An Efficient Reduction of Ranking to Classification

Nir Ailon[1] and Mehryar Mohri[2,1]

[1] Google Research,
76 Ninth Avenue, New York, NY 10011,
`nailon@google.com`.
[2] Courant Institute of Mathematical Sciences,
251 Mercer Street, New York, NY 10012,
`mohri@cims.nyu.edu`.

Abstract. This paper describes an efficient reduction of the learning problem of ranking to binary classification. The reduction guarantees an average pairwise misranking regret of at most that of the binary classifier regret, improving a recent result of Balcan et al which only guarantees a factor of 2. Moreover, our reduction applies to a broader class of ranking loss functions, admits a simpler proof, and the expected running time complexity of our algorithm in terms of number of calls to a classifier or preference function is improved from $\Omega(n^2)$ to $O(n \log n)$. In addition, when the top k ranked elements only are required ($k \ll n$), as in many applications in information extraction or search engines, the time complexity of our algorithm can be further reduced to $O(k \log k + n)$. Our reduction and algorithm are thus practical for realistic applications where the number of points to rank exceeds several thousands. Much of our results also extend beyond the bipartite case previously studied.
Our rediction is a randomized one. To complement our result, we also derive lower bounds on any deterministic reduction from binary (preference) classification to ranking, implying that our use of a randomized reduction is essentially necessary for the guarantees we provide.

1 Introduction

The learning problem of ranking arises in many modern applications, including the design of search engines, information extraction, and movie recommendation systems. In these applications, the ordering of the documents or movies returned is a critical aspect of the system.

The problem has been formulated within two distinct settings. In the *score-based setting*, the learning algorithm receives a labeled sample of pairwise preferences and returns a *scoring function* $f : U \to \mathbb{R}$ which induces a linear ordering of the points in the set U. Test points are simply ranked according to the values of h for those points. Several ranking algorithms, including RankBoost [13, 21], SVM-type ranking [17], and other algorithms such as PRank [12, 2], were

designed for this setting. Generalization bounds have been given in this setting for the pairwise misranking error [13, 1], including margin-based bounds [21]. Stability-based generalization bounds have also been given in this setting for wide classes of ranking algorithms both in the case of bipartite ranking [2] and the general case [11, 10].

A somewhat different two-stage scenario was considered in other publications starting with Cohen, Schapire, and Singer [8], and later Balcan et al. [6], which we will refer to as the *preference-based setting*. In the first stage of that setting, a *preference function* $h : U \times U \mapsto [0, 1]$ is learned, where values of $h(u, v)$ closer to one indicate that v is ranked above u and values closer to zero the opposite. h is typically assumed to be the output of a classification algorithm trained on a sample of labeled pairs, and can be for example a convex combination of simpler preference functions as in [8]. A crucial difference with the score-based setting is that, in general, the preference function h does not induce a linear ordering. The order it induces may be non-transitive, thus we may have for example $h(u, v) = h(v, w) = h(w, u) = 1$ for three distinct points u, v, and w. To rank a test subset $V \subset U$, in the second stage, the algorithm orders the points in V by making use of the preference function h learned in the first stage.

This paper deals with the preference-based ranking setting just described. The advantage of this setting is that the learning algorithm is not required to return a linear ordering of all points in U, which is impossible to achieve faultlessly in accordance with a true pairwise preference labeling that is non-transitive. This is more likely to be achievable exactly or with a better approximation when the algorithm is requested instead, as in this setting, to supply a linear ordering, only for a limited subset $V \subset U$.

When the preference function is learned by a binary classification algorithm, the preference-based setting can be viewed as a reduction of the ranking problem to a classification one. The second stage specifies how the ranking is obtained using the preference function.

Cohen, Schapire, and Singer [8] showed that in the second stage of the preference-based setting, the general problem of finding a linear ordering with as few pairwise misrankings as possible with respect to the preference function h is NP-complete. The authors presented a greedy algorithm based on the tournament *degree* for each element $u \in V$ defined as the difference between the number of elements u is preferred to versus the number of those preferred to u. The bound proven by these authors, formulated in terms of the pairwise disagreement loss l with respect to the preference function h, can be written as $l(\sigma_{greedy}, h) \leq 1/2 + l(\sigma_{optimal}, h)/2$, where $l(\sigma_{greedy}, h)$ is the loss achieved by the permutation σ_{greedy} returned by their algorithm and $l(\sigma_{optimal}, h)$ the one achieved by the optimal permutation $\sigma_{optimal}$ with respect to the preference function h. This bound was given for the general case of ranking, but in the particular case of bipartite ranking (which we define below), a random ordering can achieve a pairwise disagreement loss of $1/2$ and thus the bound is not informative.

More recently, Balcan et al [6] studied the bipartite ranking problem and showed that sorting the elements of V according to the same tournament degree used by [8] guarantees a pairwise misranking regret of at most $2r$ using a binary classifier with regret r. However, due to the quadratic nature of the definition of the tournament degree, their algorithm requires $\Omega(n^2)$ calls to the preference function h, where $n = |V|$ is the number of objects to rank.

We describe an efficient algorithm for the second stage of preference-based setting and thus for reducing the learning problem of ranking to binary classification. We improve on the recent result of Balcan et al. [6], by guaranteeing an average pairwise misranking regret of at most r using a binary classifier with regret r. In other words, we improve their constant from 2 to 1. Our reduction applies (with different constants) to a broader class of ranking loss functions, admits a simpler proof, and the expected running time complexity of our algorithm in terms of number of calls to a classifier or preference function is improved from $\Omega(n^2)$ to $O(n \log n)$. Furthermore, when the top k ranked elements only are required ($k \ll n$), as in many applications in information extraction or search engines, the time complexity of our algorithm can be further reduced to $O(k \log k + n)$. Our reduction and algorithm are thus practical for realistic applications where the number of points to rank exceeds several thousands. Much of our results also extend beyond the bipartite case previously studied by [6] to the general case of ranking. A by-product of our proofs is also a bound on the pairwise disagreement loss with respect to the preference function h that we will compare to the result given by Cohen, Schapire, and Singer [8].

The algorithm used by Balcan et al. [7] to produce a ranking based on the preference function is known as sort-by-degree and has been recently used in the context of minimizing the feedback arcset in tournaments [9]. Here, we use a different algorithm, QuickSort, which has also been recently used for minimizing the feedback arcset in tournaments [4, 3]. The techniques presented make use of the earlier work by Ailon et al. on combinatorial optimization problems over rankings and clustering [4, 3].

The remainder of the paper is structured as follows. In Section 2, we introduce the definitions and notation used in future sections and introduce a family of general loss functions that can be used to measure the quality of a ranking hypothesis. Section 3 describes a simple and efficient algorithm for reducing ranking to binary classification, proves several bounds guaranteeing the quality of the ranking produced by the algorithm, and shows the running-time complexity of our algorithm to be very efficient. In Section 4 we discuss the relationship of the algorithm and its proof with previous related work in combinatorial optimization. In Section ?? we derive a lower bound of factor 2 on any deterministic reduction from binary (preference) classification to ranking, implying that our use of a randomized reduction is essentially necessary for the improved guarantees we provide.

2 Preliminaries

This section introduces several preliminary definitions necessary for the presentation of our results. In what follows, U will denote a universe of elements (e.g. the collection of all possible query-result pairs returned by a web search task) and $V \subseteq U$ will denote a small subset thereof (e.g. a preliminary list of relevant results for a given query). For simplicity of notation we will assume that U is a set of integers, so that we are always able to choose a minimal (canonical) element in a finite subset (as we do in (9) below). This arbitrary ordering should not be confused with the ranking problem we are considering.

