive Bayes classifier or a neural network, naturally yield an instance *probability* or *score*, a numeric value that represents the degree to which an instance is a member of a class. These values can be strict probabilities, in which case they adhere to standard theorems of probability; or they can be general, uncalibrated scores, in which case the only property that holds is that a higher score indicates a higher probability. We shall call both a *probabilistic* classifier, in spite of the fact that the output may not be a proper probability[1].

Such a *ranking* or *scoring* classifier can be used with a threshold to produce a discrete (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. Conceptually, we may imagine varying a threshold from $-\infty$ to $+\infty$ and tracing a

[1] Techniques exist for converting an uncalibrated score into a proper probability but this conversion is unnecesary for ROC curves.

Algorithm 1 Conceptual method for calculating an ROC curve. See algorithm 2 for a practical method.

Inputs: L, the set of test instances; $f(i)$, the probabilistic classifier's estimate that instance i is positive; min and max, the smallest and largest values returned by f; $increment$, the smallest difference between any two f values.

```
for t = min to max by increment do
  FP ⇐ 0
  TP ⇐ 0
  for i ∈ L do
    if f(i) ≥ t then                          /* This example is over threshold */
      if i is a positive example then
        TP ⇐ TP + 1
      else                                    /* i is a negative example, so this is a false positive */
        FP ⇐ FP + 1
  Add point (FP/N, TP/P) to ROC curve
end
```

curve through ROC space. Algorithm 1 describes this basic idea. Computationally, this is a poor way of generating an ROC curve, and the next section describes a more efficient and careful method.

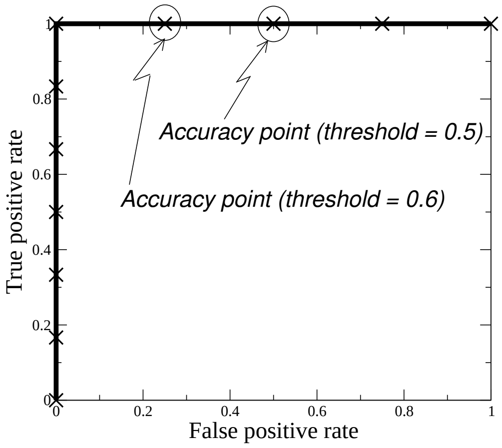
Figure 3 shows an example of an ROC "curve" on a test set of twenty instances. The instances, ten positive and ten negative, are shown in the table beside the graph. Any ROC curve generated from a finite set of instances is actually a step function, which approaches a true curve as the number of instances approaches infinity. The step function in figure 3 is taken from a very small instance set so that each point's derivation can be understood. In the table of figure 3, the instances are sorted by their scores, and each point in the ROC graph is labeled by the score threshold that produces it. A threshold of $+\infty$ produces the point $(0, 0)$. As we lower the threshold to 0.9 the first positive instance is classified positive, yielding $(0, 0.1)$. As the threshold is further reduced, the curve climbs up and to the right, ending up at $(1, 1)$ with a threshold of 0.1. Note that lowering this threshold corresponds to moving from the "conservative" to the "liberal" areas of the graph.

Although the test set is very small, we can make some tentative observations about the classifier. It appears to perform better in the more conservative region of the graph; the ROC point at $(0.1, 0.5)$ produces its highest accuracy (70%). This is equivalent to saying that the classifier is better at identifying likely positives than at identifying likely negatives. Note also that the classifier's best accuracy occurs at a threshold of .54, rather than at 0.5 as we might expect (which yields 60%). The next section discusses this phenomenon.

4.1 Relative versus absolute scores

An important point about ROC graphs is that they measure the ability of a classifier to produce good *relative* instance scores. A classifier need not produce accurate, calibrated probability estimates; it need only produce relative accurate scores that serve to discriminate positive and negative instances.

Consider the simple instance scores shown in figure 4, which came from a Naive Bayes classifier. Comparing the hypothesized class (which is **Y** if score$>$ 0.5, else **N**) against the true classes, we can see that the classifier gets instances 7 and 8 wrong, yielding 80% accuracy. However, consider the ROC curve on the left side of the figure. The curve rises vertically from $(0, 0)$ to $(0, 1)$, then horizontally to $(1, 1)$. This indicates perfect classification performance



Inst no.	Class True	Class Hyp	Score
1	**p**	**Y**	0.99999
2	**p**	**Y**	0.99999
3	**p**	**Y**	0.99993
4	**p**	**Y**	0.99986
5	**p**	**Y**	0.99964
6	**p**	**Y**	0.99955
7	**n**	**Y**	0.68139
8	**n**	**Y**	0.50961
9	**n**	**N**	0.48880
10	**n**	**N**	0.44951

Figure 4: Scores and classifications of ten instances, and the resulting ROC curve.

on this test set. Why is there a discrepancy?

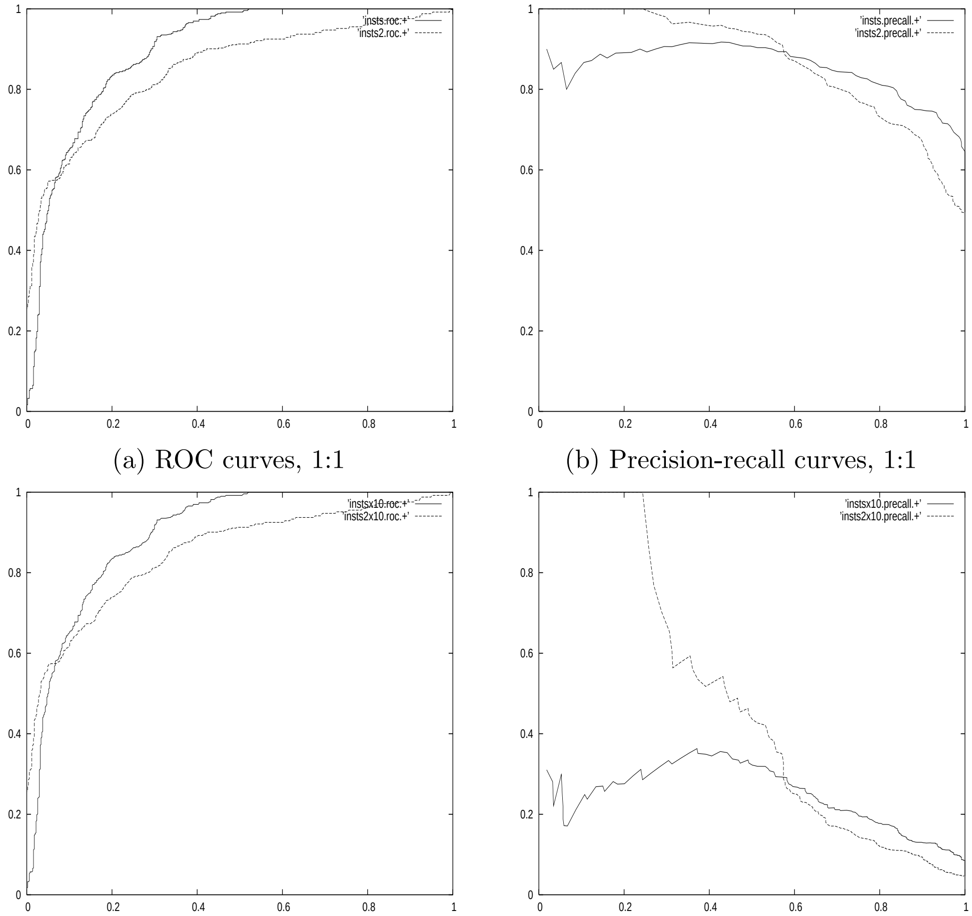
The explanation lies in what each is measuring. The ROC curve shows the ability of the classifier to rank the positive instances relative to the negative instances, and it is indeed perfect in this ability. The accuracy metric imposes a threshold (score> 0.5) and measures the resulting classifications with respect to the scores. The accuracy measure would be appropriate if the scores were proper probabilities, but they are not. Another way of saying this is that the scores are not *properly calibrated*, as true probabilities are. In ROC space, the imposition of a 0.5 threshold results in the performance designated by the circled "accuracy point" in figure 4. This operating point is suboptimal. We could use the training set to estimate a prior for $p(\mathbf{p}) = 6/10 = 0.6$ and use this as a threshold, but it would still produce suboptimal performance (90% accuracy).

