Robust Reductions from Ranking to Classification

Anonymous Author(s)
Affiliation
Address
City, State/Province, Postal Code, Country
email

Abstract

We prove that the problem of optimizing the Area under the Receiver Operating Characteristic curve (AUC) can be reduced to a related classification problem in an assumption-free setting. This proves that the ability to rank well (at least with respect to AUC) is equivalent to the ability to classify well. Used constructively, this proof provides a host of new algorithms for optimizing AUC which exhibit good performance. The analysis guarantees that the resulting ranking algorithms are robust to errors made by the classifier, in the sense that a large AUC cannot be induced with a small number of mis-classifications.

1 Introduction

Problems. In the bipartite ranking problem, we are given a set of unlabeled instances belonging to two classes (0 and 1), and the goal is to rank all instances from class 0 before any instance from class 1. Area under the ROC curve (AUC) is a measure of successful ranking where the loss is greater for mistakes at the beginning or the end of an ordering. This satisfies the intuition that an unwanted item placed at the top of a recommendation list, has higher asssociated loss than when placed in the middle. A handy shorthand for understanding AUC, is that it is one minus the bubble sort distance between the predicted ordering and the true ordering.

The classification problem is simply predicting whether a label is 0 or 1 with success measured according to the probability of a misprediction.

These problems appear quite different. For example, the classification loss function is defined on a per-example basis while AUC is defined for sets of examples.

A Fundamental Question. A natural fundamental question to ask is "Are these problems truly different?" If the answer is yes, it implies that we require fundamentally different algorithms to optimize these different loss functions. If the answer is "no", then we can reuse our existing algorithms and techniques to optimize AUC. An answer of "no" also suggests that other ranking losses such as those used by search companies may be solved directly with reuse of our existing technology.

Question Complexities. Several basic observations help define this problem.

1. The question can not be answered satisfactorily under any assumptions about the way the world produces data. Even a minimal IID sample assumption is not met by a number of real-world applications where AUC is important. This observation implies most forms of analysis are not suitable.

Machine learning reductions analysis *can* meet this challenge. At a high level, learning reductions analysis bounds the realized AUC performance in terms of the realized classification performance. Since the analysis is relative, no assumptions about the real-world process need be made.

A consequence of this observation is that our analysis must cope with arbitrary high order dependencies between the labels of examples.

2. A natural approach to solving ranking is to order examples according to some "score" or estimated conditional class probability. There does indeed exist a general reduction from conditional class probability estimation to (hard) 0/1 classification [LZ05]. But, as discussed in [LZ05] this provides no satisfying solution for AUC. The fundamental difficulty is exhibited by test sets with one '1' and many '0's. For these datasets, a classification error on the '1' with perfect prediction for the '0's may result in a dramatic AUC loss implying a score based process is not robust to misprediction. It is this observation which necessitates the dependence on the ratio of 0's to 1's in the IID stability analysis of AUC [SHD05].

   A consequence of a generalization of this argument is that no robust solution can make predictions given only a single example.

Our Approach. Given these observations, the minimal complexity method for which we can hope to succeed uses pairwise classifiers $c(x_1, x_2)$ for predicting which element is larger. We show this hope is realized—a pairwise classifier is able to transform 0/1 prediction performance into AUC performance no matter what joint (nonIID!) distribution produces the test set.

Suppose we are given a good (but not perfect) classifier $c(x_1, x_2)$ which predicts whether $x_1 < x_2$ or not. How should we build a good ranking? One natural algorithm is to find the ordering which minimizes disagreement with the learned classifier. This algorithm is known to be NP-hard to execute in general. Fortunately for everyone, it turns out there is a better performing natural algorithm that is computationally tractable.

We prove that a second natural algorithm (order according to the number of times one element is preferred to another breaking ties randomly) results in better performance. In particular, for this algorithm an ϵ suboptimality for the learned pairwise classifier translates into at most a 2ϵ AUC suboptimality for constructed ordering.