2.1 General Definitions and Notation

We first briefly discuss the learning setting and assumptions made by Balcan et al.'s [7] and Cohen et al. [8] and introduce a consistent notation to make it easier to compare our results with that of this previous work.

Ground truth In [7], the *ground truth* is a *bipartite ranking* of the set V of elements that one wishes to rank.[3] A bipartite ranking is a partition of V into *positive* and *negative* elements where positive elements are preferred over negative ones and elements sharing the same label are in a tie. This is a natural setting when the human raters assign a positive or negative label to each element. Here, we will allow a more general structure where the ground truth is a ranking σ^* equipped with a weight function ω, which can be used for encoding ties. The bipartite case can be encoded by choosing a specific ω as we shall further discuss below.

In [8], the "ground truth" has a different interpretation, which we briefly discuss in Section 3.4.

Preference function In both [8] and [7], a preference function $h : U \times U \to [0,1]$ is assumed, which is learned in a first learning stage. The convention is that the higher $h(u,v)$ is, the more our belief that u should be ahead of v. The function h satisfies *pairwise consistency*: $h(u,v) + h(v,u) = 1$, but need not even be transitive on 3-tuples. The second stage uses h to output a proper ranking σ, as we shall further discuss below. The running time complexity of the second stage is measured with respect to the number of calls to h.

Output of learning algorithm The final output of the second stage of the algorithm, σ, is a proper ranking of V. Its cost is measured differently in [7] and [8]. In [7], it is measured against σ^* and compared to the cost of h against

---

[3] More generally, the ground truth may also be a distribution of bipartite rankings, but the error bounds both in our work and that of previous work are achieved by fixing one ground truth and taking conditional expectation as a final step. Thus, we can assume that it is fixed.

σ^*. This can be thought of as the best one could hope for if h encodes all the available information. In [8], σ is measured against the given preference function h, and compared to the best one can get.

2.2 Loss Functions

We are now ready to define the loss functions used to measure the quality of an output ranking σ either with respect to σ^* (as in [7]) or with respect to h (as in [8]).

Let $V \subseteq U$ be a finite subset that we wish to rank and let $S(V)$ denote the set of *rankings* on V, that is the set of injections from V to $[n] = \{1, \ldots, n\}$, where $n = |V|$. If $\sigma \in S(V)$ is such a ranking, then $\sigma(u)$ is the rank of an element $u \in V$, where "lower" is interpreted as "ahead". More precisely, we say that u is preferred over v with respect to σ if $\sigma(u) < \sigma(v)$. For compatibility with the notation used for general preference functions, we also write $\sigma(u, v) = 1$ if $\sigma(u) < \sigma(v)$ and $\sigma(u, v) = 0$ otherwise.

The following general loss function L_ω measures the quality of a ranking σ with respect to a desired one σ^* using a weight function ω (described below):

$$L_\omega(\sigma, \sigma^*) = \binom{n}{2}^{-1} \sum_{u \neq v} \sigma(u, v) \sigma^*(v, u) \omega(\sigma^*(u), \sigma^*(v)) . \tag{1}$$

The sum is over all pairs u, v in the domain V of the rankings σ, σ^*. It counts the number of inverted pairs $u, v \in V$ weighed by ω, which assigns importance coefficients to pairs, based on their positions in σ^*. The function ω must satisfy the following three natural axioms, which will be necessary in our analysis:

(P1) Symmetry: $\omega(i, j) = \omega(j, i)$ for all i, j;
(P2) Monotonicity: $\omega(i, j) \leq \omega(i, k)$ if either $i < j < k$ or $i > j > k$;
(P3) Triangle inequality: $\omega(i, j) \leq \omega(i, k) + \omega(k, j)$.

This definition is very general and encompasses many useful, well studied distance functions. Setting $\omega(i, j) = 1$ for all $i \neq j$ yields the unweighted pairwise misranking measure or the so-called Kemeny distance function.

For a fixed integer k, the following function

$$\omega(i, j) = \begin{cases} 1 & \text{if } ((i \leq k) \vee (j \leq k)) \wedge (i \neq j) \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

can be used to emphasize ranking at the top k elements. Misranking of pairs with one element ranked among the top k is penalized by this function. This can be of interest in applications such as information extraction or search engines where the ranking of the top documents matters more. For this emphasis function, all elements ranked below k are in a tie. In fact, it is possible to encode any tie relation using ω.

The loss function considered in [6] can also be straightforwardly encoded with the following emphasis *bipartite* function

$$\omega(i,j) = \begin{cases} 1 & (i \leq k) \wedge (j > k) \\ 1 & (j \leq k) \wedge (i > k) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Items in positions $1, \ldots, k$ for the permutation σ can be thought of as the positive items (in a tie), and those in $k+1, \ldots, |V|$ as negative (also in a tie). This choice coincides with $(1-\text{AUC})$, where AUC is the area under the ROC curve commonly used in statistics and machine learning problems [14, 19].

Clearly, setting $\omega(i,j) = |s(i)-s(j)|$ for any monotone *score* function s works as well. It is well known though that such a function can in fact be expressed as a convex combination of functions of the type (3). Hence, a bipartite function should be thought of as the simplest

In general, one may wish to work with a collection of ground truths $\sigma_1^*, \ldots, \sigma_N^*$ and weight functions $\omega_1, \ldots, \omega_N$ and a loss function which is a sum over the individual losses with respect to σ_i^*, ω_i, e.g. in meta searching[4]. Since our bound is based on the expected loss, it will straightforwardly generalize to this setting using the linearity of expectation. Thus, we can assume without loss of generality a single ground truth σ^* equipped with a single ω.

Preference Loss Function We need to extend the definition to measure the loss of a preference function h with respect to σ^*. Recall that $h(u,v)$ is In contrast with the loss function just defined, we need to define a *preference loss* measuring a ranking's disagreements with respect to a preference function h. When measured against σ^*, ω, the function L_ω can be readily used:

$$L_\omega(h, \sigma^*) = \binom{n}{2}^{-1} \sum_{u \neq v} h(u,v)\sigma^*(v,u)\omega(\sigma^*(u), \sigma^*(v)) . \tag{4}$$

We use L to denote L_ω for the special case where ω is the constant function 1, $\omega = 1$:

$$L(h, \sigma^*) = \binom{n}{2}^{-1} \sum_{u \neq v} h(u,v)\sigma(v,u) . \tag{5}$$

The special case of L coincides with the standard pairwise disagreement loss of a ranking with respect to a preference function as used in [8].

2.3 The Special Bipartite Case

A particular case of interest is when ω belongs to the family of weight functions defined in (3). For this particular case we will use a slightly more convenient

---

[4] The reason we have separate weight function ω's is e.g. each search engine may output top-k outputs for different values of k.

notation. For a set of elements $V \subseteq U$, Let Π_V denote the set of partitions of V into two sets (positive and negative). More formally, $\tau \in \Pi_V$ is a function from V to $\{0, 1\}$ (where 0 should be thought of as the *preferred* or *positive* value, and 1 the negative; we choose this convention so that $\tau(u)$ can be interepreted as the *rank* of u, where there are two possible ranks). Abusing notation we say that $\tau(u, v) = 1$ if $\tau(u) < \tau(v)$ (u is preferrede over v) and $\tau(u, v) = 0$ otherwise (note that here we can have $\tau(u, v) = \tau(v, u) = 0$). Our abuse of notation allows us to use the readily defined function L to measure the loss of a ranking $\sigma \in S_V$ against $\tau^* \in \Pi_V$ (which will usually take the role of a ground truth):

$$L(\sigma, \tau^*) = \binom{n}{2}^{-1} \sum_{u \neq v} \sigma(u, v) \tau^*(v, u) \ .$$

Note that this coincides with $L_\omega(\sigma, \sigma^*)$, where σ^* is any ranking on V with $\sigma^*(u) < \sigma^*(v)$ whenever $\tau^*(u) < \tau^*(v)$, and ω is as in (3) with

$$k = |\{u \in V : \tau^*(u) = 0\}| \ .$$

A note on normalization: The bipartite case is the one considered in [6], with a small different which is crucial for some of the bounds we derive. There, the loss function is defined as

$$|\{u, v : \tau^*(u) < \tau^*(v)\}|^{-1} \sum_{u \neq v} \sigma(u, v) \tau^*(v, u) \ . \tag{6}$$

If we are working with just one τ^*, the two loss functions are the same up to a constant. However, if we have a distribution over τ^* and consider the expected loss, then there may be a difference. For simplicity we will work with the definition derived from (4), and will leave the other choice for discussion in Section 4.

