Efficient AUC Learning Curve Calculation

Remco R. Bouckaert

Computer Science Department, University of Waikato, New Zealand
remco@cs.waikato.ac.nz

Abstract. A learning curve of a performance measure provides a graphical method with many benefits for judging classifier properties. The area under the ROC curve (AUC) is a useful and increasingly popular performance measure. In this paper, we consider the computational aspects of calculating AUC learning curves. A new method is provided for incrementally updating exact AUC curves and for calculating approximate AUC curves for datasets with millions of instances. Both theoretical and empirical justifications are given for the approximation. Variants for incremental exact and approximate AUC curves are provided as well.

1 Introduction

Receiver operator characteristics (ROC) graphs allow for easily visualizing performance of classifiers. ROC graphs stem from electronics [3], but has been widely adopted by the medical decision making community [2, 8]. Recently, the machine learning community has taken an interest as well [4, 5][1].

Probabilistic classifiers, such as naive Bayes, are often selected based on accuracy, where the class with the highest probability is taken to be the class prediction. However, classification performance can be increased by optimally choosing the threshold for selecting a class [6], so that for instance a binary class the positive class is selected when the classifier predicts at least a 0.6 probability for this class, and otherwise selects the negative class. One of the benefits of ROC curves is that it allows selection of classifiers that make a probabilistic prediction without committing to a threshold. ROC has further desirable properties for situations where misclassification has different costs for the classes and where the classes are unbalanced [4, 5]. The area under the ROC curve (AUC) is a summary statistic that can be used to select such a classifier.

Another graphical tool for visualizing classifier performance is the learning curve where the x-axis represents the number of training instances and the y-axis the classifiers performance. Learning curves show when enough data is obtained, and no further learning takes place. Also, it can show relative performance of different classifiers at various data set sizes and crossing points. The advantages of AUC as a measure of classification performance especially for probabilistic classifiers make it a good candidate for the y-axis of a learning curve. However,

[1] See for example Workshops on ROC Analysis in AI and ML http://www.dsic.upv.es/~flip/ROCAI2004, ROCML2005, ROCML2006.

calculating the AUC exactly requires sorting over all instances in the dataset, which may become cumbersome when there are many AUCs to be calculated or when there the size of the dataset becomes very large.

In this paper, we consider the computational problems of calculating the AUC for learning curves. Creating a learning curve can be done by incrementally testing the classifier with an instance in the data set, and then training it with the same instance. In this situation, it would be computationally desirable to efficiently update the AUC incrementally with the prediction made by the classifier. We address this issue in Section 3. For datasets with millions of instances, it may become computationally infeasible to calculate the AUC exactly, so we look at a way to approximate AUC in Section 4 and justify its performance. Finally, we consider incremental updating of approximate AUC in Section 5.

2 Exact Area Under ROC Curve

ROC curves deal with situations with a binary class where the class outcomes are called positive and negative. The classifier assigns a probability that the outcome is positive or negative. For a given threshold t, if the probability of the positive class is higher than t, the prediction is positive, otherwise negative. So, by varying t, the number of positive and negative prediction changes. The *true positives* are instances where the prediction is positive while the outcome is indeed positive, and the *false positives* are instances where the prediction is erroneously positive. The same definitions hold for *true negatives* and *false negatives* but for the negative outcomes. A ROC curve is a graph where the x-axis represents the number of true negatives and the y-axis the number of true positives.

The area under the ROC curve is a summary statistic that indicates higher overall performance with higher AUC. AUC is typically calculated [4] by

- Order the sequence of outcomes giving the sequence $o_1, \ldots, o_n$ such that $o_i \leq o_{i+1}$ and let c_i be the class value corresponding to outcome o_i.
- Determine points on ROC curve as follows; sequentially processing the class values $c_1, \ldots, c_n$ the curve starts in the origin and goes one unit up, for every negative outcome the curve goes one unit to the right. Units on the x-axis are $\frac{1}{\#TN}$ and on the y-axis $\frac{1}{\#TP}$ where $\#TP$ ($\#TN$) is the total number of true positives (true negatives). This gives the points on the ROC curve $(0,0), (x_1, y_1), \ldots, (x_n, y_n), (1,1)$.
- Calculate the area under the ROC curve, as the sum over all trapezoids with base x_{i+1} to x_i, that is, $AUC = \sum_{i=0}^{n} (y_{i+1} + y_i)/2(x_{i+1} - x_i)$.

