ROC Curves and Other Ways to Evaluate Classifiers
Duke Course Notes
Cynthia Rudin

The misclassification error doesn’t provide as much detail as we might want about our classifiers. Receiver Operating Characteristic (ROC) Curves provide much more detail. Let us build up to their definition. There are two types of ROC curves. One is a property of a model, the other is a property of an algorithm. Before I explain ROC curves, I need to tell you about typical evaluation functions for machine learning that trade off false positives with false negatives.

Confusion Matrices
Let’s take a binary classifier, which predicts either \( \hat{y} = -1 \) or \( \hat{y} = 1 \). (Typically that notation indicates predictions, such as \( \hat{y} \).) Its confusion matrix consists of four numbers in a table. The definition is on the left below, and I put an example from a dataset on the right.

<table>
<thead>
<tr>
<th></th>
<th>( y = +1 )</th>
<th>( y = -1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{y} = 1 )</td>
<td>TP</td>
<td>FP (Type I error)</td>
</tr>
<tr>
<td>( \hat{y} = 1 )</td>
<td>FN (Type II error)</td>
<td>TN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>( y = +1 )</th>
<th>( y = -1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{y} = 1 )</td>
<td>723</td>
<td>15</td>
</tr>
<tr>
<td>( \hat{y} = 1 )</td>
<td>72</td>
<td>409</td>
</tr>
</tbody>
</table>

Here, TP is the number of true positives (where \( y = 1 \) and \( \hat{y} = 1 \)) and TN is the number of true negatives (\( y = -1 \) and \( \hat{y} = -1 \)). False positives (FPs) occur when the classifier says that the point is positive but it’s not (\( y = -1 \) and \( \hat{y} = 1 \)). False negatives (FNs) occur when the classifier says that the point is negative but it’s not (\( y = 1 \) and \( \hat{y} = -1 \)). (The numbers in the table are just counts from a dataset.) The confusion matrix often has normalized values but I personally prefer to look at the counts themselves.

Sometimes it is more important to reduce false positives than false negatives or vice versa, depending on the specifics of the problem. For instance, FP might be more important than FN.

There are lots of evaluation measures for classifiers that use various parts of the confusion matrix.
• The misclassification error is \( \frac{\text{FP}+\text{FN}}{n} = \frac{1}{n} \sum_{i=1}^{n} 1[y_i \neq \hat{y}_i] \).

• True Positive Rate (TPR) = Sensitivity = Recall = \( \frac{\text{TP}}{\#\text{Positives}} \). In the context of information retrieval, where we want to retrieve relevant documents to a search query, the “recall” is the fraction of relevant documents our algorithm returned. My pneumonic device is that “recall” is the fraction of relevant documents our algorithm correctly recalled.

• True Negative Rate (TNR) = Specificity = \( \frac{\text{TN}}{\#\text{Negatives}} \). This is analogous to recall but for negative points.

• False positive rate (FPR) = \( \frac{\#\text{Negatives}}{\#\text{Positives}} \). The fraction of negatives we thought were positive.

• Precision = \( \frac{\text{TP}}{\#\text{Predicted Positive}} \). Precision is really useful for search engines. If the first page of search returns has 10 search results, the precision is how many of these 10 are relevant to the search query.

• F1-score is the harmonic balance between precision and recall.

\[
\text{F1} = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.
\]

The confusion matrix is only defined for binary classifiers. If you have a real-valued predictive model \( f \), it doesn’t correspond to a confusion matrix until you define a threshold \( \theta \), where if \( f(x) > \theta \), we predict positive, and otherwise negative. But where might you place this threshold? You could place it at \( f(x) = 0 \) for a trained classifier, but since we do not always value false positives the same as false negatives, this might not be a good choice.

ROC curves (as discussed next) would place the threshold in every possible position.

**ROC Curves for Functions**

ROC Curves are an excellent way to evaluate real-valued classifiers. I use ROC curves as a basic tool. (I would say that ROC curves and histograms are my data science bread and butter).