Information source varieties. With slight modifications, the analysis we prove (and the implied learning algorithims) can be used to train rankers from a variety of information sources including the following:

1. Standard labeled training data. There are a number of classification-like settings where a ranking is actually preferred. For example, sometimes the false positive/false negative ratio is either unknown or variable so it is desirable to have a "nob" on a learning system which can trade-off between the possibilities.
2. Strong pairwise preferences. A strong preference has the form "the first item is (strictly) better than the second item". Sometimes the information encoded by experts is pairwise. For example, it might be easier to predict the relative importance of papers under review than it is to predict some absolute value.
3. Weak pairwise preferences. A weak preference has the form "the first item is at least as good as the second item". Our results again apply here.

Organization

Section 2 formalizes the setting. Section 1 presents the regret analysis with respect to the AUC. Section 4 shows that the NP-hard algorithm is actually suboptimal w.r.t. the simple randomized tie breaking algorithm analyzed in section 1. Section 5 presents some empirical experiments suggesting this approach is sane.

2 Formal Setup

We first define the problems we are considering and then their derived quantities.

A *binary classification problem* is defined by a distribution B over $X \times \{0,1\}$, where X is a feature space and $\{0,1\}$ is the binary prediction space. The goal is to find a classifier $c : X \to \{0,1\}$ minimizing the *classification loss*,

$$e(c, B) = \mathbf{Pr}_{(x,y)\sim B}[c(x) \neq y].$$

In *importance weighted* binary classification, there is some importance associated with misclassifying each example. The learner gets to know the importances of training examples, and the goal is to minimize the expected importance-weighted loss,

$$e(c, B) = \mathbf{E}_{(x,y,w)\sim B}\left[wI(c(x) \neq y)\right],$$

where the distribution B is now over $X \times \{0,1\} \times [0,\infty)$. Importance-weighted classification can be reduced to binary classification (for example, using the Costing reduction [ZLA03]).

Let $o : X \times X \to \{0,1\}$ be an ordering function that given as input any two instances in X outputs 1 if it agrees with the ordering of its arguments; otherwise it outputs 0. The *AUC loss* of an ordering o on a set $S \in (X \times \{0,1\})^*$ is defined as

$$l_{\mathrm{AUC}}(o, S) = \frac{\sum_{i,j} I(y_i > y_j) o(x_i, x_j)}{\sum_{i,j} I(y_i \neq y_j)}.$$

(Indices i and j in the summations range from 1 to n, with $i \neq j$.)

An *AUC problem* is defined by a distribution D over $(X \times \{0,1\})^*$. The goal is to find an ordering $o : X \times X \to \{0,1\}$ minimizing the expected AUC loss on D,

$$l(o, D) = \mathbf{E}_{S\sim D} l(o, S).$$

Note that D may encode arbitrary dependencies between examples.

Regret We prove a transformation bound on *regret* rather than loss. Regret (generally) is how well we could have done in comparison to how well we did. It separates errors from unremovable noise in the problem, thus the bounds apply nontrivially even on problems with large inherent noise.

Formally, the *classification regret* of a classifier c on a distribution B on (importance-weighted) binary examples is defined as

$$r(c, B) = e(c, B) - \min_{c^*} e(c^*, B).$$

Similarly, the *AUC regret* of an ordering function o on a distribution D over $(X \times \{0,1\})^*$ is given by

$$r_{\mathrm{AUC}}(o, D) = l(o, D) - \min_{o^*} l(o^*, D).$$

Our goal is to design a ranking algorithm for which a small binary regret incurred by the selector cannot imply a large ranking regret.

3 Ordering by the Number of Wins: Regret Transform

The reductions we analyze consist of two components. The training part, Auc-Train (Algorithm 1), transforms mixed pairs of examples into binary data. (A pair $(x_1, y_1), (x_2, y_2)$ is *mixed* if $y_1 \neq y_2$.)

For any process D generating datasets $S \in (X \times \{0,1\})^*$, we can define an induced distribution on importance weighted binary examples in $(X \times X) \times \{0,1\} \times [0,\infty)$ by first drawing S from D, and then applying Auc-Train to S. We denote this induced distribution by Auc-Train(D).