The first step requires sorting the n outcomes, which is $O(n \log n)$ in computational cost. This may not be desirable, for instance, when n is large (say in the millions) and one is interested in the learning curve in terms of AUC. Then the number of times the AUC needs to be calculated is proportional to n, giving $O(n^2 \log n)$ computational cost.

3 Incremental Exact AUC

We start with an ordered sequence of outcomes o_1, o_2 with $o_1 = 0$ and $o_2 = 1$. For each outcome o_{new} with class c_{new}, we record the number of preceding true positives TP_i^n and true negatives TN_i^n and the total number of true positives $\#TP^n$ and negatives $\#TN^n$. So, we have the points (x_i, y_i) on the ROC curve with $x_i = TN_i^n/\#TN^n$ and $y_i = TP_i^n/\#TP^n$. Suppose we have the $n-1$ outcomes ordered $o_1, \ldots, o_{n-1}$ for which we have calculated the AUC as AUC_{n-1}. Further, we have the number of true positives TP_i^{n-1} and the number of true negatives TN_i^{n-1} $(1 \leq i \leq n-1)$. If we receive a new outcome o_{new}, it should be inserted in the sequence of outcomes. Let j be the index of the outcome such that $o_j \leq o_{new} \leq o_{j+1}$. Then we have by definition $AUC_n = \sum_{i=0}^{n}(y_{i+1}+y_i)/2 \times (x_{i+1}-x_i)$ which equals $\sum_{i=0}^{n} (TP_{i+1}^n/\#TP^n + TP_i^n/\#TP^n)/2 \times (TN_{i+1}^n/\#TN^n - TN_i^n/\#TN^n)$ $= \sum_{i=0}^{n} (TP_{i+1}^n + TP_i^n)/2 \times (TN_{i+1}^n - TN_i^n)\frac{1}{\#TN^n\#TP^n}$. We have to distinguish between c_{new} being positive or negative.

If c_{new} is negative, then TP_i^{n-1} equals TP_i^n for any $0 \leq i <\leq n$ and TN_i^{n-1} equals TN_i^n for $i < j$ but $TN_i^{n-1} + 1 = TN_{i+1}^n$ for $i > j$.

So, we can write the above sum as $\sum_{i=0}^{j-1} \frac{(TP_{i+1}^{n-1}+TP_i^{n-1})/2\times(TN_{i+1}^{n-1}-TN_i^{n-1})}{\#TN^n\#TP^n} + \sum_{i=j}^{n-1} \frac{(TP_{i+1}^{n-1}+TP_i^{n-1})/2\times(TN_{i+1}^{n-1}-TN_i^{n-1})}{\#TN^n\#TP^n} + \frac{(TP_{j+1}^{n-1}+TP_i^{n-1})/2\times(TN_{i+1}^{n-1}-TN_i^{n-1})}{\#TN^n\#TP^n} - \frac{(TP_{j+1}^{n-1}+TP_i^{n-1})/2\times(TN_{i+1}^{n-1}-TN_i^{n-1})}{\#TN^n\#TP^n} + \frac{(TP_j^n+TP_{j-1}^n)/2\times(TN_j^n-TN_{j-1}^n)}{\#TN^n\#TP^n} + \frac{(TP_{j+1}^n+TP_i^n)/2\times(TN_{j+1}^n-TN_j^n)}{\#TN^n\#TP^n}$. Now, the first three terms are equal to AUC_{n-1} weighed with $\frac{\#TN^{n-1}}{\#TN^n}\frac{\#TP^{n-1}}{\#TP^n}$. So, we have $AUC_n = AUC_{n-1}\frac{\#TN^{n-1}}{\#TN^n}\frac{\#TP^{n-1}}{\#TP^n} - \frac{(TP_{j+1}^{n-1}+TP_i^{n-1})/1\times(TN_{i+1}^{n-1}-TN_i^{n-1})}{\#TN^n\#TP^n} + \frac{(TP_j^n+TP_{j-1}^n)/2\times(TN_j^n-TN_{j-1}^n)}{\#TN^n\#TP^n} + \frac{(TP_{j+1}^n+TP_i^n)/2\times(TN_{j+1}^n-TN_j^n)}{\#TN^n\#TP^n}$.