One way to eliminate this phenomenon is to calibrate the classifier scores. There are some methods for doing this (Zadrozny & Elkan, 2001). Another approach is to use an ROC method that chooses operating points based on their relative performance, and there are methods for doing this as well (Provost & Fawcett, 1998, 2001). These latter methods will be discussed briefly in section 8.1.

A consequence of relative scoring is that classifier scores should not be compared across model classes. One model class may be designed to produce scores in the range [0,1] while another produces scores in [-1,+1] or [1,100]. Comparing model performance at a common threshold will be meaningless.

4.2 Class skew

ROC curves have an attractive property: they are insensitive to changes in class distribution. If the proportion of positive to negative instances changes in a test set, the ROC curves will not change. To see why this is so, consider the confusion matrix in figure 1. Note that the class distribution—the proportion of positive to negative instances—is the relationship of the left (+) column to the right (-) column. Any performance metric that uses values from both columns will be inherently sensitive to class skews. Metrics such as accuracy, precision, lift and F scores use values from both columns of the confusion matrix. As a class distribution changes these measures will change as well, even



(a) ROC curves, 1:1 (b) Precision-recall curves, 1:1

(c) ROC curves, 1:10 (d) Precision-recall curves, 1:10

Figure 5: ROC and precision-recall curves under class skew.

if the fundamental classifier performance does not. ROC graphs are based upon TP rate and FP rate, in which each dimension is a strict columnar ratio, so do not depend on class distributions.

To some researchers, large class skews and large changes in class distributions may seem contrived and unrealistic. However, class skews of 10^1 and 10^2 are very common in real world domains, and skews up to 10^6 have been observed in some domains (Clearwater & Stern, 1991; Fawcett & Provost, 1996; Kubat, Holte, & Matwin, 1998; Saitta & Neri, 1998). Substantial changes in class distributions are not unrealistic either. For example, in medical decision making epidemics may cause the incidence of a disease to increase over time. In fraud detection, proportions of fraud varied significantly from month to month and place to place (Fawcett & Provost, 1997). Changes in a manufacturing practice may cause the proportion of defective units produced by a manufacturing line to increase or decrease. In each of these examples the prevalance of a class may change drastically without altering the fundamental characteristic of the class, *i.e.*, the target concept.

Precision and recall are common in information retrieval for evaluating retrieval (classification) performance (Lewis, 1990, 1991). Precision-recall graphs are commonly used where static document sets can sometimes be assumed; however, they are also used in dynamic environments such as web page retrieval, where the number of pages irrelevant to a query (N) is many orders of magnitude greater than P and probably increases steadily over time as web pages are created.

To see the effect of class skew, consider the curves in figure 5, which show two classifiers evaluated using ROC curves and precision-recall curves. In 5a and b, the test set has a balanced 1:1 class distribution. Graphs 5c and d show the same two classifiers on the same domain, but the number of negative instances has been increased ten-fold. Note that the classifiers and the underlying concept has not changed; only the class distribution is different. Observe that the ROC graphs in 5a and 5c are identical, while the precision-recall graphs in 5b and 5d differ dramatically. In some cases, the conclusion of which classifier has superior performance can change with a shifted distribution.

4.3 Creating scoring classifiers

Many classifier models are discrete: they are designed to produce only a class label from each test instance. However, we often want to generate a full ROC curve from a classifier instead of just a single point. To this end we want to generate scores from a classifier rather than just a class label. There are several ways of producing such scores.

Many discrete classifier models may easily be converted to scoring classifiers by "looking inside" them at the instance statistics they keep. For example, a decision tree determines a class label of a leaf node from the proportion of instances at the node; the class decision is simply the most prevalent class. These class proportions may serve as a score (Provost & Domingos, 2001). A rule learner keeps similar statistics on rule confidence, and the confidence of a rule matching an instance can be used as a score (Fawcett, 2001).

Even if a classifier only produces a class label, an aggregation of them may be used to generate a score. MetaCost (Domingos, 1999) employs bagging to generate an ensemble of discrete classifiers, each of which produces a vote. The set of votes could be used to generate a score[2].

Finally, some combination of scoring and voting can be employed. For example, rules can be provide basic probability estimates, which may then be used in weighted voting (Fawcett, 2001)

5 Efficient generation of ROC curves

Given a test set, we often want to generate an ROC curve efficiently from it. Although some researchers have employed methods like algorithm 1, this method is neither efficient nor practical: it requires knowing max, min and $increment$, which must be estimated from the test set and f values. It involves two nested loops; because the outer loop must increment t at least n times, the complexity is $O(n^2)$ in the number of test set instances.

A much better algorithm can be created if we exploit the monotonicity of thresholded classifications. Any instance that is classified positive with respect to a given threshold will be classified positive for all lower thresholds as well. Therefore, we can simply sort the test instances by f scores, from highest to lowest, and move down the list, processing one instance at a time and updating TP and FP as we go. In this way we can create an ROC graph from a linear scan.

[2]MetaCost actually works in the opposite direction because its goal is to generate a discrete classifier. It first creates a probabilistic classifier, then applies knowledge of the error costs and class skews to relabel the instances so as to "optimize" their classifications. Finally, it learns a specific discrete classifier from this new instance set. Thus, MetaCost is not a good method for creating a good scoring classifier, though its bagging method may be.

Algorithm 2 Practical method for calculating an ROC curve from a test set

Inputs: L, the set of test instances; $f(i)$, the probabilistic classifier's estimate that instance i is positive.
Outputs: R, a list of ROC points from (0,0) to (1,1)

1: $L_{sorted} \Leftarrow L$ sorted decreasing by f scores
2: $FP \Leftarrow 0$
3: $TP \Leftarrow 0$
4: $R \Leftarrow \langle\rangle$
5: $f_{prev} \Leftarrow -\infty$
6: **for** $i \in L_{sorted}$ **do**
7:   **if** $f(i) \neq f_{prev}$ **then**
8:     ADD_POINT$\left(\left(\frac{FP}{N}, \frac{TP}{P}\right), R\right)$
9:     $f_{prev} \Leftarrow f(i)$
10:   **if** i is a positive example **then**
11:     $TP \Leftarrow TP + 1$
12:   **else** /* i is a negative example, so this is a false positive */
13:     $FP \Leftarrow FP + 1$
14: ADD_POINT$\left(\left(\frac{FP}{N}, \frac{TP}{P}\right), R\right)$
15: **end**

1: **subroutine** ADD_POINT(P, R)
2: push P onto R
3: **end subroutine**

The new algorithm is shown in algorithm 2. TP and FP both 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. R is initialized to contain the point $(0, 0)$ and when the final point is processed (either by statement 13 or 18), R will end with $(1, 1)$.

Let n be the number of points in the test set. This algorithm requires an $O(n \log n)$ sort followed by an $O(n)$ scan down the list, resulting in $O(n \log n)$ total complexity.