2.4   Independence on Irrelevant Alternatives and Regret Functions

The subset V is chosen from the universe U from some distribution. Together with V, a ground truth ranking $\sigma^* \in S(V)$, and an admissible weight function ω are also chosen randomly. We let D denote the distribution on V, σ^*, ω. (In the bipartite case, D is a distribution on V and on $\tau^* \in \Pi_v$.)

Definition 1. *A distribution D on V, σ^*, ω satisfies the pairwise independence on irrelevant alternatives (IIA) property if for all distinct $u, v \in U$, conditioned on $u, v \in V$ the random variables $\sigma^*(u, v)\omega(u, v)$ and $V \setminus \{u, v\}$ are independent.*

In the bipartite case this translates to

Definition 2. *A distribution D on V, τ^* satisfies the pairwise IIA property if for all distinct $u, v \in U$, conditioned on $u, v \in V$ the random variables $\tau^*(u, v)$ and $V \setminus \{u, v\}$ are independent.*

Note that in the bipartite case, D can satisfy pairwise IIA while not satisfying pointwise IIA. (Pointwise IIA means that conditioned on $u \in V$, $\tau^*(u)$ and $V \setminus \{u\}$ are independent.) In certain applications, e.g. when ground truth is obtained from humans, it is reasonable *not* to assume pointwise IIA. Think of the "grass is greener" phenomenon: a satisfactory option may seem unsatisfactory in the presence of an alternative. Continuing the analogue, assuming pairwise IIA means that choosing between two options does not depend on the presence of a third alternative. (By *choosing* we mean that ties are allowed.)

In this work we do not assume pointwise IIA, and when deriving loss bounds we will not assume pairwise IIA either. We will need pairwise IIA when working with *regret*, which is an adjustment of the loss designed so that an optimal solution would have a value of 0 with respect to the ground truth. As pointed out in [6], the regret measures the loss modulo "noise".

Using regret (here) makes sense when the optimal solution has a strictly positive loss value. In our case it can only happen if the ground truth is a proper distribution, namely, the probability mass is not concentrated on one point.

To define ranking regret, assume we are learning how to obtain a full ranking σ of V, using an algorithm A, so that $\sigma = A_s(V)$, where s is a random stream of bits possibly used by the algorithm. For ranking learning, we define the regret of A against D as

$$\mathrm{R}_{rank}(A,D) = E_{V,\sigma^*,\omega,s}[L_\omega(A_s(V),\sigma^*)] - \min_{\tilde{\sigma} \in S(U)} E_{V,\sigma^*,\omega}[L_\omega(\tilde{\sigma}_{|V},\sigma^*)] \ ,$$

where $\tilde{\sigma}_{|V} \in S(V)$ is defined by restricting the ranking $\tilde{\sigma} \in S(U)$ to V in a natural way.

In the preference classification setting, it makes sense to define the regret of a preference function $h : U \times U \mapsto \{0,1\}$ as follows:

$$\mathrm{R}_{class}(h,D) = E_{V,\sigma^*,\omega}[L_\omega(h_{|V},\sigma^*)] - \min_{\tilde{h}} E_{V,\sigma^*,\omega}[L_\omega(\tilde{h}_{|V},\sigma^*)] \ ,$$

where the minimum is over $\tilde{h}$ a preference function over U, and $\cdot_{|V}$ is a restriction operator on preference functions defined in the natural way. For the bipartite special case, we have the simplified form:

$$\mathrm{R}_{rank}(A,D) = E_{V,\tau^*,s}[L(A_s(V),\tau^*)] - \min_{\tilde{\sigma} \in S(U)} E_{V,\tau^*}[L(\tilde{\sigma}_{|V},\tau^*)] \tag{7}$$

$$\mathrm{R}_{class}(h,D) = E_{V,\tau^*}[L(h_{|V},\tau^*)] - \min_{\tilde{h}} E_{V,\tau^*}[L(\tilde{h}_{|V},\tau^*)] \ . \tag{8}$$

The regret measures how well an algorithm or a classifier performs compared to the best "static" algorithm, namely, one that ranks U in advance (in R_{rank}) or provides preference information on U in advanced (in R_{class}). Note that the minimizer $\tilde{h}$ in (8) can be easily found by considering each $u,v \in U$ separately. More precisely, one can take

$$\tilde{h}(u,v) = \begin{cases} 1 & E_{\tau^*}[\tau^*(u,v)|u,v \in V] > E_{\tau^*}[\tau^*(v,u)|u,v \in V] \\ 0 & E_{\tau^*}[\tau^*(u,v)|u,v \in V] < E_{\tau^*}[\tau^*(v,u)|u,v \in V] \\ \mathbf{1}_{u>v} & \text{otherwise (equality)} \end{cases} \tag{9}$$

Now notice that if D satisfies pairwise IIA, then for any set V_0 containing u, v,

$$E_{\tau^*}[\tau^*(u, v)|V = V_0] = E_{\tau^*}[\tau^*(u, v)|u, v \in V] \ .$$

Therefore, in this case the $\min_{\tilde{h}}$ and E_V operators commute:

$$\min_{\tilde{h}} E_{V, \tau^*}[L(\tilde{h}_{|V}, \tau^*)] = E_V \min_{\tilde{h}} E_{\tau^*}[L(\tilde{h}_{|V}, \tau^*)] \ .$$

For our analysis it will indeed be useful to swap the min and E_V operators. We define

$$R'_{rank}(A, D) = E_{V, \tau^*, s}[L(A_s(V), \tau^*)] - E_V \min_{\tilde{\sigma} \in S(V)} E_{\tau^*}[L(\tilde{\sigma}, \tau^*)] \qquad (10)$$

$$R'_{class}(h, D) = E_{V, \tau^*}[L(h_{|V}, \tau^*)] - E_V \min_{\tilde{h}} E_{\tau^*}[L(\tilde{h}, \tau^*)] \ , \qquad (11)$$

where now $\min_{\tilde{h}}$ is over preference functions $\tilde{h}$ on V. We summarize this section with the following:

Observation 1 *1. In general (using the concavity of* min *and Jensen's inequality):* $R'_{rank}(A, D) \geq R_{rank}(A, D)$;
  2. Assuming pairwise IIA: $R'_{class}(h, D) = R_{class}(h, D)$.

3 Algorithm for Ranking Using a Preference Function

This section describes and analyzes an algorithm for obtaining a global ranking of a subset using a prelearned preference function h, which corresponds to the second stage of the preference-based setting. Our bound on the loss will be derived using conditional expectation on the preference loss assuming a fixed subset $V \subset U$, and fixed σ^* and ω. To further simplify the analysis, we assume that h is binary, that is $h(u, v) \in \{0, 1\}$ for all $u, v \in U$.

