

# Robust Reductions from Ranking to Classification

Maria-Florina Balcan<sup>1</sup>, Nikhil Bansal<sup>2</sup>, Alina Beygelzimer<sup>2</sup>,  
Don Coppersmith<sup>3</sup>, John Langford<sup>4</sup>, and Gregory B. Sorkin<sup>2</sup>

<sup>1</sup> Carnegie Mellon University, Pittsburgh, PA  
`ninamf@cs.cmu.edu`

<sup>2</sup> IBM Thomas J. Watson Research Center, Yorktown Heights + Hawthorne, NY  
`{bansal,beygel,sorkin}@us.ibm.com`

<sup>3</sup> IDA Center for Communications Research, Princeton, NJ  
`dcopper@idaccr.org`

<sup>4</sup> Yahoo Research, New York, NY  
`jl@yahoo-inc.com`

**Abstract.** We reduce ranking, as measured by the Area Under the Receiver Operating Characteristic Curve (AUC), to binary classification. The core theorem shows that a binary classification regret of  $r$  on the induced binary problem implies an AUC regret of at most  $2r$ . This is a large improvement over naive approaches such as ordering according to regressed scores, which have a regret transform of  $r \rightarrow nr$  where  $n$  is the number of elements.

## 1 Introduction

We consider the problem of ranking a set of instances, robustly. In the most basic version, 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. A common measure of success for a ranking algorithm is the area under the ROC curve (AUC). The associated loss,  $1 - \text{AUC}$ , measures how many pairs of neighboring instances would have to be swapped to repair the ranking, normalized by the number of 0s times the number of 1s. The loss is zero precisely when all 0s are ordered before all 1s. It is greater for mistakes at the beginning and the end of an ordering, which satisfies the intuition that an unwanted item placed at the top of a recommendation list should have a higher associated loss than when placed in the middle.

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

These two problems appear quite different. For the classification loss function, a misclassified instance incurs the same loss independently of how other instances are classified. The AUC loss, on the other hand, is defined on sets of instances. It

is natural to ask whether we need fundamentally different algorithms to optimize these two loss functions. This paper shows that, in some precise sense, the answer is no. We prove that the problem of optimizing the AUC can be reduced to classification in such a way that a good performance on the classification problem implies a good performance on the AUC problem. We call a pair of instances *mixed* if they have different labels. The classification problem is to predict, given a random mixed pair of instances in the test set, whether the first instance should be ordered before the second. In other words, we show that there is a robust mechanism for translating any classifier learning algorithm into a ranking algorithm.

Several observations should help understand the setting and the result better.

**Relation to Regression and Classification:** A common way to generate a ranking is to order examples according to some regressed score or estimated conditional class probability. The problem with this approach is that it is not necessarily robust. The fundamental difficulty is exhibited by highly unbalanced test sets. If we have one 1 and many 0s, a point-wise (i.e., regression or classification) loss on the 1 with a perfect prediction for the 0s can greatly harm the AUC while only slightly affecting the point-wise loss with respect to the induced distribution. This observation implies that such schemes transform point-wise loss  $l$  to AUC loss  $nl$ , where  $n$  is the number of elements in the test set.

A similar observation holds for regrets in place of losses: point-wise regret  $r$  translates into AUC regret  $nr$ . *Regret* is the difference between the incurred loss and the lowest achievable loss on the problem (see Section 2 for a formal definition). The motivation for regret analysis is that it separates errors from the stochastic noise in the problem, so the bounds apply nontrivially even on problems with a large conditional noise.

Our core theorem shows that a pairwise classifier with a regret of  $r$  on pairs implies an AUC regret of at most  $2r$ , for arbitrary distributions over instances. (The same statement holds for losses.) Thus, for example, if the binary error rate is 20% due to inherent noise and 5% due to errors made by the classifier, then only the 5% would be (at most) doubled. Section 5 shows that this is the best possible. The theorem is a large improvement over the approaches discussed above, which have a dependence on  $nr$ . For comparison, the relationship of ranking to classification is functionally tighter than has been proven for regression to binary classification ( $r \rightarrow \sqrt{r}$ ) [LZ05].

**Relation to the Feedback Arc Set Problem:** Consider running a tournament on the set  $U$  we want to rank. Every element of  $U$  plays all other elements and the outcome of each play is determined by a classifier  $c$  trained to predict which of the two given instances should be ordered first.

What is the best way to rank the players in  $U$  from strongest to weakest?

The tournament induced by  $c$  on  $U$  does not have to be consistent with any linear ordering, while a ranking algorithm must predict an ordering. A natural

objective is to find an ordering which agrees with the tournament on as many player pairs as possible, i.e., minimizes the number of inconsistent pairs where a higher-ranked player (a player ordered closer to the beginning of the list) actually lost to a lower-ranked player. This is the NP-hard minimum feedback arc set problem in tournaments. (Although the hardness was conjectured for a long time, it has been proved only recently; see [A06].) A basic guarantee holds for a solution to this problem: If  $c$  makes at most  $k$  mistakes on winner–loser pairs, then the ordering minimizing the number of inconsistent pairs has at most  $2k$  winner–loser pairs where the loser comes before the winner (see Section 4). Section 5 exhibits a tournament matching this bound.

Another natural way to break cycles is to rank instances according to their number of wins in the tournament. (The way ties are broken is inessential; for definiteness, let us say they are broken against us.) Coppersmith, Fleischer, and Rudra [CFR06] proved that this algorithm provides a factor of 5-approximation for the feedback arc set problem. An approximation, however, does not generally imply any finite regret transform for the AUC problem. For example,  $c$  may make no mistakes on the 0–1 (i.e., winner–loser) pairs while inducing a non-transitive tournament on the 0s or the 1s, so an approximation that does not know the labeling can incur a non-zero number of 0–1 pair mistakes.