ROC curves started to be used during World War II for analyzing radar signals. After Pearl Harbor in 1941, the U.S. wanted to detect Japanese aircraft from
their radar signals. They measured the ability of radar receiver operators to
detect the Japanese planes using these curves, which they called the Receiver
Operating Characteristics. If they set the dial one way on the receiver, they got
many true positives, which are Japanese planes that were detected. But they
also got a lot of false positives, which were just bogus signals that were labeled
as Japanese planes. (This is a very high TPR and high FPR.) If they turned
the dial the other way, the detector would hardly ever predict positive (low FPR
but low TPR). To compare receivers, it is useful to look at the TPR and FPR
for each possible setting on the dial. We would like a high TPR for each possible
value of the FPR.

To create an ROC curve for a real valued function, think about the knob on the
receiver as being a threshold on the function. We compute the TPR and FPR
for each possible value of the threshold. Let us use an example.

\[
\begin{array}{c|cccccccc}
 f(x) & 15 & 12 & 10 & 8 & 6 & 2 & -1 & -3 & -14 & -20 \\
 y & - & + & + & - & + & - & - & - & + & - \\
\end{array}
\]

Let’s put the thresholds everywhere between each pair of points and at the ex-
tremes. It doesn’t matter exactly where we put the thresholds between each pair
of points, since it doesn’t affect TPR and FPR. Possible thresholds are: 16, 13.5,
11, 9, 7, 4, etc. At each of these values, we compute TPR and FPR. The de-
nominators are the number of positives (which is 4) and the number of negatives
(which is 6). For instance, if the threshold is at 9, then for \( f(x) > 9 \), there are
two positives (at 10 and 12) and a negative (at 15). That’s two true positives
and one false positive. So the TPR is 2/4 and the FPR is 1/6 for a threshold of
9. Here’s the full table.

\[
\begin{array}{ccccccccccccc}
\text{threshold} & 16 & 13 & 11 & 9 & 7 & 4 & 0 & -2 & -9 & -16 & -21 \\
\text{FPR} & 0/6 & 1/6 & 1/6 & 1/6 & 2/6 & 2/6 & 3/6 & 4/6 & 5/6 & 5/6 & 6/6 \\
\end{array}
\]

The ROC curve is the scatter plot of these points.
There is an intuitive way to think of the ROC plot. Put the labels \( y \) in order of \( f(x) \). We did that earlier, but I’ll put it here too: \(+ + + + + + + + + +\). To construct the ROC curve, go from left to right, starting with the bottom left corner of the ROC curve. Go one unit to the right when you encounter a negative example, go up when you encounter a positive. Here, we go right, up up, right, up, right, right right, up, right. That’s the full ROC curve.

If there are ties in score, you would go diagonally.

Viewing the ROC curve this way, you can see that the left part of the curve corresponds to the larger values of \( f(x) \) and the right part of the curve to the smaller values of \( f(x) \).

A perfect ROC curve looks like this, where the positives are all ranked higher than the negatives.
The worst ROC curve is when \( f(x) \) is as bad as a random guess, and the true false positive rate and false positive rate go up at the same time.

If you have a curve that is below the diagonal line for random guessing, you probably should negate \( f(x) \) and re-plot it. :)

**Area Under the ROC Curve (AUC or AUROC)**

The AUC is one of the main evaluation metrics for classification. The worst AUC is 0.5 (like random guessing) and the best is 1.0 (ranking all positives have a higher score than all negatives).

**ROC Curves and Ranking**
There is a strong connection between ROC curves and ranking. Why is that? Because the area under the curve is a rank statistic.

Each small block under the ROC curve corresponds to a pair of points, one positive and one negative.

The block is only under the curve because the positive in the pair is ranked higher than the negative, according to the scores from \( f(x) \). The count of these blocks is the area under the curve. We could write it as:

\[
\frac{1}{(\#\text{positives})(\#\text{negatives})} \sum_{i\text{ positive}} \sum_{k\text{ negative}} 1[f(x_i) > f(x_k)]
\]

which is the fraction of positive-negative pairs where the positive ranks higher. (This is what I mean when I say AUC is a rank statistic.) Interestingly, if you look closely, you’ll realize that this is also \( 1 - \) the misranking error used in a special supervised ranking problem, when the data are positives and negatives and we create positive-negative pairs as training data. This is called supervised bipartite ranking.

\[
\frac{1}{(\#\text{positives})(\#\text{negatives})} \sum_{i\text{ positive}} \sum_{k\text{ negative}} 1[f(x_i) > f(x_k)] = 1 - \frac{1}{(\#\text{positives})(\#\text{negatives})} \sum_{i\text{ positive}} \sum_{k\text{ negative}} 1[f(x_i) \leq f(x_k)].
\]

Thus, if you minimize the misranking error while performing supervised ranking, you are actually optimizing the AUC. That is, AUC maximization is equivalent
to supervised ranking.