5.1 Equally scored instances

The statements in lines 7–9 need some explanation. These steps are necessary in order to correctly handle sequences of equally scored instances. Consider the ROC curve shown in figure 6. Assume we have a test set in which there is a sequence of instances, four negatives and six positives, all scored equally by f. The sort in line 1 of algorithm 2 does not impose any specific ordering on these instances since their f scores are equal. What happens when we create an ROC curve? In one extreme case, all the positives end up at the top of the sequence, and we generate the "optimistic" upper L segment shown in figure 6. In the opposite extreme, all the negatives end up at the top of the sequence, and we get the "pessimistic" lower L shown in figure 6. Any mixed ordering of the instances will give a different set of step segments within the rectangle formed by these two extremes. However, we want the ROC curve to represent the *expected* performance of the classifier, which, lacking any other information, is the average of the pessimistic and optimistic segments. This average is the diagonal of the rectangle, and can be created in the ROC curve algorithm by not emitting an ROC point until all instances of equal f values have been processed. This is what the f_{prev} variable and the **if** statement of line 7 accomplish.

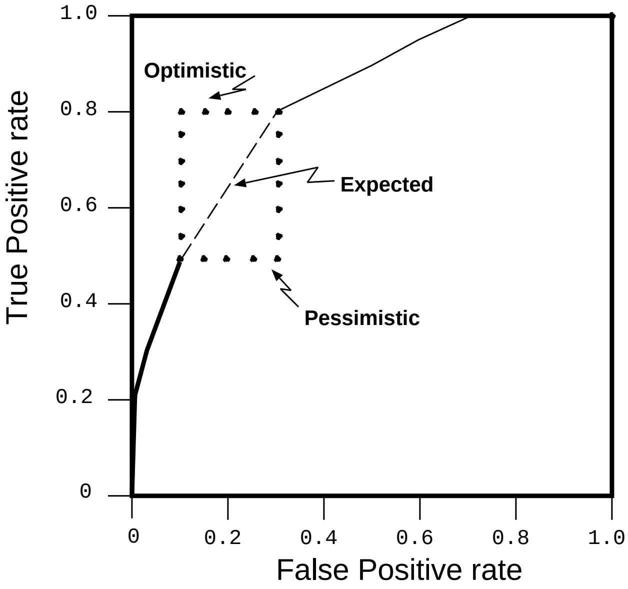


Figure 6: The optimistic, pessimistic and expected ROC segments resulting from a sequence of ten equally scored instances.

Algorithm 3 Modifications to algorithm 2 to avoid introducing concavities.

subroutine ADD_POINT(P, R)
loop
    if $|R| < 2$ **then**
        push P onto R
        return
    else
        $T \Leftarrow pop(R)$
        $T2 \Leftarrow top\_of\_stack(R)$
        if SLOPE$(T2, T) <$ SLOPE(T, P) **then**
            push T onto R
            push P onto R
            return
end subroutine

Instances that are scored equally may seem unusual but with some classifier models they are common. For example, if we use instance counts at nodes in a decision tree to score instances, a high-entropy leaf node may produce many equally scored instances. If such instances are not averaged, the resulting ROC curves will be sensitive to the test set ordering, and different orderings can yield very misleading curves. This can be especially critical in calculating the area under an ROC curve, discussed in section 6. Consider a decision tree containing a leaf node accounting for n positives and m negatives. Every instance that is classified to this leaf node will be assigned the same score. The rectangle of figure 6 will be of size $\frac{nm}{PN}$, and if these instances are not averaged this one leaf may account for errors in ROC curve area as high as $\frac{nm}{2PN}$.

5.2 Creating convex ROC curves

For proper testing purposes, the ROC algorithm 2 should be used since it calculates an unmodified ROC curve.

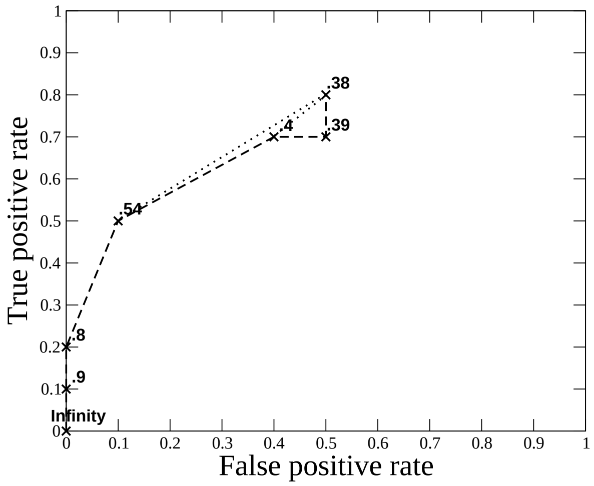


Figure 7: The points of figure 3 being processed by algorithm 3.

The curves it produces will depict expected classifier performance on the test set.

However, for some purposes we may want to generate a *convex* ROC curve, *i.e.*, one with all concavities removed. A concavity in an ROC curve represents a sub-optimality in the classifier[3]. In practice, concavities in ROC curves produced by learned classifiers may be due either to idiosyncracies in learning or to small test set effects.[4]

Algorithm 2 may be modified to remove concavities simply by replacing the ADD_POINT subroutine. The new definition is shown in algorithm 3. In the new definition of ADD_POINT, segment slopes of R are examined prior to adding a new point. A convex ROC curve should have monotonically decreasing slopes. If adding a new point would introduce a line segment of greater slope than the one preceeding it, the preceeding ROC point is discarded. The test is then repeated, popping and discarding points as necessary from R until a segment of greater slope is reached (or until the curve has only a single point left). This procedure ensures monotonicity of slopes, and thus convexity of the curve.

Algorithm 3 can be optimized by caching line segment slopes along with R points. Doing so avoids repeated unnecessary calls to the SLOPE function. For clarity, the optimization is not shown.

Algorithm 3 is computationally no more complex than algorithm 2. A point is only added in one place, at line 11 (the push in line 10 only serves to reverse the pop of line 7). A point can be popped at most once. The computational complexity remains $O(n \log n)$.

Figure 7 shows the points of figure 3 processed by algorithm 3 rather than by algorithm 2. Figure 7a shows an enlargement of an intermediate step, in which the point of score 0.4 is added to the curve. Slopes back to point .505, .51 and .54 are considered in turn, and points .505 and .51 are removed.

[3]Specifically, a concavity occurs whenever a segment of slope r is joined at the right to a segment of slope s where $s > r$. The slope of an ROC curve represents the class likelihood ratio. A concavity indicates that the group of instances producing s have a higher posterior class ratio than those accounting for r. Because s occurs to the right of r, r's instances should have been ranked more highly than s's, but were not. This is a sub-optimality of the classifier.

[4]Bradley's (1997) ROC curves exhibit noticeable concavities, as do the Breast cancer and RoadGrass domains of Provost *et al.* (1998).

Algorithm 4 Calculating the area under an ROC curve

Inputs: L, the set of test instances; $f(i)$, the probabilistic classifier's estimate that instance i is positive.
Outputs: A, the area under the ROC curve.

```
L_sorted ⇐ L sorted decreasing by f scores
FP ⇐ TP ⇐ 0
FP_prev ⇐ TP_prev ⇐ 0
A ⇐ 0
f_prev ⇐ −∞
for i ∈ L_sorted do
  if f(i) ≠ f_prev then
    A ⇐ A + TRAP_AREA(FP, FP_prev, TP, TP_prev)          /* See A.3 for TRAP_AREA */
    f_prev ⇐ f(i)
    FP_prev ⇐ FP
    TP_prev ⇐ TP
  if i is a positive example then
    TP ⇐ TP + 1
  else
    FP ⇐ FP + 1
A ⇐ A + TRAP_AREA(1, FP_prev, 1, TP_prev)
A ⇐ A/(P · N)                          /* scale from P × N onto the unit square */
end
```

6 Area under an ROC Curve (AUC)

An ROC curve is a two-dimensional depiction of classifier performance. To compare classifiers we may want to reduce ROC performance to a single scalar value representing expected performance. A common method is to calculate the area under the ROC curve, abbreviated **AUC** (Bradley, 1997; Hanley & 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.