We prove, however, that the algorithm that orders the elements by their number of wins, transforms classification regret  $k$  into AUC regret of at most  $2k$ . In other words, this shows that there is an alternative to solving an NP-hard problem with the same optimality guarantee: Ordering by the number of wins has the *same* regret and loss transform as an optimal solution to the feedback arc set problem.

**Relation to Generalization Bounds:** A number of papers analyze generalization properties of ranking algorithms (see, e.g., [FIS+03,SHD05,SN05,RCM+05]). These results analyze ranking directly by estimating the rate of convergence of empirical estimates of the ranking loss to its expectation. The bounds typically involve some complexity parameter of the class of functions searched by the algorithms (which serves as a regularizer), and some additional quantities considered relevant for the analysis. The examples are assumed to be drawn independently from some fixed distribution and the labels are often assumed to be deterministic.

The type of results in this paper is different. We bound the realized AUC performance in terms of the realized classification performance. Since the analysis is relative, it does not have to rely on any assumptions about the way the world produces data. In particular, the bounds apply when there are arbitrary high-order dependencies between examples. This seems important in a number of real-world applications where ranking is of interest.

That said, our analysis does not say anything about the number of samples needed to achieve a certain level of performance. Instead it says that achieved performance can be robustly transferred from classification to ranking. (And

so, for example, any generalization result for the induced classification problem immediately implies a generalization result for the AUC problem.)

## 2 Preliminaries

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

$$e(c, P) = \Pr_{(x,y) \sim P}[c(x) \neq y].$$

The *classification regret* of  $c$  on  $P$  is defined as

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

**Ranking:** Let  $\pi : X \times X \rightarrow \{0, 1\}$  be a *preference function* that, given as input any two instances in  $X$ , outputs 1 if it agrees with the ordering of its arguments, and 0 otherwise. We say that  $\pi$  is an *ordering* of a set  $S$  if it is transitive on  $S$ , i.e., its pairwise preferences are consistent with some linear ordering of elements in  $S$ . The *AUC loss* of an ordering  $\pi$  on a set  $S \in (X \times \{0, 1\})^n$  is defined as

$$l(\pi, S) = \frac{\sum_{i \neq j} \mathbf{1}(y_i > y_j) \pi(x_i, x_j)}{\sum_{i < j} \mathbf{1}(y_i \neq y_j)}.$$

(Indices  $i$  and  $j$  in the summations range from 1 to  $n$ , and  $\mathbf{1}(\cdot)$  is the indicator function which is 1 if its argument is true, and 0 otherwise.)

A pair  $(x_1, y_1), (x_2, y_2)$  is called *mixed* if  $y_1 \neq y_2$ .

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

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

Note that  $D$  may encode arbitrary dependencies between examples.

The *AUC regret* of  $\pi$  on  $D$  is given by

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

We are interested in ranking algorithms that use a preference function as an oracle and are *robust* in the sense that a small classification regret of the oracle cannot induce a large AUC regret.

**Tournaments:** A *tournament* is a complete graph with no self-loops, in which each edge is directed one way or the other, so that for every pair of vertices  $i \neq j$ , either  $i \rightarrow j$  is an edge or  $j \rightarrow i$  is an edge, but not both. The edge  $i \rightarrow j$  says that  $i$  *beats*  $j$ ; so edges point from winners to losers. We adopt the

---

**Algorithm 1** AUC-TRAIN (labeled set  $S$ , binary learning algorithm  $A$ )

---

1. Let  $S' = \{ \langle (x_1, x_2), \mathbf{1}(y_1 < y_2) \rangle : (x_1, y_1), (x_2, y_2) \in S \text{ and } y_1 \neq y_2 \}$
  2. return  $c = A(S')$ .
- 

---

**Algorithm 2** DEGREE (unlabeled set  $U$ , pairwise classifier  $c$ )

---

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

convention that 0s should be ordered ahead of 1s, so 0s should ideally beat 1s. We write  $\deg(i)$  for the *outdegree* of vertex  $i$ , so  $\deg(i) = \sum_j \mathbf{1}(i \rightarrow j)$ , where the indicator function  $\mathbf{1}(i \rightarrow j)$  is 1 if  $i \rightarrow j$  is an edge and 0 otherwise. Thus we generally expect 0s to have large outdegree and 1s small outdegree; however, we allow and analyze arbitrary tournaments.

### 3 Ordering by the Number of Wins

In this section, we describe the reduction and prove the main result.

The reduction consists of two components. The training part, AUC-TRAIN (Algorithm 1), takes a set  $S$  of labeled examples of type  $X \times \{0, 1\}$  and transforms all mixed pairs in  $S$  into binary examples for the oracle learning algorithm. The binary classification problem induced by the reduction is to predict, given a random mixed pair of examples in  $S$ , whether the first example should be ordered before the second. For any process  $D$  generating datasets  $S$ , we can define the induced distribution over  $(X \times X) \times \{0, 1\}$  by first drawing  $S$  from  $D$ , and then drawing a random mixed pair from  $S$ . We denote this induced distribution by  $\text{AUC-TRAIN}(D)$  (admittedly overloading the notation).

The test part, DEGREE (Algorithm 2), uses the pairwise classifier  $c$  learned in Algorithm 1 to run a tournament on the test set  $U$ , and then ranks the elements of  $U$  in the decreasing order of their number of wins in the tournament, breaking ties arbitrarily. (Again, we expect 0s to beat 1s, and thus have larger outdegree.)

