Improved Bounds for Online Learning Over the Permutahedron and Other Ranking Polytopes

Nir Ailon
Department of Computer Science, Technion IIT, Haifa, Israel
nailon@cs.technion.ac.il

Abstract

Consider the following game: There is a fixed set V of n items. At each step an adversary chooses a score function $s_t : V \mapsto [0,1]$, a learner outputs a ranking of V, and then s_t is revealed. The learner's loss is the sum over $v \in V$, of $s_t(v)$ times v's position (0th, 1st, 2nd, ...) in the ranking. This problem captures, for example, online systems that iteratively present ranked lists of items to users, who then respond by choosing one (or more) sought items. The loss measures the users' burden, which increases the further the sought items are from the top. It also captures a version of online rank aggregation.

We present an algorithm of expected regret $O(n\sqrt{OPT} + n^2)$, where OPT is the loss of the best (single) ranking in hindsight. This improves the previously best known algorithm of Suehiro et. al (2012) by saving a factor of $\Omega(\sqrt{\log n})$. We also reduce the per-step running time from $O(n^2)$ to $O(n \log n)$. We provide matching lower bounds.

1 Introduction

An online ranking system outputs at each step a complete ranking of some ground set V of n elements, observes some feedback and suffers a correpsonding loss. In this work we study a particular case of this game in which the feedback is a nonegative score function over V, and the loss is the sum, over the elements of V, of their score times their position (0th, 1st, 2nd...) in the ranking. Equivalently, this is an online linear optimization game over a polytope called the *permutahedron*.

Appearing in Proceedings of the 17^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33.

The goal of the system is to minimize its total loss after T steps. (For simplicity assume T is known in this work.) The total loss of the system is (additively) compared against that of the best (in hindsight) single ranking played throughout. The difference is called regret.

1.1 Motivation

We present several motivating examples that are addressed by our setting.

Online ranking for discrete choice. Many interactive online information systems (search, recommendation) present to a stream of users rankings of a set items in response to a specific query. As feedback, these systems often observe a the user's choice (expressed by clicking or tapping) on one (or more) of these items. Such systems are considered to be good if users choose items that are closer to the top of the retrieved ranked list, because it means they suffered less burden when seeking their information needs (making the simplifying assumption that a typical user scans the list from top to bottom).[1] We call this game *online ranking for discrete choice*.

We model this as the following online ranking game. The set of items is V. At each step t, nature chooses (and hides) a subset U_t of V, which models the next user's needs. We denote this set by its indicator (score) function $s_t : V \mapsto \{0,1\}$. The system outputs a ranking π_t of the set using some (randomized) algorithm, and then s_t is revealed to it. The system (and user) suffer a loss which is low if the elements of U_t are close to the top of π_t, and high if they are close to the bottom. More precisely, the loss function penalizes the algorithm by the sum, over the elements of U_t, of their position in π_t.

Online rank aggregation. Here, both nature and the algorithm select a ranking of V at each step, and the loss is a measure of distance, or discordance between the rankings.

[1] Note that the exact order a typical user scans the list is immaterial, as long as all users are assumed to scan the list in the same order.

This may arise in online settings in which users specify a complete ranking of a set of alternatives presented to them in some order. This problem has been extensively studied in the offline setting (see e.g. Ailon et al. (2008) and references therein). As an online game, Yasutake et al. (2012) considered the case in which the distance between rankings is the Kendall-τ metric, defined as the number of pairwise inversions. (Note that the corresponding offline problem is NP-Hard to solve exactly, see Dwork et al. (2001).) If, instead we take the *complement* of the well known Spearman correlation measure (see Section 2 for an exact definition) then the offline problem becomes polynomial time solvable (via matching). Our setting allows to address the online version via a simple transformation.

1.2 Main Results and Contribution

We define the general online game exactly in Section 2. We design in Section 3 an extremely efficient algorithm and derive bounds on its maximal expected regret. We express the bound as a function of n and a bound $\tilde{L}$ on the optimal loss in hindsight. (Standard tricks can be used to deal with the case in which no nontrivial bound is known. We omit the details from this extended abstract.)

Our main result is given in Theorem 3.1 below. Essentially, we show an expected regret bound of $O\left(n\sqrt{\tilde{L}}+n^2\right)$. We argue in Theorem 3.2 that this bound is tight, even in the extreme case in which s_t is an indicator function of a singleton.

The proofs of these theorems are given in Sections 5 and 6. In Section 4 we compare our results to previous approaches. To the best of our knowledge, the previously best known bound is that of Suehiro et al. (2012), who devise an algorithm with a bound of

$$O\left(n\sqrt{\tilde{L}\log n}+n^2\log n\right) \tag{1.1}$$

on the expected regret. We hence reduce their bound by a factor of $\Omega(\sqrt{\log n})$.[2] Additionally, our algorithms can be executed in running time $O(n\log n)$ per iteration, while theirs require time $O(n^2)$. It should be noted that Suehiro et al. (2012) also provide an alternative algorithm with a regret bound of $O(n^2\sqrt{T})$ for time horizon T, which is better than (1.1) under the assumption that $\tilde{L}=\Omega(n^2T/\log n)$. Note that this assumption is extreme in the sense that always trivially $\tilde{L}$ can be taken as $\Theta(n^2T)$ (in which case, incidentally, their result matches ours).