If c_{new} is positive $TP_i^{n-1} + 1 = TP_{i+1}^n$ and $TN_i^{n-1} = TN_{i+1}^n$ $TP_i^{n-1} = TP_i^n$ and $TN_i^{n-1} = TN_i^n$ for $i < j$ and TN_i^{n-1} equals TN_i^n but TP_i^{n-1} equals $TP_{i+1}^n - 1$ for $i > j$. So, we get an extra term $+\sum_{i=j}^{n} \frac{1/2\times(TN_j^n-TN_{j-1}^n)}{\#TN^n\#TP^n} = \frac{1/2\times(\#TN^n-TN_{j-1}^n)}{\#TN^n\#TP^n}$ to add.

In short, AUC_n can be calculated in constant time, provided we know the terms in the formula. Effectively, we need to maintain correct administration of the number of preceding true positives and negatives at each point. Wlog, suppose the outcome c_j is positive, then $TP_i^n = TP_i^{n-1}$ for $i < j$ and $TP_i^n = TP_i^{n-1} + 1$ for $j \geq i$, so updating the true positives is linear in the number of instances. The total complexity of calculating AUC_n given AUC_{n-1} thus is $O(\log n)$ for finding the location of o_{new} in the outcomes and $O(n)$ for maintaining the cumulative true positive and negative counts. So, instead of spending $O(n \log n)$ for calculating a complete AUC, this can be achieved in $O(n + \log n)$. Therefore, calculating a learning curve becomes $O(n^2 + n\log n)$ instead of $O(n^2 \log n)$ for the non-incremental method. Note that most of the work goes into maintaining the true positive and negative counts (TP_i^n and TN_i^n).

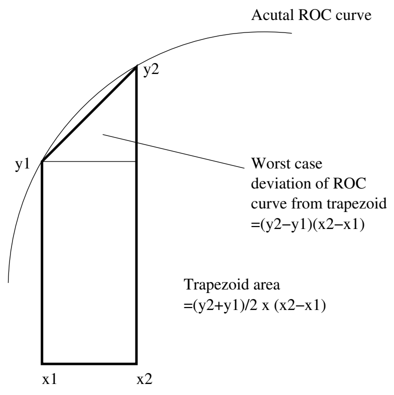
Section 5 provides a technique that can be adapted to reduce the amount of work of updating for every new outcome from $O(n)$ to $O(\sqrt{n})$.