It is best to think of the classifier  $c$  as an adversary trying to induce a large AUC regret without paying much in classification regret: The adversary  $c$  specifies a tournament on  $U$ . There is some realized bipartition of  $U$  into a set of 0s and a set of 1s (drawn from the underlying conditional distribution of label sequences given  $U$ ). The bipartition is known to the adversary but unknown to the ranking algorithm. The adversary starts with a tournament of its choice where every 0 beats every 1, and she can invert (and is charged for) the outcomes of some games between a 0 and a 1. Again, the adversary can choose any subtournaments on the 0s and on the 1s for free. Unless we are extremely lucky, the resulting tournament is not consistent with any linear ordering of  $U$ . Given  $c$ 's tournament, DEGREE

orders the elements of  $U$ , possibly introducing additional mistakes, i.e., pairs where a 1 beats a 0. Our goal is to show that the reduction is robust in the sense that  $c$  cannot cause DEGREE to make many mistakes without making many mistakes itself.

The remainder of this section proves the following theorem.

**Theorem 1.** *For all distributions  $D$  and all pairwise classifiers  $c$ ,*

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

Note the quantification in the above theorem: it applies to *all* settings where Algorithms 1 and 2 are used; in particular,  $D$  may encode arbitrary dependences between examples.

*Proof.* Given an unlabeled test set  $U \in X^n$ , the joint distribution  $D$  induces a conditional distribution  $D(Y_1, \dots, Y_n \mid U)$  over the set of label sequences  $\{0, 1\}^n$ . In the remainder of the paper, let  $Q$  denote this conditional distribution. We identify  $U$  with  $\{1, \dots, n\}$ .

We prove the theorem for any fixed  $U$ , and then take the expectation over the draw of  $U$  at the end.

The first step is to rewrite both sides of the inequality in the theorem statement as sums over pairwise regrets. A *pairwise loss* is defined by

$$l_Q(i, j) = \mathbf{E}_{y^n \sim Q} \frac{\mathbf{1}(y_i > y_j)}{\sum_{u < v} \mathbf{1}(y_u \neq y_v)}.$$

It is the loss of ordering  $i$  before  $j$ . If  $l_Q(i, j) < l_Q(j, i)$ , the *regret*  $r_Q(i, j)$  of ordering  $i$  before  $j$  is 0; otherwise,  $r_Q(i, j) = l_Q(i, j) - l_Q(j, i)$ .

We can assume without loss of generality that the ordering minimizing the AUC loss (thus having zero AUC regret) is  $\langle 1 \ 2 \ \dots \ n \rangle$ . All regret-zero pairwise predictions must be consistent with the ordering; i.e., for all  $i < j$ , we have  $r_Q(i, j) = 0$ . (See Lemma 2 in Section A for a proof.)

Lemma 1 in Section A establishes a basic property of pairwise regrets: For any pair  $i < j$ , the regret  $r_Q(j, i)$  can be decomposed as

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

The AUC regret of  $\pi$  on  $Q$  can be decomposed as a sum of pairwise regrets:

$$\begin{aligned}
r_{\text{AUC}}(\pi, Q) &= l(\pi, Q) - \min_{\pi^*} l(\pi^*, Q) = \mathbf{E}_{y^n \sim Q} [l(\pi, U)] - \min_{\pi^*} \mathbf{E}_{y^n \sim Q} [l(\pi^*, U)] \\
&= \mathbf{E}_{y^n \sim Q} \frac{\sum_{i,j} \mathbf{1}(y_i > y_j) \pi(i, j)}{\sum_{u < v} \mathbf{1}(y_u \neq y_v)} - \min_{\pi^*} \mathbf{E}_{y^n \sim Q} \frac{\sum_{i,j} \mathbf{1}(y_i > y_j) \pi^*(i, j)}{\sum_{u < v} \mathbf{1}(y_u \neq y_v)} \\
&= \max_{\pi^*} \mathbf{E}_{y^n \sim Q} \frac{\sum_{i,j} [\mathbf{1}(y_i > y_j) \pi(i, j) - \mathbf{1}(y_i > y_j) \pi^*(i, j)]}{\sum_{u < v} \mathbf{1}(y_u \neq y_v)} \\
&= \sum_{i < j: \pi(j, i) = 1} r_Q(j, i) = \sum_{k=1}^{n-1} |\{i \leq k < j : \pi(j, i) = 1\}| \cdot r_Q(k+1, k).
\end{aligned}$$

The last equality follows from the repeated use of Lemma 1.

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

$$\begin{aligned}
r(c, \text{AUC-TRAIN}(Q)) &= e(c, \text{AUC-TRAIN}(Q)) - \min_{c^*} e(c^*, \text{AUC-TRAIN}(Q)) \\
&= \max_{c^*} \mathbf{E}_{y^n \sim Q} \left[ \frac{\sum_{i,j} [\mathbf{1}(y_i > y_j) c(i, j) - \mathbf{1}(y_i > y_j) c^*(i, j)]}{\sum_{u < v} \mathbf{1}(y_u \neq y_v)} \right] \\
&= \sum_{i < j: c(j, i) = 1} r_Q(j, i) = \sum_{k=1}^{n-1} |\{i \leq k < j : c(j, i) = 1\}| \cdot r_Q(k+1, k).
\end{aligned}$$

Let  $g_k$  and  $f_k$  denote the coefficients with which the term  $r_Q(k+1, k)$  appears in the above decompositions of  $r_{\text{AUC}}(\pi, Q)$  and  $r(c, \text{AUC-TRAIN}(Q))$  respectively. To complete the proof we need to show that  $g_k < 2f_k$  for each  $k$ .