The test portion (Algorithm 2) uses the pairwise classifier c learned in Algorithm 1 to run a tournament on the test set, and then creates an ordering according to the number of wins in the tournament, breaking ties randomly.

Algorithm 1 AUC-TRAIN (labeled set $S \in X^n$, binary learning algorithm A)

1. Set $n^0 = |\{(x, y) \in S : y = 0\}|$.
2. Let
$$S' = \left\{ \left\langle (x_1, x_2), I(y_1 > y_2), \frac{1}{n^0(n - n^0)} \right\rangle : (x_1, y_1), (x_2, y_2) \in S \text{ and } y_1 \neq y_2 \right\}$$
3. return $c = A(S')$.

Algorithm 2 DEGREE (Unlabeled set Q, importance-weighted pairwise classifier c)

1. For $x \in Q$, let $\deg(x) = |\{x' : c(x, x') = 1, x' \in Q\}|$.
2. Sort Q in the descending order of $\deg(x)$, breaking ties randomly.

Theorem 1. *For all joint distributions D, for all pairwise classifiers c,*

$$r_{\text{AUC}}(\text{DEGREE}(D, c), D) \leq 2r(c, \text{AUC-TRAIN}(D)).$$

Note the quantification in the above theorem: it applies to *all* settings where algorithms 1 and 2 are used, in particular to D with arbitrary dependences between examples.

Proof: Given an unlabeled test set $Q \in X^n$, the joint distribution D induces a conditional distribution $D(Y_1, \ldots, Y_n \mid Q)$ over the set of label sequences $\{0, 1\}^n$. We will prove the theorem for any fixed Q, and then take the expectation over the draw of Q at the end.

Let AUC-TRAIN(Q) denote the distribution on binary examples obtained by drawing a label sequence from $D(Y_1, \ldots, Y_n \mid Q)$ and running AUC-TRAIN on Q augmented with the drawn labels. For a label sequence $y^n \in \{0, 1\}^n$, let $\#(y^n) = |\{(i, j) : y_i < y_j\}|$ denote the number of mixed pairs in $y^n = y_1 \ldots y_n$.

It will be convenient to define loss and regret on pairs of instances in Q. Given $x_i, x_j \in Q$, we define the *loss* of ordering x_i before x_j as

$$l_Q(x_i, x_j) = \mathbf{E}_{y^n \sim D(Y^n|Q)} \frac{I(y_i > y_j)}{\#(y^n)}.$$

The weighting by the inverse of the number of mixed pairs inside the expectation essentially upweights example pairs from highly asymmetric label sets.

If $l_Q(x_i, x_j) < l_Q(x_j, x_i)$, the *regret* $r_Q(x_i, x_j)$ of ordering x_i before x_j is 0; otherwise, $r_Q(x_i, x_j) = l_Q(x_i, x_j) - l_Q(x_j, x_i)$.

The AUC loss of an ordering o on D given Q can be written as

$$\begin{aligned} l_{\text{AUC}}(o, Q) &= \mathbf{E}_{y^n \sim D(Y^n|Q)} \frac{\sum_{i,j} I(y_i > y_j) o(x_i, x_j)}{\#(y^n)} \\ &= \sum_{i,j:o(x_i,x_j)=1} l_Q(x_i, x_j). \end{aligned}$$

We can assume without loss of generality that the ordering minimizing the AUC loss (thus having zero AUC regret) is $x_1 x_2 \ldots x_n$. All regret-zero pairwise predictions must be consistent with the ordering; i.e., $r_Q(x_i, x_j) = 0$ for all $i < j$. (Otherwise there must exist a pair $i < j$ with $l_Q(x_i, x_j) > l_Q(x_j, x_i)$; but then interchanging x_i and x_j in the ordering decreases the ranking loss, contradicting the assumption that $x_1 x_2 \ldots x_n$ is loss minimizing.)