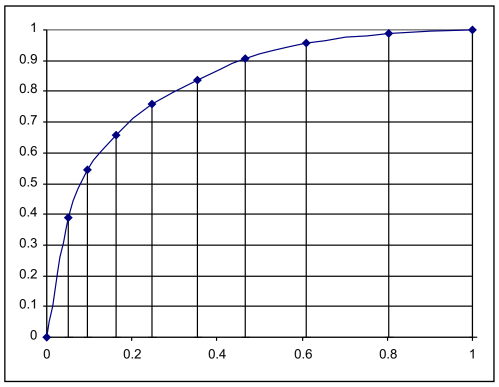
4 Approximate AUC

Instead of calculating the AUC exactly, we can approximate it by taking m bins, each bin containing the number of positive and negative outcomes in an interval. The procedure then goes as follows;
- For every result r_i representing the probability of the positive class, identify the bin $j = \lfloor r_i \cdot m \rfloor$ and update bin j with outcome o_i. To prevent many bins remaining empty (see experiments later in this section), apply a log-log transform first ($r'_i = \frac{1}{2} - \alpha \log(-log(r_i))$ if $r_i \leq \frac{1}{2}$ and $r'_i = \frac{1}{2} + \alpha \log(-log(1-r_i))$ if $r_i > \frac{1}{2}$ where α is an appropriate chosen constant which can be determined from a small sample of the data)
- Approximate ROC curve. Let b_i^+ be the number of positive outcomes in bin i, and b_i^- be the number of negative outcomes. The curve starts in the origin and goes with a straight line to (b_0^+, b_0^-). Then, from (b_0^+, b_0^-) to $(b_0^+ + b_1^+, b_0^- + b_1^-)$. From $(b_0^+ + b_1^+, b_0^- + b_1^-)$ to $(b_0^+ + b_1^+ + b_2^+, b_0^- + b_1^- + b_2^-)$, etc. So, we build the ROC curve from line pieces from $(\sum_{j=0}^{i} b_j^+, \sum_{j=0}^{i} b_j^-)$ to $(\sum_{i=0}^{j+1} b_j^+, \sum_{j=0}^{i+1} b_j^-)$.
- Let $y_i = \sum_{i=0}^{j+1} b_j^+$ and $x_i = \sum_{j=0}^{i+1} b_j^-$ the cumulative number of true positives and negatives, then we can calculate the area under the approximate ROC curve, as before, by taking the sum over the trapezoids $AUC = \sum_{i=0}^{m}(y_i + y_{y-1})/2 \times (x_i - x_{i-1})$, where by convention $x_{-1} = y_{-1} = 0$.

The complexity of this procedure with m bins is $O(n+m)$. Note that we need to keep track of the cumulative number of true positives y_i, but not necessarily the cumulative number of true negatives x_i, since $AUC = \sum_{i=1}^{m}(y_i + y_{y-1})/2 \times (x_i - x_{i-1}) = \sum_{i=1}^{m}(y_i + y_{y-1})/2 \times b_i^-$. Calculating a learning curve then becomes $O(n^2 + n \cdot m)$ instead of $O(n^2 \log n)$.

Fig. 1. Trapezoid approximation of ROC curve (left). Bins can have different widths (right). Here 10 bins are shown, the smallest is 5% the widest 20%.


Acutal ROC curve
y2
y1
Worst case deviation of ROC curve from trapezoid =(y2−y1)(x2−x1)
Trapezoid area =(y2+y1)/2 x (x2−x1)
x1
x2

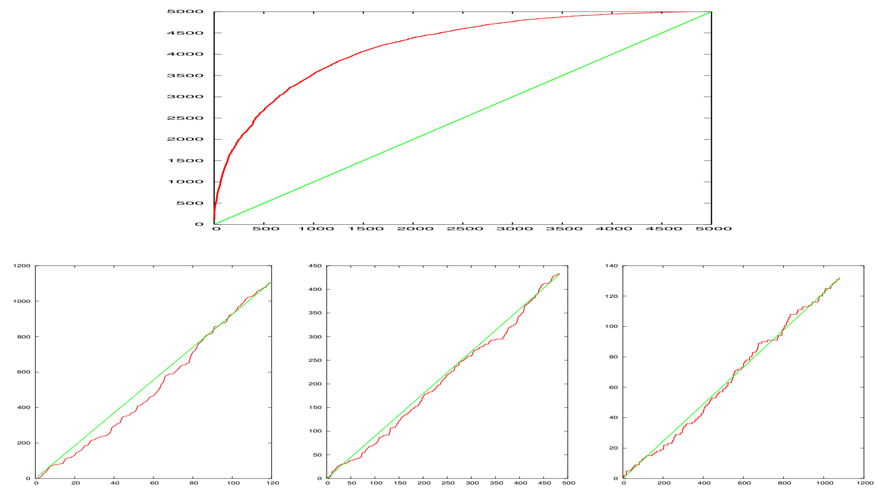


4.1 Maximum Error of Approximation

A trapezoid is used to approximate the ROC curve in various steps where the beginning and ending of the upward part of the trapezoid coincide with the ROC curve. Since the ROC curve is a monotonely rising function, the worst possible deviation of the ROC curve from the trapezoid is if it goes straight up and then straight sidewards (or vise versa). In this case, the difference between the ROC curve and its approximation is the area in the triangle of the trapezoid (see left of Figure 1).