Fix  $k$  and consider a bipartition of  $U$  into a set  $\{1, \dots, k\}$  of “winners” and a set  $\{k+1, \dots, n\}$  of “losers”. In this terminology,  $g_k$  is the number of winner-loser pairs where the loser has at least as many wins as the winner, and  $f_k$  is the number of winner-loser pairs where the loser beats the winner (in the tournament induced by  $c$  on  $U$ ). Theorem 2 below shows that  $g_k \leq 2f_k$ , completing this proof.  $\blacksquare$

Let  $T$  be a tournament and let the vertices of  $T$  be arbitrarily partitioned into a set  $W$  of “winners” and a set  $L$  of “losers”. Call the triple  $(T, W, L)$  a winner-loser partitioned tournament, and denote it by  $\mathbf{T}$ .

We will show that for any  $\mathbf{T}$ , the number of winner-loser pairs where the loser larger degree than the winner is at most twice the number of winner-loser pairs where the loser beats the winner. Formally, define two measures:

$$\begin{aligned}
g(\mathbf{T}) &= \sum_{\ell \in L} \sum_{w \in W} \mathbf{1}(\deg(\ell) \geq \deg(w)), \\
f(\mathbf{T}) &= \sum_{\ell \in L} \sum_{w \in W} \mathbf{1}(\ell \rightarrow w).
\end{aligned}$$

**Theorem 2.** *For any winner–loser partitioned tournament  $\mathbf{T}$ ,  $g(\mathbf{T}) \leq 2f(\mathbf{T})$ .*

Since the number of edges from  $L$  to  $W$  is equal to the total number of edges out of  $L$  minus the number of edges from  $L$  to  $L$ , we can rewrite

$$f(\mathbf{T}) = \sum_{\ell \in L} \sum_{w \in W} \mathbf{1}(\ell \rightarrow w) = \sum_{\ell \in L} \deg(\ell) - \binom{|L|}{2}.$$

Both  $f(\mathbf{T})$  and  $g(\mathbf{T})$  depend only on the degrees of the vertices of  $T$ , so rather than working with a (labeled) tournament, a relatively complex object, we can work with a (labeled) degree sequence.

Landau’s theorem [Lan53] says that there exists a tournament with outdegree sequence  $d_1 \leq d_2 \leq \dots \leq d_n$  if and only if, for all  $1 \leq i \leq n$ ,  $\sum_{j=1}^i d_j \geq \sum_{j=1}^i (j-1)$ , with equality for  $i = n$ .

Recall that a sequence  $\langle a_1, \dots, a_n \rangle$  is *majorized* by  $\langle b_1, \dots, b_n \rangle$  if the two sums are equal and if, when each sequence is sorted in non-increasing order, the prefix sums of the  $b$  sequence are larger than (dominate) those of the  $a$  sequence. (For a comprehensive treatment of majorization, see [MO79].) Landau’s condition is precisely that  $\langle d_1, \dots, d_n \rangle$  is majorized by  $\langle 0, \dots, n-1 \rangle$ . (With the sequences sorted in increasing order, Landau’s condition is that prefix sums of the degree sequence dominate those of the progression, which is the same as saying that the suffix sums of the degree sequence are dominated by the suffix sums of the progression.) This allows us to take advantage of well-known properties of majorization, notably that if  $A'$  is obtained by averaging together any elements of  $A$ , then  $A$  majorizes  $A'$ .

This allows us to re-state Theorem 2 in terms of a sequence and majorization, rather than a tournament, but first we relax the constraints. First, where the original statement requires elements of the degree sequence to be non-negative integers, we allow them to be non-negative reals. Second, the original statement requires that we attach a winner/loser label to each element of the degree sequence. Instead, we aggregate equal elements of the degree sequence, and for a degree  $d_i$  of (integral) multiplicity  $m_i$ , assign arbitrary non-negative (but not necessarily integral) portions to winners and losers:  $w_i + \ell_i = m_i$ . Note that the majorization condition applies only to the “compressed sequence”  $\{d_i, m_i\}$ , not the labeling.

**Theorem 3.** *For any winner–loser labeled compressed sequence  $\mathbf{D} = (D, W, L)$  where  $D$  is majorized by  $\langle 0, \dots, n-1 \rangle$ ,  $g(\mathbf{D}) \leq 2f(\mathbf{D})$ .*

Note that this implies theorem 2 because it is about the same maximum over a smaller set of possibilities.

*Proof.* We begin with an outline of the proof. Define a compressed sequence  $\mathbf{D}$  as being *canonical* if it consists of at most three degrees: a smallest degree  $d_1$



having only losers ( $w_1 = 0$ ), a middle degree  $d_2$  potentially with both winners and losers ( $w_2, \ell_2 \geq 0$ ), and a largest degree  $d_3$  having only winners ( $\ell_3 = 0$ ). We first establish that any canonical sequence has  $g(\mathbf{D}) - 2f(\mathbf{D}) \leq 0$ . We then show how to transform *any* degree sequence to a canonical one with a larger (or equal) value of  $g - 2f$ , which completes the argument.