The 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 (Hanley & McNeil, 1982). The AUC is also closely related to the Gini index (Breiman, Friedman, Olshen, & Stone, 1984), which is twice the area between the diagonal and the ROC curve. Hand and Till (2001) point out that $\text{Gini} + 1 = 2 \times \text{AUC}$.

Figure 8a shows the areas under two ROC curves, A and B. Classifier B has greater area and therefore better average performance. Figure 8b 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.

It is possible for a high-AUC classifier to perform worse in a specific region of ROC space than a low-AUC classifier. Figure 2a shows an example of this: classifier B is generally better than A except at FPrate > 0.6 where A has a slight advantage. But in practice the AUC performs very well and is often used when a general measure of predictiveness is desired.

The AUC may be computed easily using a small modification of algorithm 2, shown in algorithm 4. Instead of

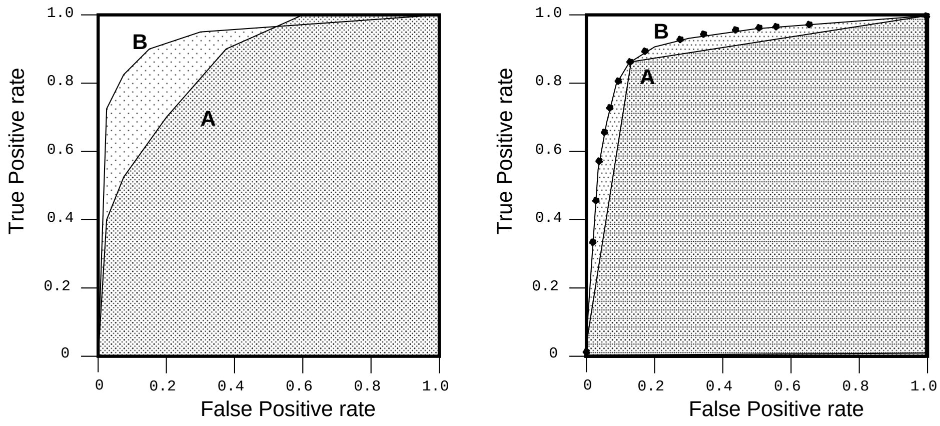


Figure 8: Two ROC graphs. The graph on the left shows the area under two ROC curves. The graph on the right shows the area under the curves of a discrete classifier (A) and a probabilistic classifier (B).

collecting ROC points, the algorithm adds successive areas of trapezoids to *Area*. Finally, it divides by the total possible area to scale the value to the unit square.

7 Averaging ROC curves

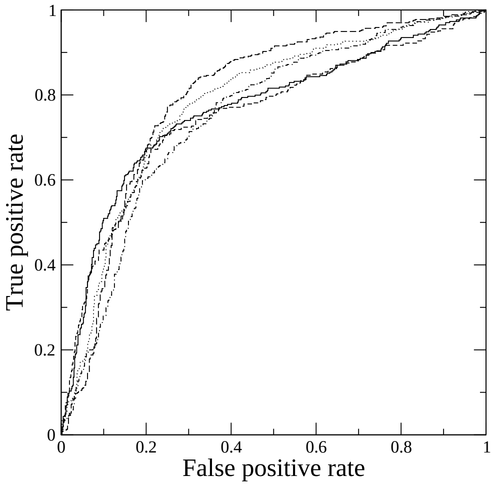
Although ROC curves may be used to evaluate classifiers, care should be taken when using them to make conclusions about classifier superiority. Some researchers have assumed that an ROC graph may be used to select the best classifiers simply by graphing them in ROC space and seeing which ones dominate. This is misleading; it is analogous to taking the maximum of a set of accuracy figures from a single test set. Without a measure of variance we cannot easily compare the classifiers.

Averaging ROC curves is easy if the original instances are available. Given test sets $T_1, T_2, \cdots, T_n$, generated from cross-validation or the bootstrap method, we can simply merge sort the instances together by their assigned scores into one large test set T_M. We then run an ROC curve generation algorithm such as algorithm 2 on T_M and plot the result. This yields the average expected ROC performance.

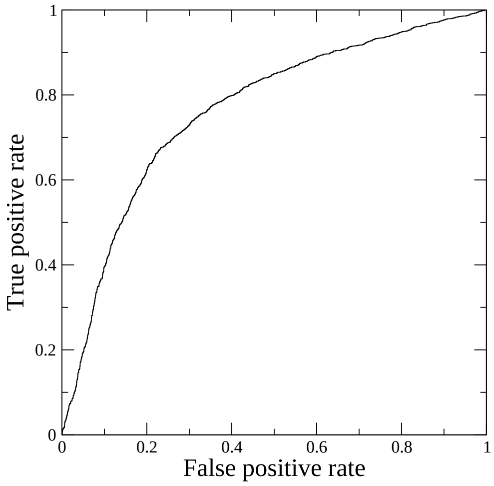
However, the primary reason for using multiple test sets is to derive a measure of variance, which this simple merging does not provide. For this we need a more sophisticated method that samples the individual curves at different points.

ROC space is two-dimensional, and any average is necessarily one-dimensional. ROC curves can be projected onto a single dimension and averaged conventionally, but this leads to the question of whether the projection is appropriate, or more precisely, whether it preserves characteristics of interest. The answer depends upon the reason for averaging the curves. This section presents two methods for averaging ROC curves: vertical and threshold averaging.

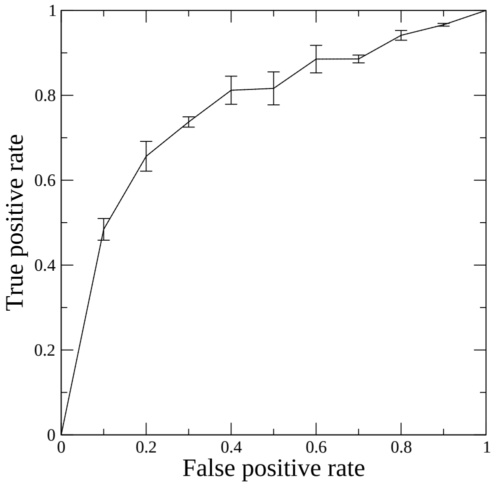
Figure 9a shows five ROC curves to be averaged. Each contains a thousand points and has some concavities. Figure 9b shows the curve formed by merging the five test sets and computing their combined ROC curve. Figures 9b and 9c show average curves formed by sampling the five individual ROC curves. The error bars are 95% confidence intervals.



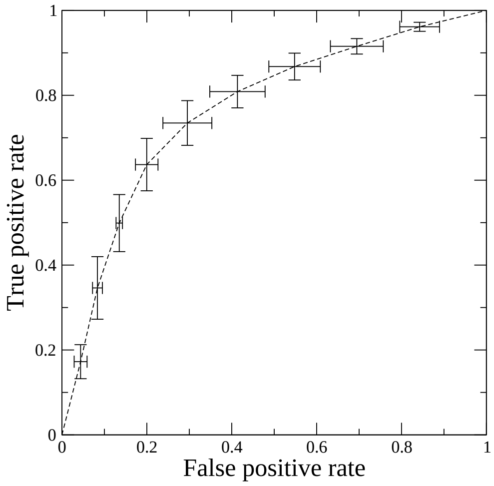
(a) ROC curves of five instance samples



(b) ROC curve formed by merging the five samples



(c) The curves of a averaged vertically



(d) The curves of a averaged by threshold

Figure 9: ROC curve averaging

7.1 Vertical averaging

Vertical averaging takes vertical samples of the ROC curves for fixed FP rates and averages the corresponding TP rates. Such averaging is appropriate when the FP rate can indeed be fixed by the researcher, or when a single-dimensional measure of variation is desired. Provost, Fawcett and Kohavi (1998) used this method in their work of averaging ROC curves of a classifier for k-fold cross-validation.