Figure 1 (right) illustrates that the bins need not be of equal width in terms of the ROC curve. This is caused by the fact that the class probability does not need to be uniformly distributed, and in fact it often is not. Therefore, the number of negatives represented on the x-axis need not be uniformly distributed over the bins. As a consequence, the maximum error cannot be estimated before the data has been seen. However, once the bins are filled, the error cannot exceed the triangle above the trapezoid of the bin. which can be calculated as $max\_error = \sum_{i=0}^{m}(y_i - y_{i-1})/2 \times (x_i - x_{i-1})$ (taking the same form as the approximate AUC, but with a plus sign replaced by a minus) which equals $\sum_{i=0}^{m} \frac{1}{2} b_i^+ \times b_i^-$.

Fig. 2. ROC-plots of extended simple model. Top, full curve, below zoomed into true negative intervals starting at 0.1 (left), 0.5 (middle) and 0.9 (right) and extending 0.001



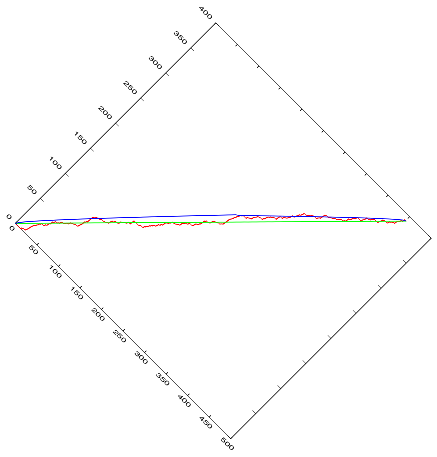
To verify that the ROC curve does not deviate too far from the straight line, we created a simple probabilistic model $P(y, x1, x2, x3)$ taking the decomposition $P(y)P(x1|y)P(x2|y)P(x3|y)$ with $P(y = 0) = P(y = 1) = \frac{1}{2}$, $P(x1|y) = P(x2|y) = N(y, 1)$ where $N(m, \sigma)$ is the normal distribution with mean m and variance σ. The distribution over the binary attribute $x3$ is defined by $P(x3 = 0|y = 0) = P(x3 = 1|y = 1) = \frac{1}{4}$. The full ROC curve was generated using 10 thousand instances and is shown in Figure 2. The snapshots at the bottom of the figure show the ROC curve from 0.1 to 0.1001 (left), from 0.5

to 0.5001 (middle), and from 0.9 to 0.9001 (right), generated with 10 million samples (with x and y axis scaled to make them equally sized). A line starting in the left lower corner to the right upper corner is shown as well. Similar results hold when the discrete attribute $x3$ is removed from the model (not shown).

4.2 Error Estimate

To get a better estimate of the error in the approximation, consider Figure 2 bottom middle. Judging from Figure 2 it appears to be a reasonable assumption that within a sufficiently small bin, the probability of getting a positive or negative class is constant. With every positive class, the curve goes up, with every negative class the curve goes sideways. So, the ROC curve within a bin can be considered a one dimensional random walk with probability p of going up, and $1-p$ of going down once we rotate the curve 45% (see Figure 3). Furthermore, we know the start and end point of the random walk, namely starting at the origin and ending for bin j at (b_j^+, b_j^-).

Fig. 3. Figure 2 bottom middle rotated, showing random walk. Expected deviation from line is drawn as well



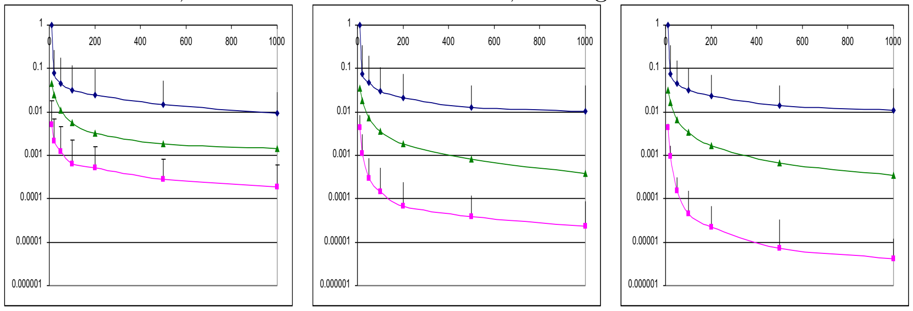
The expected value of a one dimensional random walk with unit steps and equal probability of positive and negative steps after k steps is $\sqrt{k}$. So, the expected area (representing the error in the AUC estimate) after k steps is bounded by $\sum_{i=0}^{k} \sqrt{i} \approx \int_0^k \sqrt{x}dx = \frac{2}{3}k\sqrt{k}$. However, the steps in the ROC curve are not of unit length, but only of size $\sqrt{\frac{1}{2}}$ due to the rotation, leaving an expected area of $\sqrt{\frac{1}{2}} \times \frac{2}{3}k\sqrt{k} = \frac{1}{3}k\sqrt{2k}$ after k steps. Now, due to symmetry, the expected area due to the first half number of steps contained in a bin equals the expected area in the second half. So a bin containing $q = b^+ + b^-$ ($b^+ = b^-$) outcomes has an expected error of $2 \times \frac{1}{3}\frac{q}{2}\sqrt{2\frac{q}{2}} = \frac{1}{3}q\sqrt{q}$ for a single bin. Furthermore, though the probability of a positive or negative outcome remains approximately constant within a bin, it changes from bin to bin. A bin with b^+