We first show that a canonical sequence  $\mathbf{D}$  has  $g - 2f \leq 0$ . For the canonical configuration,  $g = w_2\ell_2$  and  $f = \ell_1d_1 + \ell_2d_2 - \frac{(\ell_1+\ell_2)(\ell_1+\ell_2-1)}{2}$ , and hence our goal is to show that

$$\ell_1d_1 + \ell_2d_2 \geq (\ell_1 + \ell_2)(\ell_1 + \ell_2 - 1)/2 + w_2\ell_2/2 \quad (1)$$

By Landau's condition applied to  $\ell_1$  and  $\ell_1 + w_2 + \ell_2$ , we have the following two relations:

$$\ell_1d_1 \geq \binom{\ell_1}{2} \quad (2)$$

and

$$\ell_1d_1 + (\ell_2 + w_2)d_2 \geq \binom{\ell_1 + w_2 + \ell_2}{2}. \quad (3)$$

Observe that  $\ell_1$  and  $\ell_1 + \ell_2 + w_2$  are both integers.

Multiplying (2) by  $w_2/(\ell_2 + w_2)$  and (3) by  $\ell_2/(\ell_2 + w_2)$  and adding them, we obtain that

$$\ell_1d_1 + \ell_2d_2 \geq \frac{1}{\ell_2 + w_2} \left( w_2 \binom{\ell_1}{2} + \ell_2 \binom{\ell_1 + \ell_2 + w_2}{2} \right). \quad (4)$$

A simple calculation shows that the right side of inequality (4) is exactly equal to the right hand side of (1). This proves that  $g \leq 2f$  for a canonical sequence.

If a sequence is not canonical then there are two consecutive degrees  $d_i$  and  $d_j$  ( $j = i + 1$ ) such that one of the cases 1a, 1b, or 2 (described below) holds. In each case we apply a transformation producing from the degree sequence  $\mathbf{D}$  a new degree sequence  $\mathbf{D}'$ , where:

- the total weight of winners in  $\mathbf{D}'$  is equal to that of  $\mathbf{D}$ ;
- the same is true for losers (and thus for the total weight);
- the total weight on each degree remains integral;
- $\mathbf{D}'$  maintains the majorization needed for Landau's theorem;
- the value of  $g - 2f$  is at least as large for  $\mathbf{D}'$  as for  $\mathbf{D}$ ; and
- either the number of nonzero values  $w_i$  and  $\ell_i$  or the number of distinct degrees  $d_i$  is strictly smaller for  $\mathbf{D}'$  than for  $\mathbf{D}$ , and the other is no larger for  $\mathbf{D}'$  than for  $\mathbf{D}$  (for Transformation 2, this may first require application of another transformation).

We first sketch the cases and then detail the transformations.

**Case 1a**  $d_i$  has only winners ( $l_i = 0$ ).

Apply Transformation 1a, combining the two degrees into one.

**Case 1b**  $d_j$  has only losers ( $w_j = 0$ ).

Apply Transformation 1b, combining the two degrees into one.

**Case 2** All of  $w_i$ ,  $l_i$ ,  $w_j$  and  $l_j$  are nonzero.

Apply Transformation 2, leaving the degrees the same but transforming the weights so that one of them is equal to 0 and one of the preceding cases applies, or the weights obey an equality allowing application of Transformation 3, which combines the two degrees into one.

### Transformation 1a:

In Case 1a, where  $d_i$  has only winners, change  $\mathbf{D}$  to a new sequence  $\mathbf{D}'$  by replacing the pair  $(d_i, w_i, 0)$ ,  $(d_j, w_j, l_j)$  by their “average”: the single degree  $(d', w', l')$ , where

$$w' = w_i + w_j, \quad l' = l_j, \quad d' = \frac{w_i d_i + (w_j + l_j) d_j}{w_i + w_j + l_j}.$$

The stated conditions on a transformation are easily checked. The total weight of winners is clearly preserved, as is the total weight of losers and the total degree (out-edges). Summing weights preserves integrality. The number of distinct degrees is reduced by one, and the number of nonzero weights may be decreased by one or may remain unchanged. The Landau majorization condition holds because  $\mathbf{D}'$ , as an averaging of  $\mathbf{D}$ , is majorized by it, and majorization is transitive. The only non-trivial condition is the non-decrease in  $g - 2f$ . The number of loser–winner pairs where the loser outranks the winner is increased by  $l_j w_j$ , so  $g(\mathbf{D}) \leq g(\mathbf{D}')$ . Also,  $f$  depends only on the total weight of losers (which is unchanged) and on the average degree of losers. This average degree would be unchanged if  $w_i$  were 0; since  $w_i \geq 0$ , the average degree may decrease. Thus  $f(\mathbf{D}) \geq f(\mathbf{D}')$ , and  $(g - 2f)(\mathbf{D}) \leq (g - 2f)(\mathbf{D}')$ , as desired.

### Transformation 1b:

Symmetrically to Transformation 1a, obtain  $\mathbf{D}'$  by replacing the pair of labeled weighted degrees  $(d_i, w_i, l_i)$  and  $(d_j, 0, l_j)$  with a single one  $(d', w', l')$ , where

$$w' = w_i, \quad l' = l_i + l_j, \quad d' = \frac{(l_i + w_i) d_i + l_j d_j}{l_i + w_i + l_j}.$$

### Transformation 2:

Where  $w_i$ ,  $l_i$ ,  $w_j$  and  $l_j$  are all nonzero, we begin with one case, which leads to one other. In the usual case, we transform  $\mathbf{D}$  to  $\mathbf{D}'$  by replacing the pair  $(d_i, w_i, l_i)$ ,  $(d_j, w_j, l_j)$  with

$$(d_i, w_i + x, l_i - x), \quad (d_j, w_j - x, l_j + x),$$

for some value of  $x$  (positive or negative) to be determined. This affects only the labeling, not the weighted degree sequence itself, and is therefore legitimate as long as the four quantities  $w_i + x$ ,  $l_i - x$ ,  $w_j - x$  and  $l_j + x$  are all non-negative.