The AUC regret of o on D given Q can thus be decomposed as a sum of pairwise regrets:

$$r_{\text{AUC}}(o, Q) = \sum_{i<j:o(x_j,x_i)=1} r_Q(x_j, x_i).$$

The classification loss of c on AUC-TRAIN(Q) can be written as

$$e(c, \text{AUC-TRAIN}(Q)) = \mathbf{E}_{y^n \sim D(Y^n|Q)} \left[\frac{1}{\#(y^n)} \sum_{i,j} I(y_i > y_j) c(x_i, x_j) \right]$$
$$= \sum_{i,j:c(x_i,x_j)=1} \mathbf{E}_{y^n \sim D(Y^n|Q)} \frac{I(y_i > y_j)}{\#(y^n)}$$

Observe that even though c makes predictions on all pairs in Q, it is only charged for mistakes on mixed pairs (i.e., for choosing a bad example when a good one was available).

The classification regret can thus also be written in terms of pairwise regrets:

$$r(c, \text{AUC-TRAIN}(Q)) = \sum_{i<j:c(x_j,x_i)=1} r_Q(x_j, x_i).$$

We will think of c as an adversary trying to induce a large AUC regret without paying much in classification regret. Our goal is to bound the regret ratio. Notice that the more lopsided the label set of Q is, the more c is paying for putting a 1 before a 0.

The *degree* of an example x in Q, denoted $\deg(x)$, is the number of other examples in Q that it beats, according to c. The algorithm ranks examples in the non-increasing order of their degrees, breaking ties randomly.

If the set of pairwise predictions made by c on Q is consistent with some linear ordering, then all degrees are unique, and DEGREE(c, Q) outputs the consistent ordering. As readily seen from the decompositions above, the ratio of the AUC regret to the classification regret in this case is 1. Otherwise, DEGREE(c, Q) randomizes over a set of linear orderings consistent with the degrees. Let $U = U(c, Q)$ denote this set. Examples with unique degrees occupy fixed positions in all orderings in U. The set U is formed by all possible permutations within subsets of elements having the same degree.

The first observation is that for every pair (i, j), either all orderings in U agree on how to order x_i and x_j (when $\deg(x_i) \neq \deg(x_j)$) or exactly half of them orders x_i before x_j, and the other half orders x_j before x_i (when $\deg(x_i) = \deg(x_j)$). This follows immediately from how U is constructed. The AUC regret of the algorithm on Q is the average AUC regret over all orderings in U, $\frac{1}{|U|} \sum_{o \in U} r_{\text{AUC}}(o, Q)$. Thus each pairwise regret contributes to the AUC regret with a factor in $\{0, 1, \frac{1}{2}\}$.

Pairwise regrets have a useful property (Lemma 3.1) that for any x_i, x_j, and x_k in Q,

$$r_Q(x_i, x_j) + r_Q(x_j, x_k) = r_Q(x_i, x_k).$$

This observation implies that for any $i < j$,

$$r_Q(x_j, x_i) = \sum_{k=i}^{j-1} r_Q(x_{k+1}, x_k).$$

The pairs (x_{k+1}, x_k), $1 \leq k < n$, will be called *basic*.

The adversary c induces a tournament on Q by specifying the direction of each edge. The algorithm then outputs an ordering $o = \text{DEGREE}(Q, c)$ according to the number of wins in the tournament, breaking ties randomly.

We saw that both $r_{\text{AUC}}(o, Q)$ and $r(c, \text{AUC-TRAIN}(Q))$ can be written as sums of basic regrets. Thus upper bounding the ratio of the coefficients with which any given basic pair appears in the two sums gives an upper bound on the regret ratio.

For each k, the coefficient of $r_Q(x_{k+1}, x_k)$ in the AUC regret corresponds to the number of pairs (x_i, x_j) such that $i \leq k < j$ and x_j is ordered before x_i in o, in which case

$r_Q(x_{k+1}, x_k)$ contributes to a non-zero $r_Q(x_j, x_i)$. The coefficient of $r_Q(x_{k+1}, x_k)$ in the classification regret is the number of edges (x_i, x_j) with $i \leq k < j$ that c directs incorrectly. We need to upper bound the ratio of the two coefficients, over all k.