positive classes and b^- negative classes can be rescaled so that every positive step is $b^-/(b^+ + b^-)$ and every negative step $b^+/(b^+ + b^-)$. Instead of a unit length random walk, we get a $\sqrt{(\frac{b^+}{(b^++b^-)})^2 + (\frac{b^-}{(b^++b^-)})^2} = \sqrt{\frac{(b^+)^2+(b^-)^2}{(b^++b^-)^2}}$. So, a bin with $q = b^+ + b^-$ outcomes can expect to have its error bounded by $2 \times \sqrt{\frac{(b^+)^2+(b^-)^2}{(b^++b^-)^2}}\frac{1}{3}\frac{q}{2}\sqrt{2\frac{q}{2}} = \frac{1}{3}\sqrt{(b^+)^2 + (b^-)^2}\sqrt{q}$ (using $\sqrt{\frac{1}{(b^++b^-)^2}} = \frac{1}{q}$). The complete set of bins thus can be calculated as $\sum_{i=1}^{m} \frac{1}{3}\frac{\sqrt{(b_i^+)^2+(b_i^-)^2}}{\#TP+\#TN}\sqrt{\frac{b_i^++b_i^-}{\#TP+\#TN}}$. Like for the maximum error, we see that the bin containing the most outcomes contributes most to the expected error.

4.3 Empirical Evaluation of Approximation

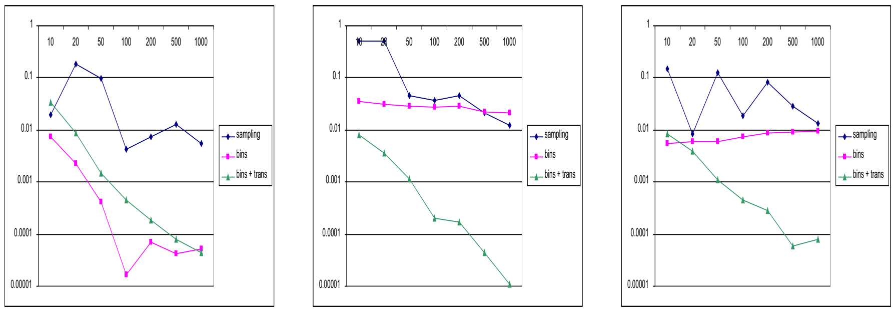
Figure 4 shows the mean error of AUC estimates for 100 samples drawn from the model $P(y, x1, x2, x3)$ described above. The exact AUC was calculated as described in Section 2. The top line shows the AUC estimate based on a sample from the outcomes drawn from the model [1]. Though this approach does not require visiting all outcomes, it does require sorting. The bottom line is the average error obtained with the bin based approach. The maximum error found is indicated with an upwards error bar. The minimum absolute error turned out to be zero for the bin based approach and just above the mean bin based error for the sample based approach (not shown to prevent cluttering the graphs). The middle line is the maximum error estimate for the bin based approach.

Fig. 4. Absolute error in AUC estimate (on y-axis) for different numbers of bins/samples (on x-axis). Top line is sampling based, middle line is maximum estimate of error for bin based estimate, bottom line is bin based. Left, dataset with 100 instances, middle with 1000 instances, and right 10000 instances.