3.1 Description

One simple idea to obtain a global ranking of the points in V consists of using a standard comparison-based sorting algorithm where the comparison operation is based on the preference function. However, since in general the preference function is not transitive, the property of the resulting permutation obtained is unclear.

This section shows however that the permutation generated by the standard QuickSort algorithm provides excellent guarantees.[5] Thus, the algorithm we suggest is the following. Pick a random *pivot* element u uniformly at random from V. For each $v \neq u$, place v on the left[6] of u if $h(v, u) = 1$, and to its right otherwise. Proceed recursively with the array to the left of u and the one to its right

---

[5] We are not assuming here transitivity as in most textbook presentations of Quick-Sort.

[6] We will use the convention that ranked items are written from left to right, starting with the most preferred ones.

and return the concatenation of the permutation returned by the left recursion, u, and the permutation returned by the right recursion.

We will denote by $Q_s^h(V)$ the permutation resulting in running QuickSort on V using preference function h, where s is the random stream of bits used by QuickSort for the selection of the pivots. As we shall see in the next two sections, on average, this algorithm produces high-quality global rankings in a time-efficient manner.

3.2 Ranking Quality Guarantees

The following theorems give bounds on the ranking quality of the algorithm described, for both loss and regret, on the general and bipartite cases.

Theorem 2 (Loss bounds in general case). *For any fixed subset $V \subseteq U$, preference function h on V, ranking $\sigma^* \in S(V)$ and admissible weight function ω the following bound holds:*

$$\mathop{\mathrm{E}}_{s}[L_\omega(Q_s^h(V), \sigma^*)] \leq 2L_\omega(h, \sigma^*) \tag{12}$$

Note: This implies by the principle of conditional expectation that

$$\mathop{\mathrm{E}}_{D,s}[L_\omega(Q_s^h(V), \sigma^*)] \leq 2E_D[L_\omega(h, \sigma^*)] \tag{13}$$

(where h can depend on V).

Theorem 3 (Loss and regret bounds in bipartite case). *For any fixed $V \subset U$, preference function h over V and $\tau^* \in \Pi(V)$, the following bound holds:*

$$\mathop{\mathrm{E}}_{s}[L(Q_s^h(V), \tau^*] = L(h, \tau^*) \ . \tag{14}$$

If V, τ^* are drawn from some distribution D satisfying pairwise IIA, then

$$\mathrm{R}_{rank}(Q_s^h(\cdot), D) \leq \mathrm{R}_{class}(h, D) \tag{15}$$

Note: Equation (14) implies by the principle of conditional expectation that if V, τ^* are drawn from a distribution D, then

$$\mathop{\mathrm{E}}_{D,s}[L(Q_s^h(V), \tau^*)] = E_D[L(h, \tau^*)] \tag{16}$$

(where h can depend on V).

To prove these theorems, we must first introduce some tools to help analyze QuickSort. These tools were first developed in [4] in the context of optimization, and here we initiate their use in learning.

3.3 Analyzing QuickSort

Assume V is fixed, and let $Q_s = Q_s^h(V)$ be the (random) ranking outputted by QuickSort on V using preference function h. During the execution of QuickSort, the order between two points $u, v \in V$ is determined in one of two ways:

- Directly: u (or v) was selected as the pivot with v (resp. u) present in the same sub-array in a recursive call to QuickSort. We denote by $p_{uv} = p_{vu}$ the probability of that event. In that case, the algorithm orders u and v according to the preference function h.
- Indirectly: a third element $w \in V$ is selected as pivot with w, u, v all present in the same sub-array in a recursive call to QuickSort, u is assigned to the left sub-array and v to the right (or vice-versa).

  Let p_{uvw} denote the probability of the event that u, v, and w be present in the same array in a recursive call to QuickSort and that one of them be selected as pivot. Note that conditioned on that event, each of these three elements is equally likely to be selected as a pivot since the pivot selection is based on a uniform distribution.

  If (say) w is selected among the three, then u will be placed on the left of v if $h(u, w) = h(w, v) = 1$, and to its right if $h(v, w) = h(w, u) = 1$. In all other cases, the order between u, v will be determined only in a deeper nested call to QuickSort.

Let $X, Y : V \times V \mapsto \mathbb{R}$ be any two functions on ordered pairs $u, v \in V$, and let $Z : \binom{V}{2} \mapsto \mathbb{R}$ be a function on unordered pairs (sets of two elements). By convention, we use $X(u, v)$ to denote ordered arguments, and Y_{uv} to denote unordered arguments. We define three functions $\alpha[X, Y] : \binom{V}{2} \mapsto \mathbb{R}$, $\beta[X] : \binom{V}{3} \mapsto \mathbb{R}$ and $\gamma[Z] : \binom{V}{3} \mapsto \mathbb{R}$ as follows:

$$\alpha[X, Y]_{uv} = X(u, v)Y(v, u) + X(v, u)Y(u, v)$$

$$\beta[X]_{uvw} = \frac{1}{3}(h(u, v)h(v, w)X(w, u) + h(w, v)h(v, u)X(u, w))$$

$$+ \frac{1}{3}(h(v, u)h(u, w)X(w, v) + h(w, u)h(u, v)X(v, w))$$

$$+ \frac{1}{3}(h(u, w)h(w, v)X(v, u) + h(v, w)h(w, u)X(u, v)) \qquad (17)$$

$$\gamma[Z]_{uvw} = \frac{1}{3}(h(u, v)h(v, w) + h(w, v)h(v, u))Z_{uw}$$

$$+ \frac{1}{3}(h(v, u)h(u, w) + h(w, u)h(u, v))Z_{vw}$$

$$+ \frac{1}{3}(h(u, w)h(w, v) + h(v, w)h(w, u))Z_{uv} .$$

Lemma 1 (QuickSort decomposition).

1. For any $Z : \binom{V}{2} \mapsto \mathbb{R}$,

$$\sum_{u<v} Z_{uv} = \sum_{u<v} p_{uv} Z_{uv} + \sum_{u<v<w} p_{uvw} \gamma[Z]_{uvw} .$$

2. For any $X : V \times V \mapsto \mathbb{R}$,

$$E_s[\sum_{u<v} \alpha[Q_s, X]_{uv}] = \sum_{u<v} p_{uv}\alpha[h, X]_{uv} + \sum_{u<v<w} p_{uvw}\beta[X]_{uvw} \ .$$

Proof. To see the first part, notice that for every unordered pair $u < v$ the expression Z_{uv} is accounted for on the RHS of the equation with total coefficient:

$$p_{uv} + \sum_{w \notin \{u,v\}} \frac{1}{3}p_{uvw}(h(u,w)h(w,v) + h(v,w)h(w,u)) \ .$$

Now, p_{uv} is the probability that the pair uv is charged directly (by definition), and $\frac{1}{3}p_{uvw}(h(u,w)h(w,v)+h(v,w)h(w,u))$ is the probability that the pair u, v is charged indirectly via w as pivot. Since each pair is charged exactly once, these probabilities are of pairwise disjoint events that cover the probability space. Hence, the total coefficient of Z_{uv} on the RHS is 1, as is on the LHS. The second part is proved similarly.