Defining  $\Delta = (g - 2f)(\mathbf{D}') - (g - 2f)(\mathbf{D})$ , we wish to choose  $x$  to make  $\Delta > 0$ .

$$\begin{aligned} \Delta &= \left\{ [(l_j + x)(w_i + x + w_j - x) + (l_i - x)(w_i + x)] - [l_j(w_i + w_j) + l_i w_i] \right\} \\ &\quad - 2 \left\{ [(l_i - x)d_i + (l_j + x)d_j] - [l_i d_i + l_j d_j] \right\} \\ &= x(w_j + l_i - 2(d_j - d_i) - x) = x(a - x), \end{aligned}$$

where  $a = w_j + l_i - 2(d_j - d_i)$ . This is a simple quadratic expression with negative coefficient on  $x^2$ , so its value increases monotonically as  $x$  is varied from 0 to  $a/2$ , where the maximum is obtained. (Note that  $a$  may be negative.) If  $a = 0$  then this transformation makes no change, and instead we use Transformation 3, below. Otherwise, vary  $x$  from 0 to  $a/2$  stopping when  $x$  reaches  $a/2$  or when any of  $w_i + x$ ,  $l_i - x$ ,  $w_j - x$  and  $l_j + x$  becomes 0; call this value  $x^*$ .

If any of  $w_i + x$ ,  $l_i - x$ ,  $w_j - x$  and  $l_j + x$  is 0 then the number of nonzero weights is decreased (while the number of distinct degrees is unchanged). Otherwise,  $x^* = a/2$ . In that case, the new  $\mathbf{D}'$  has  $a = 0$  (because another application of the same procedure would give  $x^* = 0$ ). With  $a = 0$  we apply Transformation 3, which reduces the number of nonzero weights.

### Transformation 3

Similar to Cases 1a and 1b, transform  $\mathbf{D}$  to  $\mathbf{D}'$  by replacing the pair  $(d_i, w_i, l_i)$ ,  $(d_j, w_j, l_j)$  with a single degree  $(d', w', l')$  that is their weighted average,

$$w' = w_i + w_j, \quad l' = l_i + l_j, \quad d' = \frac{(w_i + l_i)d_i + (w_j + l_j)d_j}{w_i + l_i + w_j + l_j}.$$

This gives

$$\begin{aligned} \Delta &= (g - 2f)(\mathbf{D}') - (g - 2f)(\mathbf{D}) \\ &= (l_i w_j) - 2(l_i d' + l_j d' - l_i d_i - l_j d_j) \\ &= l_i w_j + \frac{2(d_j - d_i)(w_i l_j - w_j l_i)}{w_i + l_i + w_j + l_j}. \end{aligned}$$

We apply this transformation only in the case where Transformation 2 fails to give any improvement because its “ $a$ ” expression is equal to 0, i.e.,  $d_j - d_i = (w_j + l_i)/2$ . Making the corresponding substitution gives

$$\begin{aligned}\Delta &= l_i w_j + \frac{(w_j + l_i)(w_i l_j - w_j l_i)}{w_i + l_i + w_j + l_j} \\ &= \frac{(l_i w_j)(l_j + w_i) + (l_j w_i)(l_i + w_j)}{w_i + l_i + w_j + l_j} \\ &> 0.\end{aligned}$$

This reduces the number of distinct degrees by one, without increasing the number of nonzero weights.

Concluding the argument, we have shown that any non-canonical configuration  $\mathbf{D}$  can be replaced by a configuration with a strictly smaller total of distinct degrees and nonzero weights, and at least as large a value of  $g - 2f$ . Since  $\mathbf{D}$  had at most  $n$  distinct degrees and  $2n$  nonzero weights originally, a canonical configuration  $\mathbf{D}^*$  is reached after at most  $3n - 1$  transformations. (All that is important is that the number of transformations is finite: that a canonical configuration is eventually reached.) Then,  $(g - 2f)(\mathbf{D}) \leq (g - 2f)(\mathbf{D}^*) \leq 0$ . ■

## 4 An Upper Bound for the Minimum Feedback Arc Set Ordering

This section shows an analog of Theorem 2 for an optimal solution to the feedback arc set problem. (The decomposition argument in Theorem 1 is algorithm-independent and applies here as well.) For a tournament  $T$  and an ordering  $\pi$ , a *back edge* is an edge  $i \rightarrow j$  in  $T$  such that  $j$  is ordered before  $i$  in  $\pi$ . Let  $\text{back}(T, \pi)$  denote the number of back edges induced by  $\pi$  in  $T$ .

For a winner–loser partitioned tournament  $\mathbf{T} = (T, W, L)$  and any minimum feedback arc set ordering  $\pi$  of  $T$ , let  $g'(\mathbf{T}, \pi)$  be the number of winner–loser pairs where the loser comes before the winner in  $\pi$ , and as before let

$$f(\mathbf{T}) = \sum_{\ell \in L} \sum_{w \in W} \mathbf{1}(\ell \rightarrow w)$$

be the number of such winner–loser pairs where the loser beats the winner.