In this method each ROC curve is treated as a function, R_i, such that $TP = R_i(FP)$. This is done by choosing the maximum TP for each FP and interpolating between points when necessary. The averaged ROC curve is the function $\hat{R}(FP) = mean[R_i(FP)]$. To plot an average ROC curve we can sample from $\hat{R}$ at points regularly spaced along the FP-axis. Confidence intervals of the mean of TP are computed using the common assumption of a binomial distribution.

Algorithm 5 computes this vertical average of a set of ROC points. It leaves the means in the array $TPavg$.

Algorithm 5 Vertical averaging of ROC curves.

Inputs: *samples*, the number of FP samples; *nrocs*, the number of ROC curves to be sampled, $ROCS[nrocs]$, an array of *nrocs* ROC curves; $npts[m]$, the number of points in ROC curve m. Each ROC point is a structure of two members, FP and TP, whose values are referenced by subscripts here.

Output: Array $TPavg$, containing the vertical (TP) averages.

```
s ⇐ 1
for FP_sample = 0 to 1 by 1/samples do
  TPsum ⇐ 0
  for i = 1 to nrocs do
    TPsum ⇐ TPsum + TP_FOR_FP(FP_sample, ROCS[i], npts[i])
  TPavg[s] ⇐ TPsum/i
  s ⇐ s + 1
end
```

```
function TP_FOR_FP(FP_sample, ROC, npts)
i ⇐ 1
while i < npts and ROC[i+1]_FP ≤ FP_sample do
  i ⇐ i + 1
if ROC[i]_FP = FP_sample then
  return ROC[i]_TP
else if ROC[i]_FP < FP_sample then
  return INTERPOLATE(ROC[i], ROC[i+1], FP_sample)
end function
```

Several extensions have been left out of this algorithm for clarity. The algorithm may easily be extended to compute standard deviations of the samples in order to draw confidence bars. Also, the function TP_FOR_FP may be optimized somwhat. Because it is only called on monotonically increasing values of FP, it need not scan each ROC array from the beginning every time; it could keep a record of the last point seen and initialize i from this array.

Figure 9c shows the vertical average of the five curves in figure 9a. The vertical bars on the curve show the 95% confidence region of the ROC mean. For this average curve, the curves were sampled at FP rates from 0 through 1 by 0.1. It is possible to sample curves much more finely but the confidence bars may become difficult to read.

7.2 Threshold averaging

Vertical averaging has the advantage that averages are made of a single dependent variable (the false positive rate). This simplifies computing confidence intervals. However, Holte (2002) has pointed out that false positive rate is often not under the direct control of the researcher. It may be preferable to average ROC points using an independent variable whose value can be controlled directly, such as the threshold on the classifier scores.

Threshold averaging accomplishes this. Instead of sampling points based on their positions in ROC space, as vertical averaging does, it samples based on the thresholds that produced these points. The method must generate a set of thresholds to sample, then for each threshold it finds the corresponding point of each ROC curve and averages them.

Algorithm 6 shows the basic method for doing this. It generates an array T of classifier scores which are sorted from largest to smallest and used as the set of thresholds. These thresholds are sampled at fixed intervals determined by *samples*, the number of samples desired. For a given threshold, the algorithm selects from each ROC curve the

Algorithm 6 Threshold averaging of ROC curves.

Inputs: *samples*, the number of threshold samples; *nrocs*, the number of ROC curves to be sampled; $ROCS[nrocs]$, an array of *nrocs* ROC curves; $npts[m]$, the number of points in ROC curve m. Each ROC point is a structure of three members, FP, TP and Score, whose values are referenced by subscripts here.
Output: *Avg*, an array of (X,Y) points constituting the average ROC curve.

```
T ⇐ all Scores of all ROC points
sort T in descending order
s ⇐ 1
for tidx = 1 to length(T) by int(length(T)/samples) do
   FPsum ⇐ 0
   TPsum ⇐ 0
   for i = 1 to nrocs do
      p ⇐ POINT_AT_THRESH(ROCS[i], npts[i], T[tidx])
      FPsum ⇐ FPsum + p_FP
      TPsum ⇐ TPsum + p_TP
   Avg[s] ⇐ (FPsum/i , TPsum/i)
   s ⇐ s + 1
end
function POINT_AT_THRESH(ROC, npts, thresh)
i ⇐ 1
while i < npts and ROC[i]_Score > thresh do
   i ⇐ i + 1
return ROC[i]
end function
```

the point of greatest score less than or equal to the threshold.[5] These points are then averaged separately along their X and Y axes, with the center point returned in the Avg array.

Figure 9d shows the result of averaging the five curves of 9a by thresholds. The resulting curve has average points and confidence bars in the X and Y directions. The bars shown are at the 95% confidence level.

There are some minor limitations of threshold averaging with respect to vertical averaging. To perform threshold averaging we need the classifier score information assigned to each point. Also, section 4.1 pointed out that classifier scores should not be compared across model classes. Because of this, ROC curves averaged from different model classes may be misleading because the scores may be incommensurate.

8 Additional Topics

The previous sections are intended to be self-contained and to cover the basic issues that arise in using ROC curves in data mining research. This section discusses additional, slightly more esoteric topics.

8.1 The ROC convex hull

One advantage of ROC graphs is that they enable visualizing and organizing classifier performance without regard to class distributions or error costs. 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

[5] We assume the ROC points have been generated by an algorithm like 2 that deals correctly with equally scored instances.

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.

Provost and Fawcett (1998, 2001) show that a set of operating conditions may be transformed easily into a so-called *iso-performance line* in ROC space. Specifically, the expected cost of applying the classifier represented by a point (FP,TP) in ROC space is:

$$p(\mathbf{p}) \cdot (1 - TP) \cdot c(\mathbf{N}, \mathbf{p}) \ + \ p(\mathbf{n}) \cdot FP \cdot c(\mathbf{Y}, \mathbf{n})$$

Therefore, two points, (FP_1,TP_1) and (FP_2,TP_2), have the same performance if

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

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 details are beyond the scope of this article, but more generally a classifier is potentially optimal if and only if it lies on the convex hull (Barber, Dobkin, & Huhdanpaa, 1993) 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. Since only the classifiers on the convex hull are potentially optimal, no others need be retained. The operating conditions of the classifier may be translated into an iso-performance line, which in turn may be used to identify a portion of the ROCCH. As conditions change, the hull itself does not change; only the portion of interest will.

8.2 Decision problems with more than two classes

Discussions up to this point have dealt with only two classes, and much of the ROC literature maintains this assumption. ROC analysis is commonly employed in medical decision making in which two-class diagnostic problems—presence or absence of an abnormal condition—are common. The two axes represent tradeoffs between errors (false positives) and benefits (true positives) that a classifier makes between two classes. Much of the analysis is straightforward because of the symmetry that exists in the two-class problem. The resulting performance can be graphed in two dimensions, which is easy to visualize.

8.2.1 Multi-class ROC graphs

With more than two classes the situation becomes much more complex if the entire space is to be managed. With n classes the confusion matrix becomes an $n \times n$ matrix containing the n correct classifications (the major diagonal entries) and $n^2 - n$ possible errors (the off-diagonal entries). Instead of managing trade-offs between TP and FP, we have n benefits and $n^2 - n$ errors. With only three classes, the surface becomes a $3^2 - 3 = 6$-dimensional polytope. Lane (2000) has written a short paper outlining the issues involved and the prospects for addressing them. Srinivasan

(1999) has shown that the analysis behind the ROC convex hull extends to multiple classes and multi-dimensional convex hulls.