3.4 Loss Bounds

We prove the first part of Theorems 2 and 3. We start with the general case notation. The loss incurred by QuickSort is (as a function of the random bits s), for fixed σ^*, ω, clearly $L_\omega(Q_s, \sigma^*) = \binom{n}{2}^{-1} \sum_{u<v} \alpha[Q_s, \Delta]_{uv}$, where $\Delta : V \times V \mapsto \mathbb{R}$ is defined as $\Delta(u,v) = \omega(\sigma^*(u), \sigma^*(v))\sigma^*(u,v)$. By the second part of Lemma 1, the expected loss is therefore

$$\mathop{\mathrm{E}}_{s}[L_\omega(Q_s, \sigma^*)] = \binom{n}{2}^{-1} \left(\sum_{u<v} p_{uv}\alpha[h, \Delta]_{uv} + \sum_{u<v<w} p_{uvw}\beta[\Delta]_{uvw} \right) \ . \qquad (18)$$

Similarly, we have that $L_\omega(h, \sigma^*) = \binom{n}{2}^{-1} \sum_{u<v} \alpha[h, \Delta]_{uv}$. Therefore, using the first part of Lemma 1,

$$L_\omega(h, \sigma^*) = \binom{n}{2}^{-1} \left(\sum_{u<v} p_{uv}\alpha[h, \Delta]_{uv} + \sum_{u<v<w} \gamma[\alpha[h, \Delta]]_{uvw} \right) \ . \qquad (19)$$

To complete the proof for the general (non-bipartite) case, it suffices to show that for all u, v, w, $\beta[\Delta]_{uvw} \leq 2\gamma[\alpha[h, \Delta]]_{uvw}$. Up to symmetry, there are two cases to consider. The first case assumes h induces a cycle on u, v, w, and the second assumes it doesn't.

1. Without loss of generality, assume $h(u,v) = h(v,w) = h(w,u)$. Plugging in the definitions, we get

$$\beta[\Delta]_{uvw} = \frac{1}{3}(\Delta(u,v) + \Delta(v,w) + \Delta(w,u)) \qquad (20)$$

$$\gamma[\alpha[h, \Delta]]_{uvw} = \frac{1}{3}(\Delta(v,u) + \Delta(w,v) + \Delta(u,w)) \ . \qquad (21)$$

By the properties (P1)-(P3) of ω, transitivity of σ^* and definition of Δ, we easily get that Δ satisfies the triangle inequality:

$$\Delta(u, v) \leq \Delta(u, w) + \Delta(w, v)$$
$$\Delta(v, w) \leq \Delta(v, u) + \Delta(u, w)$$
$$\Delta(w, u) \leq \Delta(w, v) + \Delta(v, u)$$

Summing up the three equations, this implies that $\beta[\Delta]_{uvw} \leq 2\gamma[\alpha[h, \Delta]]_{uvw}$.

2. Without loss of generality, assume $h(u, v) = h(v, w) = h(u, w) = 1$. By plugging in the definitions, this implies that

$$\beta[\Delta]_{uvw} = \gamma[\alpha[h, \Delta]]_{uvw} = \alpha[h, \Delta]_{uw} ,$$

as required.

This concludes the proof for the general case. As for the bipartite case, (20-21) translates to

$$\beta[\Delta]_{uvw} = \frac{1}{3}(\tau^*(u, v) + \tau^*(v, w) + \tau^*(w, u)) \tag{22}$$

$$\gamma[\alpha[h, \Delta]]_{uvw} = \frac{1}{3}(\tau^*(v, u) + \tau^*(w, v) + \tau^*(u, w)) . \tag{23}$$

It is trivial to see that the two expressions are identical for any partition τ^* (indeed, they count the number of times we cross the partition from left to right when going in a circle on u, v, w: it does not matter in which direction we are going). This concludes the loss bound part of Theorems 2 and 3.

$\square$

We place Theorem 2 in the framework used by Cohen et al [8]. There, the objective is to find a ranking σ that has a low loss measured against h compared to the *theoretical* optimal ranking $\sigma_{optimal}$. Therefore, the problem considered there (modulo learning a preference function h) is a combinatorial optimization and not a learning problem. More precisely, we define

$$\sigma_{optimal} = \underset{\sigma}{\operatorname{argmin}} \, L(h, \sigma)$$

and want to minimize $L(h, \sigma)/L(h, \sigma_{optimal})$.

Corollary 1. *For any $V \subseteq U$ and preference function h over V, the following bound holds:*

$$\underset{s}{\mathrm{E}}[L(Q_s^h(V), \sigma_{optimal})] \leq 2 \, L(h, \sigma_{optimal}) . \tag{24}$$

The corollary is immediate because technically any ranking and in particular $\sigma_{optimal}$ can be taken as σ^* in the proof of Theorem 2.

Corollary 2. *Let $V \subseteq U$ be an arbitrary subset of U and let $\sigma_{optimal}$ be as above. Then, the following bound holds for the pairwise disagreement of the ranking $Q_s^h(V)$ with respect to h:*

$$\underset{s}{\mathrm{E}}[L(h, Q_s^h(V))] \leq 3 \, L(h, \sigma_{optimal}). \tag{25}$$

Proof. This result follows directly Corollary 1 and the application of a triangle inequality. □

The result in Corollary 2 is known from previous work [4, 3], where it is proven directly without resorting to the intermediate inequality (24). In fact, a better bound of 2.5 is known to be achievable using a more complicated algorithm, which gives hope for a 1.5 bound improving Theorem 2.

3.5 Regret Bounds for Bipartite case

We prove the second part (regret bounds) of Theorem 3. By Observation 1, it is enough to prove that $\mathrm{R}'_{rank}(A, D) \leq \mathrm{R}'_{class}(h, D)$. Since in the definition of R'_{rank} and R'_{class} the expectation over V is outside the min operator, we may continue fixing V. Let D_V denote the distribution over τ^* conditioned on V It is now clearly enough to prove

$$\mathop{\mathrm{E}}_{D_V, s}[L(Q_s^h, \tau^*)] - \min_{\tilde{\sigma}} \mathop{\mathrm{E}}_{D_V}[L(\tilde{\sigma}, \tau^*)] \leq \mathop{\mathrm{E}}_{D_V}[L(h, \tau^*)] - \min_{\tilde{h}} \mathop{\mathrm{E}}_{D_V}[L(\tilde{h}, \tau^*)] \quad (26)$$

We let $\mu(u, v) = \mathrm{E}_{D_V}[\tau^*(u, v)]$. (By pairwise IIA, $\mu(u, v)$ is the same for all V such that $u, v \in V$.) By linearity of expectation, it suffices to show that

$$\mathop{\mathrm{E}}_{s}[L(Q_s^h, \mu)] - \min_{\tilde{\sigma}} L(\tilde{\sigma}, \mu) \leq L(h, \mu) - \min_{\tilde{h}} L(\tilde{h}, \mu) . \quad (27)$$

Now let $\tilde{\sigma}$ and $\tilde{h}$ be the minimizers of the min operators on the left and right sides, respectively. Recall that for all $u, v \in V$, $\tilde{h}(u, v)$ can be taken greedily as a function of $\mu(u, v)$ and $\mu(v, u)$ (as in (9)).

$$\tilde{h}(u, v) = \begin{cases} 1 & \mu(u, v) > \mu(v, u) \\ 0 & \mu(u, v) < \mu(v, u) \\ \mathbf{1}_{u>v} & \text{otherwise (equality) .} \end{cases} \quad (28)$$

Using Lemma 1 and linearity, we write the LHS of (27) as:

$$\binom{n}{2}^{-1} \left(\sum_{u<v} p_{uv} \alpha[h - \tilde{\sigma}, \mu]_{uv} + \sum_{u<v<w} p_{uvw}(\beta[\mu] - \gamma[\alpha[\tilde{\sigma}, \mu]])_{uvw} \right)$$

and the RHS of (27) as:

$$\binom{n}{2}^{-1} \left(\sum_{u<v} p_{uv} \alpha[h - \tilde{h}, \mu]_{uv} + \sum_{u<v<w} p_{uvw} \gamma[\alpha[h - \tilde{h}, \mu]]_{uvw} \right) .$$