Fix k and consider a partition of Q into $Q^0 = \{x_1, \ldots, x_k\}$ and $Q^1 = \{x_{k+1}, \ldots, x_n\}$.

The adversary pays $k(n-k) - [\sum_{i=1}^{k} \deg(x_i) - \binom{k}{2}]$. For $x_j \in Q^1$, define $A_j = \{x_i \in Q^0 : \deg(x_i) < \deg(x_j)\}$ and $B_j = \{x_i \in Q^0 : \deg(x_i) = \deg(x_j)\}$. Note that $A_{n-k} \subseteq \ldots \subseteq A_2 \subseteq A_1$. The adversary wants to choose a degree sequence to maximize $\sum_{i=1}^{k} \deg(x_i)$, thus minimizing what it pays, and to maximize $\sum_{j=1}^{n-k}(|A_j| + \frac{1}{2}|B_j|)$. Since $\deg(x_i) \leq$

Assume for simplicity that n is even. The adversary can force the algorithm to get every edge incorrectly by making the degree of every element in $\{x_{n/2+1}, \ldots, x_n\}$ larger than the degree of every element in $\{x_1, \ldots, x_{n/2}\}$. The most efficient way to do that is to assign degree $n/2$ to all elements in the first cluster, and $(n-2)/2$ to all elements in the second cluster. The number of times $r(x_{k+1}, x_k)$ appears in the AUC regret is thus given by $(n/2)^2$. The edges in $G_{n/2}$ that the adversary can order correctly are the edges that go from $G^0_{n/2}$ into $G^1_{n/2}$. The sum of out-degrees of nodes in $G^0_{n/2}$ is $\frac{n-2}{2} \cdot \frac{n}{2}$, but $\binom{n/2}{2}$ have to be absorbed internally within $G^0_{n/2}$.

Thus the maximum number of edges the adversary can order correctly is at most

$$\frac{n-2}{2} \cdot \frac{n}{2} - \binom{n/2}{2} = \frac{n(\frac{n}{2} - 1)}{4}.$$

Thus the ratio is bounded by (cancelling the factor of 4)

$$\frac{n^2}{n^2 - n(\frac{n}{2} - 1)} = \frac{n^2}{\frac{n^2}{2} + n} = 2 - \frac{4}{n+2}.$$

∎

We prove the lemma used in the proof above.

Lemma 3.1. *For any x_i, x_j, and x_k in Q,*

$$r_Q(x_i, x_j) + r_Q(x_j, x_k) = r_Q(x_i, x_k).$$

Proof: Let d_{ijk} be a short-hand for the restriction of $D(Y_1, \ldots, Y_n \mid Q)$ to $\{Y_i, Y_j, Y_k\}$. A simple algebraic manipulation verifies the claim.

$$\begin{aligned} r_Q(x_i, x_j) + r_Q(x_j, x_k) & \\ &= d_{ijk}(100) + d_{ijk}(101) - d_{ijk}(010) - d_{ijk}(011) \\ &+ d_{ijk}(010) + d_{ijk}(110) - d_{ijk}(001) - d_{ijk}(101) \\ &= d_{ijk}(100) + d_{ijk}(110) - d_{ijk}(001) - d_{ijk}(011) = r_Q(x_i, x_k), \end{aligned}$$

Notice that all label assignments above have exactly two mixed pairs, so the factor of $1/2$ is cancelled. ∎

4 The Suboptimality of Minimization

5 Experiments

Ranking as a Tool

In practice, unlabeled data typically abound while labels are often hard to get. The setting described above can be used to significantly reduce the number of labeled examples needed for learning. Given a pool of unlabeled examples, we can use ranking for predicting the labels of all examples in the pool. The ranking itself obviously won't suffice since it contains

only relative information, but it exponentially reduces the number of examples that need to be labeled.

Suppose that the selector was error-free. We could use it to construct an ordering of examples with their hidden labels forming a sequence of 0s followed by a sequence of 1s. If we can adaptively query for the labels of $\lceil \log m \rceil$ examples, we can reconstruct all m labels exactly by simply doing a binary search to discover the transition point. Notice that we reduced the problem to the case when active learning gives an exponential improvement in the number of labeled examples needed. In [BBL05], we show how to extend active learning of a threshold function to the case of adversarial noise.