Finally, we discuss a more general class of loss functions which assign arbitrary nonnegative, monotonically nondecreasing importance weights to the various positions in the output ranking (instead of 0,1,2,...). The definition, result and proof sketch of this generalization are presented in Section 7 to avoid confusion at this early point.

1.3 Main Techniques

Our algorithm maintains a weight vector $w\in\mathbb{R}^V$ which, after step t completes, is proportional to $\sum_{t'=1}^{t}s_{t'}$. In the next round, the algorithm will use this weight vector as input to a noisy sorting procedure.[3] The main result in this work is, that as long as the noisy sorting procedure's output satisfies a certain property related to its behaviour on a fixed pair of elements (see Lemma 5.1), the algorithm has the desired regret bounds. We show that two noisy sorting procedures, one a version of QuickSort and the other based on a statistical model for rank data by Plackett and Luce, satisfy this property. (We refer the reader to the book Marden (1995) for more details about the Plackett-Luce model in statistics.)

2 Definitions and Problem Statement

Let V be a ground set of n items. A ranking π over V is an injection $\pi:V\mapsto[n]$, where $[n]$ denotes $\{1,2,\ldots,n\}$. We let $S(V)$ denote the space of rankings over V. The expression $\pi(v)$ for $v\in V$ is the *position* of v in the ranking, where we think of lower positions as *more favorable*. For distinct $u,v\in V$, we say that $u\prec_\pi v$ if $\pi(u)<\pi(v)$ (in words: u *beats* v). We use $[u,v]_\pi$ as shorthand for the indicator function of the predicate $u\prec_\pi v$.

At each step $t=1,\ldots,T$ the algorithm outputs a ranking π_t over V and then observes a function $s_t:V\mapsto[0,1]$. The instantaneous loss incurred by the algorithm is then

$$\ell(\pi_t,s_t)=\pi_t\cdot s_t:=\sum_{u\in V}\pi_t(u)s_t(u)\;, \tag{2.1}$$

namely, the dot product of the π_t and s_t, both viewed as vectors in $\mathbb{R}^n\equiv\mathbb{R}^V$. In this work we are interested in bounding the expected *additive* regret by an expression of the form

$$f_1(n)\sqrt{\tilde{L}}+f_2(n)\;,$$

where $\tilde{L}$ is an upper bound on $\sum_{t=1}^{T}\ell(\pi^*,s_t)$, for π^* some optimal single ranking in hindsight: $\pi^*\in\operatorname{argmin}_{\pi\in S(V)}\sum_{t=1}^{T}\ell(\pi,s_t)$. Therefore, we can replace the loss ℓ with any nonnegative loss that is upper bounded by ℓ and differs from ℓ by a constant that may depend on s_t (but not on π_t). Taking advantage of this fact, we will use the following *pairwise* loss function, $\ell\ell$, defined as follows:

$$\ell\ell(\pi_t,s_t):=\sum_{u\neq v}[u,v]_{\pi_t}[s_t(v)-s_t(u)]_+\;, \tag{2.2}$$

[2] It turns out that the factor $\sqrt{\log n}$ in the proof of Suehiro et al. (2012) can be removed with a more careful analysis of their algorithm. This has been pointed out by an anonymous reviewer, and will be explained in detailed in the full version of this work.

[3] By this we mean, a procedure that outputs a randomized ranking of an input set.

where $[x]_+$ is x if $x \geq 0$ and 0 otherwise. In words, this will introduce a loss component of $s_t(v) - s_t(u)$ whenever the pair u, v is misordered in the sense that $u \prec_{\pi_t} v$. To see why for any $s : V \mapsto [0,1]$ and $\pi \in S(V)$ the losses $\ell(\pi, s)$ and $\ell\ell(\pi, s)$ differ by a number that depends on s only, one trivially verifies that when moving from π to a ranking π' obtained from π by swapping two *consecutive* elements, the two differences $\ell(\pi', s) - \ell(\pi, s)$ and $\ell\ell(\pi', s_t) - \ell\ell(\pi, s_t)$ are equal. To see why $\ell\ell \leq \ell$, notice that $\ell\ell(\pi, s) = 0$ if π orders V in decreasing s-value. Slightly abusing notation, we define

$$\ell\ell(\pi, s, u, v) := [u, v]_\pi [s(v) - s(u)]_+ + [v, u]_\pi [s(u) - s(v)]_+ ,$$

so that $\ell\ell(\pi_t, s_t)$ takes the form $\sum_{\{u,v\} \subseteq V} \ell\ell(\pi, s, u, v)$.[4]