Now, clearly for all u, v by construction of $\tilde{h}$ we must have $\alpha[h - \tilde{\sigma}, \mu]_{uv} \leq \alpha[h - \tilde{h}, \mu]_{uv}$. To conclude the proof of the theorem, we define $F : \binom{n}{3} \mapsto \mathbb{R}$ as follows:

$$F = \beta[\mu] - \gamma[\alpha[\tilde{\sigma}, \mu]] - (\gamma[\alpha[h, \mu]] - \gamma[\alpha[\tilde{h}, \mu]]) \ . \tag{29}$$

It now suffices to prove that $F_{uvw} \leq 0$ for all $u, v, w \in V$. Clearly F is a function of the values of

$$\begin{aligned}
\mu(a, b) &: a, b \in \{u, v, w\} \\
h(a, b) &: a, b \in \{u, v, w\} \\
\tilde{\sigma}(a, b) &: a, b \in \{u, v, w\}
\end{aligned} \tag{30}$$

(recall that $\tilde{h}$ depends on μ.) The μ-variables can take values satisfying following constraints or all $u, v, w \in V$:

$$\mu(a, c) \leq \mu(a, b) + \mu(b, c) \quad \forall \ \{a, b, c\} = \{u, v, w\} \tag{31}$$

$$\mu(u, v) + \mu(v, w) + \mu(w, u) = \mu(v, u) + \mu(w, v) + \mu(u, w) \tag{32}$$

$$\mu(a, b) \geq 0 \quad \forall \ a, b \in \{u, v, w\} \ . \tag{33}$$

(the second constraint is obvious for any partition τ^*.)

Let $P \subseteq \mathbb{R}^6$ denote the polytope defined by (31-33) in the variables $\mu(a, b)$ for $\{a, b\} \subseteq \{u, v, w\}$. We subdivide P into smaller subpolytopes on which the $\tilde{h}$ variables are constant. Up to symmetries, we can consider only two cases: (i) $\tilde{h}$ induces a cycle on u, v, w and (ii) $\tilde{h}$ is cycle free on u, v, w.

(i) Without loss of generality, assume $\tilde{h}(u, v) = \tilde{h}(v, w) = \tilde{h}(w, u) = 1$. But this implies that $\mu(u, v) \geq \mu(v, u)$, $\mu(v, w) \geq \mu(w, v)$ and $\mu(w, u) \geq \mu(u, w)$. Together with (32) and (33) this implies that $\mu(u, v) = \mu(v, u)$, $\mu(v, w) = \mu(w, v)$ and $\mu(w, u) = \mu(u, w)$. Consequently

$$\beta[\mu]_{uvw} = \gamma[\alpha[\tilde{\sigma}, \mu]]_{uvw} = \gamma[\alpha[h, \mu]]_{uvw} = \gamma[\alpha[\tilde{h}, \mu]]_{uvw}$$

$$= \frac{1}{3}(\mu(u, v) + \mu(v, w) + \mu(w, u))$$

and $F_{uvw} = 0$, as required.

(ii) Without loss of generality, assume $\tilde{h}(u, v) = \tilde{h}(v, w) = \tilde{h}(u, w) = 1$. This implies that

$$\begin{aligned}
\mu(u, v) &\geq \mu(v, u) \\
\mu(v, w) &\geq \mu(w, v) \\
\mu(u, w) &\geq \mu(w, u) \ .
\end{aligned} \tag{34}$$

Let $\tilde{P} \subseteq P$ denote the polytope defined by (34) and (31)-(33). Clearly F is linear in the 6 μ variables when all the other variables are fixed. Since F is also homogenous in the μ variables, it is enough to prove that $F \leq 0$ for μ taking values in $\tilde{P}' \subseteq \tilde{P}$, which is defined by adding the constraint, say,

$$\sum_{a, b \in \{u, v, w\}} \mu(a, b) = 2 \ .$$

It is now enough to prove that $F \leq 0$ for τ^* being a vertex of of $\tilde{P}'$. This finite set of cases can be easily checked to be:

$$(\mu(u,v), \mu(v,u), \mu(u,w), \mu(w,u), \mu(w,v), \mu(v,w)) \in A \cup B$$

where $A = \{(0,0,1,0,0,1), (1,0,1,0,0,0)\}$
$B = \{(.5,.5,.5,.5,0,0), (.5,.5,0,0,.5,.5), (0,0,.5,.5,.5,.5)\}$.

The points in B were already checked in case (i) (which is, geometrically, a boundary of case (ii)). It remains to check the two points in A.
 – case $(0,0,1,0,0,1)$: Plugging in the definitions, one checks that:

$$\beta[\mu]_{uvw} = \frac{1}{3}(h(w,v)h(v,u) + h(w,u)h(u,v))$$

$$\gamma[\alpha[h,\mu]]_{uvw} = \frac{1}{3}((h(u,v)h(v,w) + h(w,v)h(v,u))h(w,u)$$
$$+ (h(v,u)h(u,w) + h(w,u)h(u,v))h(w,v))$$

$$\gamma[\alpha[\tilde{h},\mu]]_{uvw} = 0 \ .$$

Clearly F could be positive only of $\beta_{uvw} = 1$, which happens if and only if either $h(w,v)h(v,u) = 1$ or $h(w,u)h(u,v) = 1$. In the former case we get that either $h(w,v)h(v,u)h(w,u) = 1$ or $h(v,u)h(u,w)h(w,v) = 1$, both implying $\gamma[\alpha[h,\mu]]_{uvw} \geq 1$, hence $F \leq 0$. In the latter case either $h(w,u)h(u,v)h(w,v) = 1$ or $h(u,v)h(v,w)h(w,u) = 1$, both implying again $\gamma[\alpha[h,\mu]]_{uvw} \geq 1$ and hence $F \leq 0$.
 – case $(1,0,1,0,0,0)$:Plugging in the definitions, one checks that:

$$\beta[\mu]_{uvw} = \frac{1}{3}(h(w,v)h(v,u) + h(v,w)h(w,u))$$

$$\gamma[\alpha[h,\mu]]_{uvw} = \frac{1}{3}((h(u,v)h(v,w) + h(w,v)h(v,u))h(w,u)$$
$$+ (h(u,w)h(w,v) + h(v,w)h(w,u))h(v,u)) \ .$$

$$\gamma[\alpha[\tilde{h},\mu]]_{uvw} = 0 \ .$$

Now F could be positive if and only if either $h(w,v)h(v,u) = 1$ or $h(v,w)h(w,u) = 1$. In the former case we get that either $h(w,v)h(v,u)h(w,u) = 1$ or $h(v,u)h(u,w)h(w,v) = 1$, both implying $\gamma[\alpha[h,\mu]]_{uvw} \geq 1$, hence $F \leq 0$. In the latter case either $h(v,w)h(w,u)h(v,u) = 1$ or $h(u,v)h(v,w)h(w,u) = 1$, both implying again $\gamma[\alpha[h,\mu]]_{uvw} \geq 1$ and hence $F \leq 0$.

This concludes the proof for the bipartite case. $\qquad\qquad\qquad\qquad\square$

3.6 Time Complexity

Running QuickSort does not entail $\Omega(|V|^2)$ accesses to $h_{u,v}$. The following bound on the running time is proven in Section 3.6.

Theorem 4. *The expected number of times QuickSort accesses to the preference function h is at most $O(n \log n)$. Moreover, if only the top k elements are sought then the bound is reduced to $O(k \log k + n)$ by pruning the recursion.*

It is well known that QuickSort on cycle free tournaments runs in time $O(n \log n)$, where n is the size of the set we want to sort. That it is true for QuickSort on general tournaments is a simple extension (communicated by Heikki Mannila) which we present it here for self containment. The second part requires more work.