**Theorem 4.** *For any winner–loser partitioned tournament  $\mathbf{T} = (T, W, L)$  and any minimum feedback arc set ordering  $\pi$  of  $T$ ,  $g'(\mathbf{T}, \pi) \leq 2f(\mathbf{T})$ .*

*Proof.* Let  $k_w$  be the smallest possible number of back edges in the subtournament induced by  $W$ . Define  $k_l$  similarly for the subtournament induced by  $L$ . Let  $k_w^\pi$  and  $k_l^\pi$  be the number of back edges in  $\pi$  that go from  $W$  to  $W$  and

from  $L$  to  $L$ , respectively. Denote the number of remaining (i.e., winner–loser or loser–winner) back edges in  $\pi$  by  $k_o^\pi$ .

Consider another ordering  $\sigma$  where all winners are ordered before all losers, and both the winners and the losers are ordered optimally among themselves with  $k_w$  and  $k_l$  back edges respectively. The number of back edges in  $\sigma$  is  $\text{back}(\mathbf{T}, \sigma) = k_w + k_l + f(\mathbf{T})$ . But we also have  $\text{back}(\mathbf{T}, \sigma) \geq \text{back}(\mathbf{T}, \pi)$  since  $\pi$  minimizes the number of back edges, and thus  $k_w + k_l + f(\mathbf{T}) \geq k_w^\pi + k_l^\pi + k_o^\pi$ . Since  $k_w \leq k_w^\pi$  and  $k_l \leq k_l^\pi$  by definition of  $k_w$  and  $k_l$ , we have  $f(\mathbf{T}) \geq k_o^\pi$ .

Consider any winner–loser pair with the loser ordered before the winner. If  $w \rightarrow l$  is the edge, it is a back edge in  $\pi$  and thus is counted by  $k_o^\pi$ . If  $l \rightarrow w$  is the edge instead, it is counted by  $f(\mathbf{T})$ . Thus  $g'(\mathbf{T}, \pi)$  is at most  $k_o^\pi + f(\mathbf{T})$ . Since  $f(\mathbf{T}) \geq k_o^\pi$ , this number is never more than  $2f(\mathbf{T})$ , which implies  $g'(\mathbf{T}, \pi) \leq 2f(\mathbf{T})$ . ■

## 5 Lower Bounds

We start with a lower bound for the algorithm that orders by the number of wins. Recall that  $f$  denotes the number of winner–loser pairs where the loser beats the winner, and  $g$  is the number of winner–loser pairs where the loser outranks the winner. The example below shows that there exists an infinite family of tournaments with  $g = 2f$ .

**Example 1.** With  $n$  odd, let every vertex have degree  $(n-1)/2$ ; note that the degree sequence  $\langle \frac{n-1}{2}, \dots, \frac{n-1}{2} \rangle$  does indeed respect Landau's condition, so it is realizable as a tournament. Label  $(n-1)/2$  of the vertices as winners and  $(n+1)/2$  as losers. With ties broken against us, all winners are ordered after all losers. This gives  $f = \frac{n+1}{2} \cdot \frac{n-1}{2} - \binom{(n+1)/2}{2} = (n+1)(n-1)/8$ , while  $g = \frac{n+1}{2} \cdot \frac{n-1}{2} = (n+1)(n-1)/4 = 2f$ . (A similar example gives a lower bound of  $2 - O(1/n)$  with ties broken optimally.)

The following construction gives an infinite family of tournaments for which an optimal solution to the feedback arc set problem has  $g \geq (2-\epsilon)f$ , for any  $\epsilon > 0$ .

**Example 2.** Set  $\delta = \frac{\epsilon}{1-\epsilon}$ , and let the set of vertices be partitioned into three components,  $V_1$ ,  $V_2$ , and  $V_3$ , with  $|V_1| = \delta n^2$ ,  $|V_2| = 2n^2$ , and  $|V_3| = n$ , for a sufficiently large  $n$ . The vertices in  $V_1 \cup V_2$  are the winners, the vertices in  $V_3$  are the losers.

The edges within each of the three components form acyclic tournaments. The cross-component edges are defined as follows: All edges between  $V_3$  and  $V_1$  point from  $V_3$  to  $V_1$ , and all edges between  $V_1$  and  $V_2$  point from  $V_1$  to  $V_2$ . To define the edges between  $V_2$  and  $V_3$ , divide  $V_2$  into  $2n$  consecutive blocks  $B_1, \dots, B_{2n}$  of  $n$  vertices each, such that edges point from  $B_i$  to  $B_j$  where  $1 \leq i < j \leq 2n$ . If  $i$  is odd, all edges from  $B_i$  point to  $V_3$ ; otherwise all edges from  $V_3$  point to  $B_i$ .

We have  $f = (1+\delta)n^3$ . What is the value of  $g$  induced by an ordering minimizing the number of back edges? Any such ordering must put  $V_1$  before  $V_2$ ; otherwise

we would have  $\Omega(n^4)$  back edges while an optimal ordering does not need to have more than  $O(n^3)$  such edges. Now, the tournament induced by  $V_2 \cup V_3$  has  $n^3$  edge-disjoint cycles of length 3 since there are  $n^2$  such cycles for every pair of blocks in  $V_2$  (and there are  $n$  disjoint pairs). There has to be at least one back edge for every such cycle, so any ordering must have at least  $n^3$  back edges. The ordering that puts  $V_3$  before  $V_2$  is thus optimal since it has exactly  $n^3$  back edges. Thus the ordering  $\{V_3, V_1, V_2\}$  minimizes the number of back edges. This ordering has  $(2 + \delta)n^3$  pairs where the loser is ordered before the winner, implying the bound  $g \geq (2 - \frac{\delta}{1+\delta})f = (2 - \epsilon)f$ .