One method for handling n classes is to produce n different ROC graphs, one for each class. This is called the **class reference** formulation. Specifically, if C is the set of all classes, ROC graph i plots the classification performance using class c_i as the positive class and all other classes as the negative class, *i.e.*,

$$
\begin{aligned}
P_i &= c_i & (1)\\
N_i &= \bigcup_{j \neq i} c_j \in C & (2)
\end{aligned}
$$

While this is a convenient formulation, it compromises one of the attractions of ROC graphs, namely that they are insensitive to class skew (see section 4.2). Because each N_i comprises the union of $n-1$ classes, changes in prevalence within these classes may alter the c_i's ROC graph. For example, assume that some class $c_k \in N$ is particularly easy to identify. A classifier for class $c_i, i \neq k$ may exploit some characteristic of c_k in order to produce low scores for c_k instances. Increasing the prevalence of c_k might alter the performance of the classifier, and would be tantamount to changing the target concept by increasing the prevalence of one of its disjuncts. This in turn would alter the ROC curve. However, with this caveat, this method can work well in practice and provide reasonable flexibility in evaluation.

8.2.2 Multi-class AUC

The AUC is a measure of the discriminability of a pair of classes. In a two-class problem, the AUC is a single scalar value, but a multi-class problem introduces the issue of combining multiple pairwise discriminability values. The reader is referred to Hand and Till's (2001) article for an excellent discussion of these issues.

One approach to calculating multi-class AUCs was taken by Provost and Domingos (2001) in their work on probability estimation trees. They calculated AUCs for multi-class problems by generating each class reference ROC curve in turn, measuring the area under the curve, then summing the AUCs weighted by the reference class's prevalence in the data. More precisely, they define:

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

where $AUC(c_i)$ is the area under the class reference ROC curve for c_i, as in equations 2. This definition requires only $|C|$ AUC calculations, so its overall complexity is $O(|C| n \log n)$.

The advantage of Provost and Domingos's AUC formulation is that AUC_{total} is generated directly from class reference ROC curves, and these curves can be generated and visualized easily. The disadvantage is that the class reference ROC is sensitive to class distributions and error costs, so this formulation of AUC_{total} is as well.

Hand and Till (2001) take a different approach in their derivation of a multi-class generalization of the AUC. They desired a measure that is insensitive to class distribution and error costs. The derivation is too detailed to summarize here, but it is based upon the fact that the AUC is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. From this probabilistic

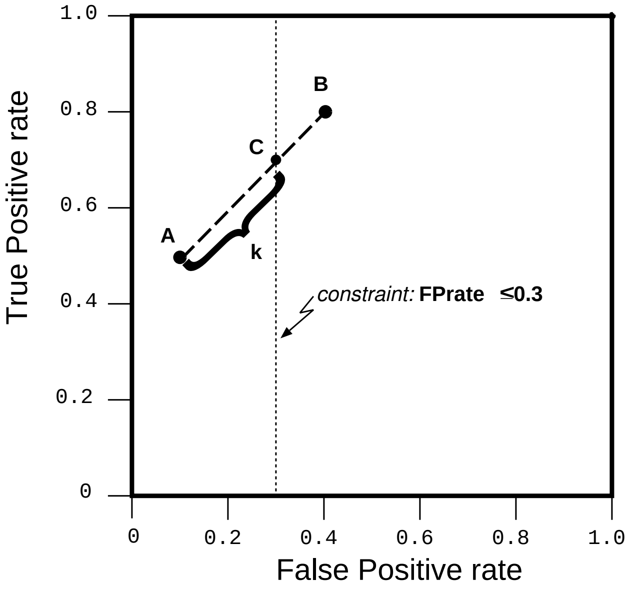


Figure 10: Interpolating classifiers

form, they derive a formulation that measures the unweighted *pairwise* discriminability of classes. Their measure, which they call M, is equivalent to:

$$AUC_{total} = \frac{2}{|C|(|C|-1)} \sum_{\{c_i,c_j\}\in C} AUC(c_i, c_j)$$

where n is the number of classes and $AUC(c_i, c_j)$ is the area under the two-class ROC curve involving classes c_i and c_j. The summation is calculated over all pairs of distinct classes, irrespective of order. There are $|C|(|C|-1)/2$ such pairs, so the time complexity of their measure is $O(|C|^2\, n \log n)$. While Hand and Till's formulation is well justified and is insensitive to changes in class distribution, there is no easy way to visualize the surface whose area is being calculated.

8.3 Combining classifiers

While ROC curves are commonly used for visualizing and evaluating individual classifiers, ROC space can also be used to estimate the performance of combinations of classifiers.

8.3.1 Interpolating classifiers

It is sometimes the case that the performance desired of a classifier is not exactly represented by any available classifier, but lies between two classifiers. The desired classification performance can be obtained by sampling the decisions of each classifier. The sampling ratio will determine where the resulting classification performance lies.

Figure 10 illustrates this idea. Assume we have generated two classifiers, A and B, whose performance is shown. A has a FP rate of 0.1 and B has an FP rate of 0.4. Assume we require a classifier with an FP rate no greater than 0.3 (the vertical dotted line in the figure). One solution is simply to choose A, since B does not satisfy the criterion.

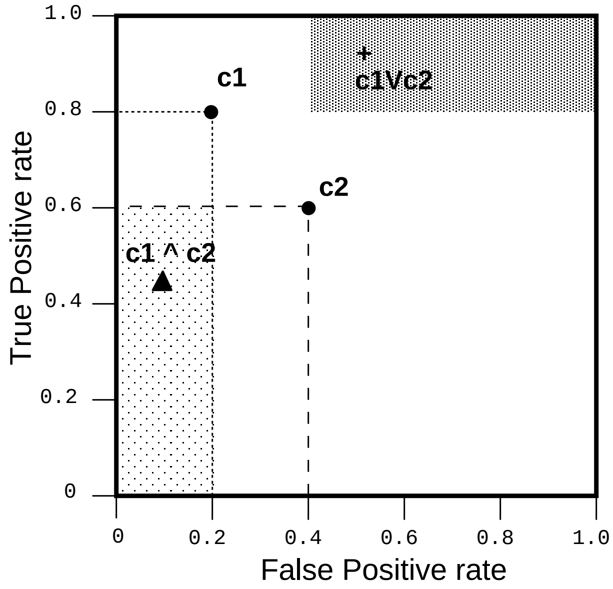


Figure 11: The expected positions of boolean combinations of c_1 and c_2. c_1 lies at $(0.2, 0.8)$ and c_2 lies at $(0.4, 0.6)$. The conjunction $c_1 \wedge c_2$ will lie somewhere in the shaded box at lower left. Under conditions of classifier independence, the conjunction will lie at the point $(0.08, 0.42)$. The disjuncton $c_1 \vee c_2$ will lie somewhere in the shaded box at upper right. Under conditions of classifier independence, the disjunction will lie at the point $(0.45, 9.0)$.

However, we can do better—we can generate a classifier C by interpolating between A and B. For the desired FP rate of 0.3, linear interpolation gives:

$$k = \frac{0.3 - 0.1}{0.4 - 0.1} = 2/3$$

If we sample B's decisions at a rate of 2/3 and A's decisions at a rate of $1 - 2/3 = 1/3$, we should attain C's classification performance. In practice this fractional sampling can be done by randomly sampling decision from each: for each instance, generate a random number between zero and one. If the random number is greater than k, apply classifier A to the instance and report its decision, else pass the instance to B.