Theorem 2. [BBL05] Let $\epsilon < \frac{1}{2}$ be an acceptable approximation error and $\eta < \frac{\epsilon}{16}$ be the noise rate. For $\delta > 0$, we need at most at most $O\left(\ln \frac{\ln \frac{1}{\epsilon}}{\delta} \ln \frac{1}{\epsilon}\right)$ labeled samples to return a threshold function within error ϵ with probability at least $1 - \delta$.

6 Comparison with Previous Work

We emphasize that the style of analysis here is orthogonal to results such as generalization [SHD05, FIS+03] or large deviation bounds [**?**]. In these results, one learns a ranking function mapping instances onto the real line; the relative position of instances on the line defines a ranking. Generalization results bound the expected performance of the learned ranking function in terms of its empirical performance on the training sequence (and some complexity term). The pairwise preferences (or labeled instances) given to the algorithm are assumed to be correct.

We analyze the transformation of losses, expressing the quality of the produced ranking in terms of the performance of the pairwise ranking expert. This allows us to avoid making any assumptions necessary to prove traditional bounds (such as the independence of examples in the test set), making the guarantees directly applicable in practice.

In our case, to map a new instance, one needs to apply the ranking

Cortes and Mohri [CM04] give a statistical analysis of the relationship between the AUC and the 0/1 error rate *on the same classification problem*, treating the two as different loss functions. We, on the other hand, relate the expected ranking loss on the original problem to the 0/1 loss on a *different* problem (that of ordering pairs of instances). The goal of Cortes and Mohri was to better understand the AUC values produced by algorithms designed to optimize the 0/1 loss. They give expressions for the expected value and the standard deviation of the AUC over all classifications with a fixed number of errors; the expressions depend on the assumption that all such classifications are equiprobable. The assumption is clearly problematic, but together with empirical results, the derived expressions allowed them to find regimes when optimizing the AUC value directly should be preferred to optimizing the error rate.

7 Discussion

The basic goal of this work is to show that good performance on the induced pairwise classification problem implies good performance with respect to ranking. To understand why this is nontrivial it is worthwhile to consider some approaches which *do not* work. One approach which does not work is using a classifier to estimate class membership probability (as in [LZ05]) and then ranking according to the class membership probability. This approach fundamentally fails because in lopsided cases where the number of label 1 examples is very small, and a few large class membership probability estimation errors can greatly harm ranking metrics while only slightly affecting classification with respect to the induced distribution (see [LZ05] for a more specific example).

References

[ACF+02] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem, *SIAM Journal on Computing*, 32(1): 48–77, 2002.

[BBL05] M. F. Balcan, A. Beygelzimer, and J. Langford. Agnostic Active Learning, *NIPS05 Workshop on Foundations of Active Learning*.

[CSS99] W. Cohen, R. Schapire, and Y. Singer. Learning to order things, *Journal of Artificial Intelligence Research*, 10: 243–270, 1999.

[CFR06] D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments, *Proceeding of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 2006.

[CM04] C. Cortes and M. Mohri. AUC Optimization vs. Error Rate Minimization, *Advances in Neural Information Processing Systems* (NIPS 2003), 2004.

[FIS+03] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences, *Journal of Machine Learning Research*, 4: 933–969, 2003.

[LZ05] J. Langford and B. Zadrozny. Estimating Class Membership Probabilities Using Classifier Learners, AI+STATS 2005.

[RCM+05] C. Rudin, C. Cortes, M. Mohri, and R. Schapire. Margin-based ranking meets Boosting in the middle, *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory* (COLT), 2005.

[SHD05] S. Agarwal, S. Har-Peled, and D. Roth. A uniform convergence bound for the area under the ROC curve, *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics* (AISTATS), 2005.

[ZLA03] Bianca Zadrozny, John Langford, and Naoki Abe. Cost Sensitive Learning by Cost-Proportionate Example Weighting, *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, 435–442, 2003.