**AUC’s for Data Exploration**

When someone hands me a dataset, one of the first things I do is plot the ROC curve for each individual feature. In other words, I create a predictive model that is just one feature, and I look at its ROC curve with respect to the label; I do this for all features, and put them on the same plot. If the feature is binary, the ROC curve will look like two connected diagonal lines. If you see diagonal lines on an ROC plot, it means there are probably a lot of tied scores. Below is a plot of the ROC curves for each of the 8 features of the pima-indians-diabetes dataset from the UCI Machine Learning Repository.

![ROC curve plot](image)

You can see that one of the features dominates the others. That feature will be very useful in constructing the combined model. You can also suspect that a couple of the features might be correlated since they have similar ROC curves (you could check that directly if you wanted to). You could also see that all of the features are better than random guessing; most datasets have some features that are not very good, so this dataset is unusual. Let’s run boosted decision trees (one of the ML algorithms covered in the course) and add its ROC curve to the plot.
The boosted models’ ROC curve dominates that of the best feature. This model is a strong predictor of the outcome. I might be concerned that I am overfitting since the predictions’ ROC curve looks so good, so I might also look at ROC curves of the test set to see if they agree.

**Imbalanced Data**

Often we have very few examples of one class, and many of the other. For instance, if we are predicting rare events, such as heart attacks or manhole fires, we might have a lot of examples of the normal class and few of the abnormal class. Think of a small number of positives swimming in a sea of negatives. In the picture below, the positives are red and the negatives are black.

If 99% percent of your data is one label, and the other 1% is the other label, then it is easy to get a classifier with 99% accuracy. Just predict all the points to be the majority class. It’s not a meaningful classifier, but it is accurate.
You can see immediately that accuracy is not an appropriate performance measure. Perhaps we would prefer a classifier like this:

This classifier views each positive as worth much more than each negative. Let’s say that each positive is worth $C_{\text{imb}}$ times as much as each negative. To get a classifier like this, we might minimize a loss function where each positive is worth $C_{\text{imb}}$ times as much as each negative.

$$
\frac{1}{n} \left( C_{\text{imb}} \sum_{i:y_i=1} \ell(y_i, f(x_i)) + \sum_{k:y_k=-1} \ell(y_k, f(x_k)) \right) + \text{Regularization}(f).
$$

Using a loss function like this is the easiest way I know of to handle imbalanced data. There are other ways to handle imbalanced data, for instance, there are methods that create extra fake positive examples, and other methods that reduce the size of the negative class so that the classes are more balanced, but I think the weighting method is the simplest. The choice of weight depends on the problem, and how much value your domain expert places on predicting a positive example correctly compared to a negative one.
If the data are imbalanced, since accuracy is not a meaningful performance measure, we would want to look at the full confusion matrix instead when evaluating performance.

**ROC Curves for Algorithms**

Earlier, we produced ROC curves for single classifiers. Now we will produce ROC curves for algorithms. The algorithm will be run repeatedly, each time producing one point on an ROC curve, and the algorithm will be optimized only for a single point on the curve. It is possible that when we optimize for one point on the curve at a time, that we will construct a better ROC curve than if we use the same model to produce all of the points like we did earlier.

The idea of weighting data that we used for imbalanced classification will be helpful here. We will sweep the imbalanced parameter over its full range and get many different classifiers. For each one, we compute the TPR and FPR, which is a point on our new ROC curve. We start with a very small value of $C_{\text{imbalanced}}$ so that the positives are essentially ignored, and we predict all negatives.

As we increase $C_{\text{imbalanced}}$, we produce a lot of classifiers, each one contributing a point to the ROC curve.
And, as I said earlier, since each point of the curve is optimized separately, the ROC curve for an algorithm is usually better than the ROC curve for a single classifier.