8.3.2 Logically combining classifiers

With two classes, a classifier c can be viewed as a predicate on an instance I where $c(I)$ is true iff $c(I) = \mathbf{Y}$. We can then speak of boolean combinations of classifiers, and an ROC graph can provide a way of visualizing the performance of such combinations. It can help to illustrate both the bounding region of the new classifier and its expected position.

If two classifiers c_1 and c_2 are conjoined to create $c_3 = c_1 \wedge c_2$, where will c_3 lie in ROC space? Let TPrate_3 and FPrate_3 be the ROC positions of c_3. The minimum number of instances c_3 can match is zero. The maximum is limited by the intersection of their positive sets. Since a new instance must satisfy both c_1 and c_2, we can bound c_3's position by:

$$0 \leq \quad \text{TPrate}_3 \quad \leq min(\text{TPrate}_1, \text{TPrate}_2)$$

$$0 \leq \quad \text{TPrate}_3 \quad \leq min(\text{FPrate}_1, \text{FPrate}_2)$$

Figure 11 shows this bounding rectangle for two classifiers $c_1 \wedge c_2$, the shaded rectangle in the lower left corner. Where within this rectangle do we expect c_3 to lie? Let x be an instance in the true positive set TP_3 of c_3. Then:

$$\begin{aligned} TPrate_3 &\approx p(x \in TP_3) \\ &\approx p(x \in TP_1 \wedge x \in TP_2) \end{aligned}$$

By assuming independence of c_1 and c_2, we can continue:

$$\begin{aligned} TPrate_3 &\approx p(x \in TP_1) \cdot p(x \in TP_2) \\ &\approx \frac{\mid TP_1 \mid}{\mid P \mid} \cdot \frac{\mid TP_2 \mid}{\mid P \mid} \\ &\approx TPrate_1 \cdot TPrate2 \end{aligned}$$

A similar derivation can be done for FPrate_3, showing that $\text{FPrate}_3 = \text{FPrate}_1 \cdot \text{FPrate}_2$. Thus, the conjunction of two classifiers c_1 and c_2 can be expected to lie at the point

$$(\text{FPrate}_1 \cdot \text{FPrate}_2 \ , \ \text{TPrate}_1 \cdot \text{TPrate}_2)$$

in ROC space. This point is shown as the triangle in figure 11 at $(0.08, 0.42)$. This estimate assumes independence of classifiers; interactions between c_1 and c_2 may cause the position of c_3 in ROC space to vary from this estimate.

We can derive similar expressions for the disjunction $c_4 = c_1 \vee c_2$. In this case the rates are bounded by:

$$\begin{aligned} max(\text{TPrate}_1, \text{TPrate}_2) \leq \quad & \text{TPrate}_4 \quad \leq min(1, \text{TPrate}_1 + \text{TPrate2}) \\ max(\text{FPrate}_1, \text{FPrate}_2) \leq \quad & \text{FPrate}_4 \quad \leq min(1, \text{FPrate}_1 + \text{FPrate2}) \end{aligned}$$

This bounding region is indicated in figure 11 by the shaded rectangle in the upper right portion of the ROC graph. The expected position, assuming independence, is:

$$\begin{aligned} \text{TPrate}_4 &= 1 - [1 - \text{TPrate}_1 - \text{TPrate}_2 + \text{TPrate}_1 \cdot \text{TPrate}_2] \\ \text{FPrate}_4 &= 1 - [1 - \text{FPrate}_1 - \text{FPrate}_2 + \text{FPrate}_1 \cdot \text{FPrate}_2] \end{aligned}$$

This point is indicted by the marked + symbol within the bounding rectangle.

These equations allow limited visualization of the results of classifier combinations in ROC space. They could also be used to direct an induction or feature construction algorithm. For example, by calculating the expected position of each combination and comparing it with the ROC convex hull, a method could filter out unpromising new classifiers before they are generated.

8.3.3 Chaining classifiers

Section 3 mentioned that classifiers on the left side of an ROC graph near $X = 0$ may be thought of as "conservative"; and classifiers on the upper side of an ROC graph near $Y = 1$ may be thought of as "liberal". With this interpretation it might be tempting to devise a composite scheme that applies classifiers sequentially like a rule list. Such a technique might work as follows: Given the classifiers on the ROC convex hull, an instance is first given to the most conservative (left-most) classifier. If that classifier returns **Y**, the composite classifier returns **Y**; otherwise, the second most conservative classifier is tested, and so on. The sequence terminates when some classifier issues a **Y** classification, or when the classifiers reach a maximum expected cost, such as may be specified by an iso-performance line. The resulting classifier is $c_1 \vee c_2 \vee \cdots \vee c_k$, where c_k has the highest expected cost tolerable.

Unfortunately, this chaining of classifiers may not work as desired. Classifiers' positions in ROC space are based upon their *independent* performance. When classifiers are applied in sequence this way, they are not being used independently but are instead being applied to instances which more conservative classifiers have already classified as negative. Due to classifier interactions (intersections among classifiers' TP and FP sets), the resulting classifier may have very different performance characteristics than any of the component classifiers. Although section 8.3.2 introduced an independence assumption that may be reasonable for combining two classifiers, this assumption becomes much less tenable as longer chains of classifiers are constructed.

8.3.4 The importance of final validation

To close this section on classifier combination, we emphasize a basic point that is easy to forget. ROC graphs are commonly used in evaluation, and are generated from a final test set. If an ROC graph is instead used to select or to combine classifiers, this use must be considered to be part of the training phase. A separate held-out validation set must be used to estimate the expected performance of the classifier(s). This is true even if the ROC curves are being used to form a convex hull.

8.4 Alternatives to ROC graphs

Recently, various alternatives to ROC graphs have been proposed. We briefly summarize them here.

8.4.1 DET curves

DET graphs (Martin, Doddington, Kamm, Ordowski, & Przybocki, 1997) are not so much an alternative to ROC curves as an alternative way of presenting them. There are two differences. First, DET graphs plot false negatives on the Y axis instead of true positives, so they plot one kind of error against another. Second, DET graphs are log scaled on both axes so that the area of the lower left part of the curve (which corresponds to the upper left portion of an ROC graph) is expanded. Martin et al. (1997) argue that well-performing classifiers, with low false positive rates and/or low false negative rates, tend to be "bunched up" together in the lower left portion of a ROC graph. The log scaling of a DET graph gives this region greater surface area and allows these classifiers to be compared more easily.

8.4.2 Cost curves

Section 8.1 showed how information about class proportions and error costs could be combined to define the slope of a so-called iso-performance line. Such a line can be placed on an ROC curve and used to identify which classifier(s) perform best under the conditions of interest. In many cost minimization scenarios, this requires inspecting the curves and judging the tangent angles for which one classifier dominates.

Drummond and Holte (2000, 2002) point out that reading slope angles from an ROC curve may be difficult to do. Determining the regions of superiority, and the amount by which one classifier is superior to another, is challenging when the comparison lines are curve tangents rather than simple vertical lines. Drummond and Holte reason that if the primary use of a curve is to compare relative costs, the graphs should represent these costs explicitly. They propose *cost curves* as an alternative to ROC curves.

On a cost curve, the X axis ranges from 0 to 1 and measures the proportion of positives in the distribution. The Y axis, also from 0 to 1, is the relative expected misclassification cost. A perfect classifier is a horizontal line from $(0, 0)$ to $(0, 1)$. Cost curves are a point-line dual of ROC curves: a point (*i.e.*, a discrete classifier) in ROC space is represented by a line in cost space, with the line designating the relative expected cost of the classifier. For any X point, the corresponding Y points represent the expected costs of the classifiers. Thus, while in ROC space the convex hull contains the set of lowest-cost classifiers, in cost space the lower envelope represents this set.