## 6 Practicality

The reductions analysis is representation independent, which implies that it works for any representation. Naturally, some representations are more computationally efficient than others. If, for example,  $c(x_i, x_j) = \mathbf{1}(s(x_i) > s(x_j))$  for some learned score function  $s : X \rightarrow [0, 1]$ , the complexity of test-time evaluation is linear rather than quadratic in the number of elements. Note that  $s$  is not trained as a simple regressor, because what we want to optimize is the pairwise ordering of elements.

## 7 Relation to Other Work

Cortes and Mohri [CM04] analyzed the relationship between the AUC and the error rate on the same classification problem, treating the two as different loss functions. They derived expressions for the expected value and the standard deviation of the AUC over all classifications with a fixed number of errors, under the assumption that all such classifications are equiprobable (i.e., the classifier is as likely to err on any one example as on any other). These expressions are of little relevance to the results presented here.

Cohen, Schapire, and Singer [CSS99], similarly, use a two-stage approach to ranking: They first learn a preference function that takes a pair of instances and returns a score predicting how certain it is that the first instance should be ranked before the second. The learned function is then evaluated on all pairs of instances in the test set and an ordering approximating<sup>5</sup> the largest  $l_1$  agreement possible with the predictions is created. They show that the agreement achieved by an optimal ordering is at most twice the agreement obtained by their algorithm. To translate this result into the language of losses, let MFA be the AUC loss of a minimum feedback arc set ordering and APPROX be the AUC loss of

---

<sup>5</sup> The approximation algorithm they use orders by the weighted sum of wins minus the weighted sum of losses, in the induced tournament obtained by eliminated instances that have already been ordered.

the approximation. Then the result says that  $1 - \text{APPROX} \geq \frac{1}{2}(1 - \text{MFA})$  or  $\text{APPROX} \leq \frac{1}{2} + \text{MFA}/2$ . The settings are different making the results given here difficult to compare. Applying the result in our setting requires specializations and yields results that are weaker than ours.

## References

- [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*, 2005.
- [SN05] S. Agarwal, P. Niyogi. Stability and Generalization of Bipartite Ranking Algorithms, *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory (COLT)*, 2005.
- [A06] N. Alon. Ranking tournaments, *SIAM Journal on Discrete Mathematics* **20**: 137–142, 2006.
- [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)*, 776–782, 2006.
- [CM04] C. Cortes and M. Mohri. AUC Optimization vs. Error Rate Minimization, *Advances in Neural Information Processing Systems (NIPS)*, 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.
- [Lan53] H. G. Landau. On Dominance Relations and the Structure of Animal Societies: III. The Condition for a Score Structure, *Bull. Math. Biophys.*, **15**: 143–148, 1953.
- [LZ05] J. Langford and B. Zadrozny. Estimating Class Membership Probabilities Using Classifier Learners, *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.
- [MO79] A. Marshall and I. Olkin. *Inequalities: Theory of majorization and its applications*, Mathematics in Science and Engineering, 143, New York, 1979.
- [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.

## A Supporting Lemmas

The proof of Theorem 1 used two simple lemmas which we prove here. Recall that  $Q$  denotes the distribution of label sequences  $\{0, 1\}^n$  of an unlabeled set  $\{1, \dots, n\}$ .

**Lemma 1.** *For any  $i, j$ , and  $k$  in  $\{1, \dots, n\}$ ,*

$$r_Q(i, j) + r_Q(j, k) = r_Q(i, k).$$

*Proof.* Let  $p$  be a short-hand for the restriction of  $Q$  to indices  $\{i, j, k\}$  (i.e., thus  $p$  is a distribution over  $\{0, 1\}^3$  obtained by summing over all label indices other than  $i, j$ , or  $k$ ). A simple algebraic manipulation verifies the claim.

$$\begin{aligned} r_Q(i, j) + r_Q(j, k) &= \\ p(100) + p(101) - p(010) - p(011) + p(010) + p(110) - p(001) - p(101) &= \\ p(100) + p(110) - p(001) - p(011) &= r_Q(i, k). \end{aligned}$$

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

**Lemma 2.** *If, with respect to  $Q$ , the ordering  $\langle 1 2 \dots n \rangle$  is the ordering minimizing the AUC loss (thus having zero AUC regret), then all regret-zero pairwise predictions must be consistent with the ordering; i.e.,  $r_Q(i, j) = 0$  for all  $1 \leq i < j \leq n$ .*

*Proof.* All regrets  $r_Q(i, i+1)$  must be 0, since swapping  $i$  and  $i+1$  does not affect other pairwise regrets and would thus decrease the overall regret, contradicting the assumption that  $1 2 \dots n$  is regret minimizing. We prove that all other pairwise regrets must be 0 recursively. Consider a triple,  $i < j < k$  with  $r_Q(i, j) = 0$  and  $r_Q(j, k) = 0$ . We show that  $r_Q(i, k) = 0$ . Let  $p$  be defined as in the proof of Lemma 1. We have  $l_Q(i, j) \leq l_Q(j, i)$  and  $l_Q(j, k) \leq l_Q(k, j)$ , which can be expanded as

$$p(100) + p(101) \leq p(010) + p(011), \quad (5)$$

$$p(010) + p(110) \leq p(001) + p(101). \quad (6)$$

We now have

$$\begin{aligned} l_Q(i, k) &= p(100) + p(110) \leq p(100) + p(101) + p(010) + p(110) \\ &\leq p(010) + p(011) + p(001) + p(101) \leq p(001) + p(011) = l_Q(k, i), \end{aligned}$$

where the second inequality follows by adding (5) and (6). Thus  $l_Q(i, k) \leq l_Q(k, i)$  implying  $r_Q(i, k) = 0$ . ■