Proof. Let $T(n)$ be the maximum expected running time of QuickSort on a possibly cyclic tournament on n vertices in terms of number of comparisons. Let $G = (V, A)$ denote a tournament. The main observation is that each vertex $v \in V$ is assigned to the left recursion with probability exactly $\text{outdeg}(v)/n$ and to the right with probability $\text{indeg}(v)/n$, over the choice of the pivot. Therefore, the expected size of both the left and right recursions is exactly $(n - 1)/2$. The separation itself costs $n - 1$ comparisons. The resulting recursion formula $T(n) \leq n - 1 + 2T((n - 1)/2)$ clearly solves to $T(n) = O(n \log n)$.

Assume now that only the k first elements of the output are sought, that is, we are interested in outputting only elements in positions $1, \ldots, k$. The algorithm which we denote by k-QuickSort is clear: recurse with $\min\{k, n_L\}$-QuickSort on the left side and $\max\{0, k - n_L - 1\}$-QuickSort on the right side, where n_L, n_R are the sizes of the left and right recursions respectively and 0-QuickSort takes 0 steps by assumption. To make the analysis simpler, we will assume that whenever $k \geq n/8$, k-QuickSort simply returns the output of the standard QuickSort, which runs in expected time $O(n \log n) = O(n + k \log k)$, within the sought bound. Fix a tournament G on n vertices, and let $t_k(G)$ denote the running time of k-QuickSort on G, where $k < n/8$. Denote the (random) left and right subtournaments by G_L and G_R respectively, and let $n_L = |G_L|, n_R = |G_R|$ denote their sizes in terms of number of vertices. Then, clearly,

$$t_k(G) = n - 1 + t_{\min\{k, n_L\}}(G_L) + t_{\max\{0, k - n_L - 1\}}(G_R) . \tag{35}$$

Assume by structural induction that for all $\{k', n' : k' \leq n' < n\}$ and for all tournaments G' on n' vertices, $\mathrm{E}[t_{k'}(G')] \leq cn' + c'k' \log k'$ for some global $c, c' > 0$. Then, by conditioning on G_L, G_R, taking expectations on both sides of (35) and by induction,

$$\mathrm{E}[t_k(G) \mid G_L, G_R] \leq$$
$$n - 1 + cn_L + c' \min\{k, n_L\} \log \min\{k, n_L\} +$$
$$cn_R \mathbf{1}_{n_L < k - 1} + c' \max\{k - n_L - 1, 0\} \log \max\{k - n_L - 1, 0\} .$$

By convexity of the function $x \mapsto x \log x$,

$$\min\{k, n_L\} \log \min\{k, n_L\} + \max\{k - n_L - 1, 0\} \log \max\{k - n_L - 1, 0\} \leq k \log k , \tag{36}$$

hence

$$\mathrm{E}[t_k(G) \mid G_L, G_R] \leq n - 1 + cn_L + cn_R \mathbf{1}_{n_L < k - 1} + c'k \log k. \tag{37}$$

By conditional expectation,

$$\mathrm{E}[t_k(G)] \le n - 1 + c(n-1)/2 + c'k\log k + c\,\mathrm{E}[n_R\mathbf{1}_{n_L<k-1}].$$

To complete the inductive hypothesis, we need to bound $\mathrm{E}[n_R\mathbf{1}_{n_L<k-1}]$ which is bounded by $n\Pr[n_L < k-1]$. The event $\{n_L < k-1\}$, equivalent to $\{n_R > n-k\}$, occurs when a vertex of out-degree at least $n - k \ge 7n/8$ is chosen as pivot. For a random pivot $v \in V$, where V is the vertex set of G, $\mathrm{E}[\mathrm{outdeg}(v)^2] \le n^2/3 + n/2 \le n^2/2.9$. Indeed, each pair of edges $(v, u_1) \in A$ and $(v, u_2) \in A$ for $u_1 \ne u_2$ gives rise to a triangle which is counted exactly twice in the cross-terms, hence $n^2/3$ which upper-bounds $2\binom{n}{3}/n$; $n/2$ bounds the diagonal). Thus, $\Pr[\mathrm{outdeg}(v) \ge 7n/8] = \Pr[\mathrm{outdeg}(v)^2 \ge 49n^2/64] \le 0.46$ (by Markov). Plugging in this value into our last estimate yields

$$\mathrm{E}[t_k(G)] \le n - 1 + c(n-1)/2 + c'k\log k + 0.46 \times cn,$$

which is at most $cn + c'k\log k$ for $c \ge 30$, as required. $\qquad\square$

4   Discussion

4.1   History of QuickSort

The now standard textbook algorithm was discovered by Hoare [16] in 1961. Montague and Aslam [20] experiment with QuickSort for information retrieval by aggregating rankings from different sources of retrieval. They claim an $O(n\log n)$ time bound on the number of comparisons although the proof seems to rely on the folklore QuickSort proof without addressing the non-transitivity problem. They prove certain combinatorial bounds on the output of QuickSort and provide empirical justification to its IR merits. Ailon, Charikar and Newman [4] also consider the rank aggregation problem and prove theoretical cost bounds for many ranking problems on weighted tournaments. They strengthen these bounds by considering nondeterministic pivoting rules (arising from solutions to certain ranking LP's). This work was extended by Ailon [3] to deal with rankings with ties (in particular, top-k rankings). Hedge et al [15] and Williamson et al [22] derandomize the random pivot selection step in QuickSort for many of the combinatorial optimization problems studied by Ailon et al.

4.2   The decomposition technique

The technique developed in Lemma 1 is very general and can used for a wide variety of loss functions and variants of QuickSort involving nondeterministic ordering rules (see [4, 3]). Such results would typically amount to bounding $\beta[X]_{uvw}/\gamma[Z]_{uvw}$ for some carefully chosen functions X, Z (depending on the application).

4.3 Combinatorial Optimization vs. Learning

In Ailon et al's work [4, 3] the QuickSort algorithm (sometimes referred to there as FAS-Pivot) is used to approximate certain NP-Hard (see [5]) weighted instances of minimum feedback arcset in tournaments. There is much similarity between the techniques used in the analyses, but there is also a significant difference that should be noted. In the minimum feedback arc-set problem we are given a tournament G and wish to find an acyclic tournament H on the same vertex set minimizing $\Delta(G, H)$, where Δ counts the number of edges pointing in opposite directions between G, H (or a weighted version thereof). However, the cost we are considering is $\Delta(G, H_\sigma)$ for some fixed acyclic tournament H_σ induced by some permutation σ (the ground truth). In this work we showed in fact that if G' is obtained from G using QuickSort, then $E[\Delta(G', H_\sigma)] \leq 2\Delta(G, H_\sigma)$ for *any* σ (from Theorem 2). If H is the optimal solution to the (weighted) minimum feedback arc-set problem corresponding to G, then it is easy to see that $\Delta(H, H_\sigma) \leq \Delta(G, H) + \Delta(G, H_\sigma) \leq 2\Delta(G, H_\sigma)$. However, recovering G is NP-Hard in general. Approximating $\Delta(G, H)$ (as done in the combinatorial optimization world) by some constant factor[7] $1 + \varepsilon$ by an acyclic tournament H' only guarantees (using trivial arguments) a constant factor of $2 + \varepsilon$ as follows:

$$\Delta(H', H_\sigma) \leq \Delta(G, H') + \Delta(G, H_\sigma) \leq (1+\varepsilon)\Delta(G, H) + \Delta(G, H_\sigma) \leq (2+\varepsilon)\Delta(G, H_\sigma) \,.$$

This work therefore adds an important contribution to [4, 3, 18].