Clearly, the bin based approach is superior over sampling in its error; the maximum error is consistently below the error of the sampling based approach. Surprisingly, even with as little as ten bins, the error remains mostly below 1%, quickly dropping off with increasing number of bins. For sufficiently large data sets and 1000 bins, the estimate can become accurate up to the fifth digit. Increasing the number of outcomes decreases the error for the bin based approach, but not for the sampling based approach. The maximum error is clearly a large upper bound on the actual error with a factor of 10.

Fig. 5. Error in AUC on letter and newsgroup data. Axis same as Figure 5. Left class is A to O in letter, middle class is A in letter, right newsgroup.



Unbalanced classes can cause many bins to be empty, and hence the error to become larger. Figure 5 shows this effect with the UCI letter dataset trained with naive Bayes. Left, the class is distinguishing between the letters A to O and the remaining letters, which gives a reasonably balanced class. The error of the approximate approach behaves as for the simple data source. In the middle, the class is recognizing A against all other letters, which gives an unbalanced class. The approximate approach does not increase performance with increasing number of bins because most bins remain empty. To remedy this, the bin can be chosen based on a log-log transform. The line ending lowest in the middle of Figure 5 represents the error with bins chosen this way. The left figure shows that the transformation does not affect the error too much in case the class is balanced. The graph on the right shows an example with on the 20 newsgroups data, which has many attributes (1000), causing almost all probabilities generated by naive Bayes to be very close to zero and one. The transformation significantly reduces the error. In all cases, the sample based approach is outperformed by the transformation.

5 Approximate Incremental AUC

If we know the approximate AUC at a given moment during the stream, we can calculate the approximate AUC for the next point in the curve. Suppose the next outcome is positive. Figure 6shows how we need to adept the curve.

Let y_{j-1} (x_{j-1}) be the number of true positives (negative) before the target bin, and y_j (x_j) be the number of true positives (negative) before and including the target bin. So, we have $y_j = y_{j-1} + b_j^+$ and $x_j = x_{j-1} + b_j^-$ where b_j^+ (b_j^-) is the number of positives (negatives) in the target bin. Then, after a similar derivation as in Section 3, we get

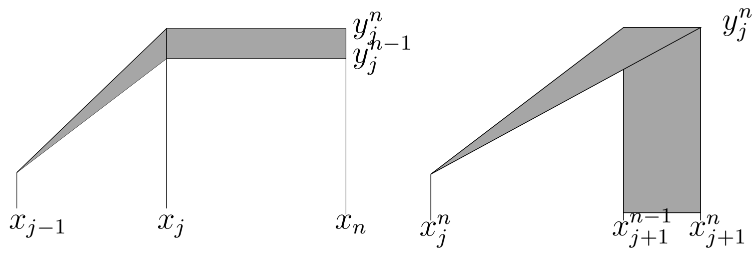
$$AUC_n = AUC_{n-1}\frac{\#TP^n - 1}{\#TP^n} + \frac{x_m - x_j}{\#TP^n \#TN^n} + \frac{(y_j + y_{j-1} + 1)b_j^-}{2\#TP^n \#TN^n} - \frac{(y_j + y_{j-1})b_j^-}{2(\#TP^n - 1)\#TN^n}$$

where $b_j^- = x_j^n - x_{j-1}^n = x_j^{n-1} - x_{j-1}^{n-1}$. And if c_j is negative, we get

$$AUC_n = AUC_{n-1}\frac{\#TN^n - 1}{\#TN^n} + \frac{(y_j + y_{j-1} + 1)b_j^-}{2\#TP^n\#TN^n} - \frac{(y_j + y_{j-1})(b_j^- - 1)}{2\#TP^n(\#TN^n - 1)}$$

where $b_j^- = x_j^n - x_{j-1}^n$ is the number of true negatives in bin j after the update. To calculate AUC_n we need to know y_j and x_j. This can be calculated by $y_j = \sum_{i=0}^{j} b_i^+$ and $x_j = \sum_{i=0}^{j} b_i^-$ however, this is linear in m. We can improve this at the cost of some memory as follows. Suppose that we evenly distribute a anchor points over the m bins, and each anchor point we track the number of positives and negative for between the anchor points. So, when a positive item is added, we update the target bin, and also the anchor point containing the target bin. When $j = \lfloor \frac{r_i}{n} \rfloor$ is the index of the target bin, then $k = \lfloor \frac{j}{a} \rfloor$ is the corresponding target anchor point.