8.4.3 Relative superiority graphs and the LC index

Like cost curves, the LC index (Adams & Hand, 1999) is a transformation of ROC curves that facilitates comparing classifiers by cost. Adams and Hand argue that precise cost information is rare, but *some* information about costs is always available, and so the AUC is too coarse of a measure of classifier performance. An expert may not be able to specify exactly what the costs of a false positive and false negative should be, but an expert usually has some idea how much more expensive one error is than another. This can be expressed as a range of values in which the error cost ratio will lie.

Adams and Hand's method maps the ratio of error costs onto the interval (0,1). It then transforms a set of ROC curves into a set of parallel lines showing which classifier dominates at which region in the interval. An expert provides a sub-range of (0,1) within which the ratio is expected to fall, as well as a most likely value for the ratio. This serves to focus attention on the interval of interest. Upon these "relative superiority" graphs a measure of confidence—the LC index—can be defined indicating how likely it is that one classifier is superior to another within this interval.

The relative superiority graphs may be seen as a binary version of cost curves, in which we are only interested in which classifier is superior. The LC index (for loss comparison) is thus a measure of confidence of superiority rather than of cost difference.

9 Conclusion

ROC graphs are a very useful tool for visualizing and evaluating classifiers. They are able to provide a richer measure of classification performance than accuracy or error rate can, and they have advantages over other evaluation measures such as precision-recall graphs and lift curves. However, as with any evaluation metric, using them wisely requires knowing their characteristics and limitations. It is hoped that this article advances the general knowledge about ROC graphs and helps to promote better evaluation practices in the data mining community.

A Basic geometry functions

The algorithms in this article make use the following basic functions from geometry.

A.1 Slope of a line

The slope of a line is defined as the ratio of the change in y to the change in x, or the "rise over run". This is straightforward to compute except for detecting infinite slopes.

function SLOPE(P, Q)
if $P_x = Q_x$ **then**
  return ∞
else
  return $\frac{P_y - Q_y}{P_x - Q_x}$
end function

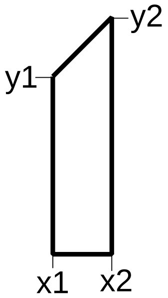
A.2 Linear interpolation between two points

Given two points, P1 and P2, and an X value, interpolate the Y value corresponding to X. It is guaranteed that X is between P1 and P2 and that $P1_X \neq P2_X$.

function INTERPOLATE$(P1, P2, X)$
$\Delta x = P2_X - P1_X$
$\Delta y = P2_Y - P1_Y$
$m = \Delta y / \Delta x$
return $P1_Y + m \cdot (X - P1_X)$
end function

A.3 Area of a trapezoid

The area of a trapezoid is simply the average height times the width of the base.



function TRAP_AREA$(X1, X2, Y1, Y2)$
$Base \Leftarrow |X1 - X2|$
$Height_{avg} \Leftarrow (Y1 + Y2)/2$
return $Base \times Height_{avg}$
end function

Acknowledgements

While at Bell Atlantic, Foster Provost and I investigated ROC graphs and ROC analysis for use in real-world domains. My understanding of ROC analysis has benefited from numerous discussions with him.

I am indebted to Rob Holte and Chris Drummond for many enlightening email exchanges on ROC graphs, especially on the topics of cost curves and averaging ROC curves. These discussions increased my understanding of the complexity of the issues involved. I wish to thank Terran Lane, David Hand and José Hernandez-Orallo for discussions clarifying their work. I wish to thank Kelly Zou and Holly Jimison for pointers to relevant articles in the medical decision making literature. Of course, any misunderstandings or errors in this article are my own responsibility.

Much open source software was used in this work. I wish to thank the authors and maintainers of XEmacs, TeXand LaTeX, Perl and its many user-contributed packages, and the Free Software Foundation's GNU Project. The figures in this paper were created with Tgif, Grace and Gnuplot.

References

Adams, N. M., & Hand, D. J. (1999). Comparing classifiers when the misallocations costs are uncertain. *Pattern Recognition*, *32*, 1139–1147.

Barber, C., Dobkin, D., & Huhdanpaa, H. (1993). The quickhull algorithm for convex hull. Tech. rep. GCG53, University of Minnesota. Available: `ftp://geom.umn.edu/pub/software/qhull.tar.Z`.

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

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Wadsworth International Group, Belmont, CA.

Clearwater, S., & Stern, E. (1991). A rule-learning program in high energy physics event classification. *Comp Physics Comm*, *67*, 159–182.

Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 155–164.

Dreiseitl, S., Ohno-Machado, L., & Binder, M. (2000). Comparing three-class diagnostic tests by three-way ROC analysis. *Medical Decision Making*, *20*, 323–331.

Drummond, C., & Holte, R. C. (2000). Explicitly representing expected cost: An alternative to ROC representation. In Ramakrishnan, R., & Stolfo, S. (Eds.), *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 198–207. ACM Press.

Drummond, C., & Holte, R. C. (2002). Classifier cost curves: Making performance evaluation easier and more informative. Unpublished manuscript available from the authors.

Egan, J. P. (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., & Provost, F. (1996). Combining data mining and machine learning for effective user profiling. In Simoudis, Han, & Fayyad (Eds.), *Proceedings on the Second International Conference on Knowledge Discovery and Data Mining*, pp. 8–13 Menlo Park, CA. AAAI Press.

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

Forman, G. (2002). A method for discovering the insignificance of one's best classifier and the unlearnability of a classification task. In Lavrac, Motoda, & Fawcett (Eds.), *Proceedings of th First International Workshop on Data Mining Lessons Learned (DMLL-2002)*. Available: `http://www.hpl.hp.com/personal/Tom_Fawcett/DMLL-2002/Forman.pdf`.

Hand, D. J., & Till, R. J. (2001). A simple generalization of the area under the ROC curve to multiple class classification problems. *Machine Learning*, *45*(2), 171–186.

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

Holte, R. (2002). Personal communication..

Kubat, M., Holte, R. C., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, *30*, xxx–yyy.

Lane, T. (2000). Extensions of ROC analysis to multi-class domains. In Dietterich, T., Margineantu, D., Provost, F., & Turney, P. (Eds.), *ICML-2000 Workshop on Cost-Sensitive Learning*.

Lewis, D. (1990). Representation quality in text classification: An introduction and experiment. In *Proceedings of Workshop on Speech and Natural Language*, pp. 288–295 Hidden Valley, PA. Morgan Kaufmann.

Lewis, D. (1991). Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop*, pp. 312–318. Morgan Kaufmann.

Martin, A., Doddington, G., Kamm, T., Ordowski, M., & Przybocki, M. (1997). The DET curve in assessment of detection task performance. In *Proc. Eurospeech '97*, pp. 1895–1898 Rhodes, Greece.

Mossman, D. (1999). Three-way ROCs. *Medical Decision Making*, *19*, 78–89.

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 the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pp. 706–713 Menlo Park, CA. AAAI Press. Available: `http://www.purl.org/NET/tfawcett/papers/aaai98-dist.ps.gz`.

Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, *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 the Fifteenth International Conference on Machine Learning*, pp. 445–453 San Francisco, CA. Morgan Kaufmann.

Saitta, L., & Neri, F. (1998). Learning in the "real world". *Machine Learning*, *30*, 133–163.

Spackman, K. A. (1989). Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pp. 160–163 San Mateo, CA. Morgan Kaufman.

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.

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

Swets, J. A., Dawes, R. M., & Monahan, J. (2000a). Better decisions through science. *Scientific American*, *283*, 82–87. Available: `http://www.psychologicalscience.org/newsresearch/publications/journals/%siam.pdf`.

Swets, J. A., Dawes, R. M., & Monahan, J. (2000b). Psychological science can improve diagnostic decisions. *Psychological Science in the Public Interest*, *1*(1), 1–26.

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*.

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