4.4 Normalization

As mentioned earlier, the loss function L used in the bipartite case is not exactly the same one used by Balcan et al in [6]. There the total number of "misordered pairs" is divided not by $\binom{n}{2}$ but rather by the number of mixed pairs u, v such that $\tau^*(u) \neq \tau^*(v)$ (see (6)). We will not discuss the merits of each choice in this work, but will show that the loss bound (first part) of Theorem 3 applies to their normalization as well. Indeed, let $\nu : \Pi(V) \rightarrow \mathbb{R}^+$ be any normalization function that depends on a partition, and define a loss

$$L(X, \tau^*) = \nu(\tau^*)^{-1} \sum_{u \neq v} X(u, v)\tau^*(v, u)$$

for any $X : V \times V \rightarrow \mathbb{R}^+$ (X can be a preference function h or a ranking). In [6], for example, $\nu(\tau^*)$ is taken as $|\{u, v : \tau^*(u) < \tau^*(v)\}|$ and here as $\binom{n}{2}$. Since V, τ^* are fixed in the loss bound of Theorem 3, this makes no difference for the proof. For the regret bound (second part) of Theorem 3 this however does not work. Indeed, the pairwise IIA is not enough to ensure that the event $u, v \in V$ determines $\nu(\tau^*)$, and we cannot simply swap E_D and $\min_{\bar{h}}$ as we did in Observation 1. Working around this problem seems to require a stronger version of IIA which does not seem natural.

---

[7] Kenyon-Mathiew and Schudy [18] recently found such a PTAS for the combinatorial optimization problem.

5 Lower Bounds

Balcan et al [6] prove a lower bound of a constant factor of 2 for the regret bound of the algorithm MFAT, defined as the solution to the minimum feedback arc-set problem on the tournament V with an edge (u, v) if $h(u, v) = 1$. More precisely, they show an example of fixed V, h and $\tau^* \in \Pi(V)$ such that the classification regret of h tends to $1/2$ of the ranking regret of MFAT on V, h. Note that in this case, since τ^* is fixed, the regret and loss are the same thing for both classification and ranking. Here we show the following stronger statement which is simpler to prove and applies in particular to the specific algorithm MFAT that is argued there.

Theorem 5. *For any deterministic algorithm A taking input $V \subseteq U$ and preference function h on V and outputting a ranking $\sigma \in S(V)$ there exists a distribution D on V, τ^* such that*

$$\mathrm{R}_{rank}(A, D) \geq 2 \, \mathrm{R}_{class}(h, D) \tag{38}$$

Note that this theorem says that in some sense, no deterministic algorithm that converts a preference function into a linear ranking can do better than a randomized algorithm (on expectation) in the bipartite case. Hence, randomization is essentially necessary in this scenario.

The proof is by an adversarial argument. In our construction, D will always put all the mass on a single V, τ^* (deterministic input), so the loss and regret are the same thing, and a similar argument will follow for the loss. Also note that the normalization ν will have no effect on the result.

Proof. Fix $V = \{u, v, w\}$, and D puts all the weight on this particular V and one partition τ^* (which we adversarially choose below). Assume $h(u, v) = h(v, w) = h(w, u) = 1$ (a cycle). Up to symmetry, there are two options for the output σ of A on V, h.

1. $\sigma(u) < \sigma(v) < \sigma(w)$. In this case, the adversary chooses $\tau^*(w) = 0$ and $\tau^*(u, v) = 1$. Clearly $\mathrm{R}_{class}(h, D)$ now equals $1/3$ (h pays only for misordering v, w) but $\mathrm{R}_{rank}(A, D) = 2/3$ (σ pays for misordering the pairs u, w and v, w).
2. $\sigma(w) < \sigma(v) < \sigma(u)$. In this case, the adversary chooses $\tau^*(u) = 0$ and $\tau^*(v, w) = 1$. Clearly $\mathrm{R}_{class}(h, D)$ now equals $1/3$ (h pays only for misordering u, w) but $\mathrm{R}_{rank}(A, D) = 2/3$ (σ pays for misordering the pairs u, v and u, w).

This concludes the proof.

6 Conclusion

We described a reduction of the learning problem of ranking to classification. The efficiency of this reduction makes it practical for large-scale information

extraction and search engine applications. A finer analysis of QuickSort is likely to further improve our reduction bound by providing a concentration inequality for the algorithm's deviation from its expected behavior using the confidence scores output by the classifier. Our reduction leads to a competitive ranking algorithm that can be viewed as an alternative to the algorithms previously designed for the score-based setting.

7 Acknowledgements

References

1. Shivani Agarwal, Thore Graepel, Ralf Herbrich, Sariel Har-Peled, and Dan Roth. Generalization bounds for the area under the roc curve. *Journal of Machine Learning Research*, 6:393–425, 2005.
2. Shivani Agarwal and Partha Niyogi. Stability and generalization of bipartite ranking algorithms. In *COLT*, pages 32–47, 2005.
3. Nir Ailon. Aggregation of partial rankings, p-ratings and top-m lists. In *SODA*, 2007.
4. Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 684–693. ACM, 2005.
5. Noga Alon. Ranking tournaments. *SIAM J. Discrete Math.*, 20(1):137–142, 2006.
6. Maria-Florina Balcan, Nikhil Bansal, Alina Beygelzimer, Don Coppersmith, John Langford, and Gregory B. Sorkin. Robust reductions from ranking to classification. In Nader H. Bshouty and Claudio Gentile, editors, *COLT*, volume 4539 of *Lecture Notes in Computer Science*, pages 604–619. Springer, 2007.
7. Alina Beygelzimer, Varsha Dani, Tom Hayes, John Langford, and Bianca Zadrozny. Error limiting reductions between classification tasks. In Luc De Raedt and Stefan Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, pages 49–56. ACM, 2005.
8. William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. Artif. Intell. Res. (JAIR)*, 10:243–270, 1999.
9. D. Coppersmith, Lisa Fleischer, and Atri Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournamnets. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.
10. Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. An Alternative Ranking Problem for Search Engines. In *Proceedings of the 6th Workshop on Experimental Algorithms (WEA 2007)*, volume 4525 of *Lecture Notes in Computer Science*, pages 1–21, Rome, Italy, June 2007. Springer-Verlag, Heidelberg, Germany.
11. Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. Magnitude-Preserving Ranking Algorithms. In *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML 2007)*, Oregon State University, Corvallis, OR, June 2007.

12. Koby Crammer and Yoram Singer. Pranking with ranking. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 641–647. MIT Press, 2001.

13. Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

14. J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 1982.

15. Rajneesh Hedge, Kamal Jain, David P. Williamson, and Anke van Zuylen. "deterministic pivoting algorithms for constrained ranking and clustering problems". In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

16. C.A.R. Hoare. Quicksort: Algorithm 64. *Comm. ACM*, 4(7):321–322, 1961.

17. Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

18. Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 95–103, New York, NY, USA, 2007. ACM Press.

19. Erich L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, San Francisco, California, 1975.

20. Mark H. Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 4-9, 2002*, pages 538–548. ACM, 2002.

21. Cynthia Rudin, Corinna Cortes, Mehryar Mohri, and Robert E. Schapire. Margin-based ranking meets boosting in the middle. In Peter Auer and Ron Meir, editors, *Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings*, pages 63–78. Springer, 2005.

22. David P. Williamson and Anke van Zuylen. "deterministic algorithms for rank aggregation and other ranking and clustering problems". In *Proceedings of the 5th Workshop on Approximation and Online Algorithms (WAOA) (to appear)*, 2007.