Fig. 6. Areas affected by updating the ROC curve with a positive class (left) and a negative class (right).



Now, to calculate y_j (and x_j), we sum all b_i to the nearest lower anchor, then proceed adding the positives (negatives) in the previous anchors. Let a_i^+ and a_i^j be the number of positives and negatives kept in anchor a_i and let i_a be the index of the bin associated with the anchor. Then we have

$$y_j = \sum_{i=i_a}^{j} b_i^+ + \sum_{j=1}^{k} a_i^+, x_j = \sum_{i=i_a}^{j} b_i^- + \sum_{j=1}^{k} a_i^-$$

The expected number of summations is the expected number of $(j - i_a)$ for the bins and the expected number of k for the anchors. Assuming both are uniformly distributed, we have $E\{j - i_a\} = \frac{1}{2}\frac{m}{a}$ and $E\{k\} = \frac{1}{2}a$. So, together we expect $f(a) = \frac{1}{2}\frac{m}{a} + \frac{1}{2}a$ summations instead of $\frac{1}{2}m$. To minimize $f(a)$ we take the derivative and get $-\frac{1}{2}\frac{m}{a^2} + \frac{1}{2} = 0$ which gives $a = \sqrt{m}$. Then the expected number of summations is $f(\sqrt(m)) = \frac{1}{2}\sqrt{m} + \frac{1}{2}\sqrt{m} = \sqrt{m}$. So, calculating the next AUC becomes $O(\sqrt{m})$ instead of $O(m)$.

Two simple optimizations are possible;
- Instead of summing over bins to the first preceding anchor point, we can sum to the *closest* anchor point. This halves the number of expected summations

over the bins. If the closest anchor point is a higher bin than the current bin, we calculate y_j as $-\sum_{i=j+1}^{i_a} jb_i^+ - \sum_{j=1}^{k+1} a_i^+$.
- We can keep track of the total number of positives and negatives and for every anchor points over half the number of bins we sum over all following anchors (instead of all preceding ones). This halves the expected number of anchor points. To calculate the contribution at the anchor point, just subtract the obtained sum from the total number of positives or negatives. Note that the technique with the anchor points can be adapted to the exact AUC calculation in Section 3.

6 Conclusions

This article presents a computationally efficient approach for calculating AUC learning curves by incrementally updating the AUC with each new sample point. Further, a method for approximating AUC was presented and both theoretically and empirically supported to perform well. In particular, the estimate is better than a sample based approach at very modest computational cost, which is especially important for calculating learning curves for very large data sets. Error estimates of the approximation can be calculated on the fly.

An issue for further research is the question how to efficiently combine a number of ROC curves in order to create a single ROC curve from various curves obtained from a cross validation run on the data as addressed in [7].

Acknowledgements

I thank all Waikato University Machine Learning group members for his stimulating discussions on this topic, especially Richard Kirkby for making his Moa system for experimenting on large data streams available.

References

1. Andriy Bandos, Howard E. Rockette, D. Gur. Resampling Methods for the Area Under the ROC Curve. ROCML'06.
2. J. Beck. and E. Schultz. The use of ROC curves in test performance evaluation. Archives of Pathology and Laboratory Medicine, 110:13-20. 1986
3. J. P. Egan. Signal Detection Theory and ROC analysis. Academic Press, New York, 1975.
4. Tom Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers Technical Report HPL-2003-4, HP Labs
5. P.A. Flach. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. ICML, 2003.
6. N. Lachiche and P.A. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. ICML, 2003.
7. Sofus A. Macskassy, Foster Provost and Saharon Rosset. Pointwise ROC Confidence Bounds: An Empirical Evaluation. ROCML'05.
8. J.A. Swets. Measuring the accuracy of diagnostic systems. Science 240, 1285-93, 1988.
9. I.H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco, 2000.