Over a horizon of T steps, the algorithm's total loss is now defined as $L_T(\mathrm{Alg}) := \sum_{t=1}^T \ell\ell(\pi_t, s_t)$, and $L_T(\pi^*)$ is now defined as $\sum_{t=1}^T \ell\ell(\pi^*, s_t)$.[5]

We will say that we are in the k-choice setting if, for all t, the function s_t is an indicator function of a set of size at most k. The 1-choice (or, *single choice*) case is extremely well motivated in applications such as online search and recommendation systems in which users choose at most a single result. We will say that we are in the *Spearman rank aggregation* case if for all t,

$$s_t = 1 - \sigma_t / n \tag{2.3}$$

for some $\sigma_t \in S(V)$. The functional $\ell(\pi_t, s_t)$ is then an affine transformation of the Spearman correlation $\pi_t \cdot \sigma_t = \sum_{v \in V} \pi_t(v) \sigma_t(v)$. (Note that we take σ_t with a minus sign in 2.3 so that we can consistently use the terminology of *loss minimization* when thinking of aggregating the rankings $\sigma_1, \ldots, \sigma_T$.)

3 The Algorithm and its Guarantee

Our algorithm OnlineRank (Algorithm 1) takes as input the ground set V, a learning rate parameter $\eta \in [0,1]$, a reference to a randomized sorting procedure SortProc and a time horizon T. We present two possible randomized sorting procedures, QuickSort (Algorithm 2) and PlackettLuce (Algorithm 3). Both options satisfy an important property, described below in Lemma 5.1.

Very much like standard online gradient descent algorithms, OnlineRank maintains a weight vector over V at each step t, which is proportional to the total cost gradient $\sum_{t'=1}^t s_t$. Unlike previous approaches, the algorithm (by using QuickSort or PlackettLuce) takes advantage of the structure of permutations in drawing the action in the next step. Our main lower and upper bounds are as follows:

[4]Note that this expression makes sense because $\ell\ell(\pi, s, \cdot, \cdot)$ is symmetric in its last two arguments.

[5]We slightly abuse notation by thinking of π^* both as a ranking and as an algorithm that outputs the same ranking at each step.

Algorithm 1 Algorithm $\mathrm{OnlineRank}(V, \eta, \mathrm{SortProc}, T)$

given: ground set V, learning rate η, randomized sorting procedure SortProc, time horizon T
set $w_0(u) = 0$ for all $u \in V$
for $t = 1..T$ **do**
  output $\pi_t = \mathrm{SortProc}(V, w_{t-1})$
  observe $s_t : V \mapsto \{0, 1\}$
  set $w_t(u) = w_{t-1}(u) + \eta s_t(u)$ for all $u \in V$
end for

Algorithm 2 Algorithm $\mathrm{QuickSort}(V, w)$

given: ground set V, score function $w : V \mapsto \mathbb{R}$
choose $p \in V$ (pivot) uniformly at random
set $V_L = V_R = \emptyset$
for $v \in V, v \neq p$ **do**
  with probability $\frac{e^{w(v)}}{e^{w(v)} + e^{w(p)}}$ add v to V_L
  otherwise, add v to V_R
end for
return concatenation of $\mathrm{QuickSort}(V_L, w), p, \mathrm{QuickSort}(V_R, w)$

Theorem 3.1. *Let $\tilde{L}$ be an upper bound on $L_T(\pi^*)$. If strategy* OnlineRank *is used with either* SortProc $=$ QuickSort *or* SortProc $=$ PlackettLuce *and with* $\eta = \ln\left(1 + \sqrt{n^2(\log 2)/\tilde{L}}\right)$ *then* $\mathbb{E}[L_T(\mathrm{OnlineRank})]$ *is upper bounded by*

$$L_T(\pi^*) + n\sqrt{(\log 2)\tilde{L}} + n^2(\log 2)/2 \,. \tag{3.1}$$

Additionally, the expected running time of the algorithm per time step is $O(n \log n)$.

Theorem 3.2. *There exists an integer n_0 and some function h such that for all $n \geq n_0$ and $T \geq h(n)$, for any algorithm, the minimax expected total regret in the k-choice setting after T steps is at least $0.003 \cdot n^{3/2}\sqrt{Tk}$.*

In this extended abstract, we only show (in Section 6) the proof for the case $k = 1$, with helpful notes for extending to general k. Also note that we did not make an effort to bound the function h in the theorem, which relies on weak convergence properties guaranteed by the central limit theorem. Better bounds could be derived by considering tight convergence rates of binomial distributions to the normal distribution. We leave this to future work.

4 Comparison With Previous Work

There has been much work on online ranking with various types of feedback and loss functions.

Yasutake et al. (2012) consider online learning for *Kendall-τ rank aggregation*, where at each step nature chooses a permutation $\sigma_t \in S(V)$, and the algorithm incurs the loss

Algorithm 3 Algorithm $\mathrm{PlackettLuce}(V, w)$

given: ground set V, score function $w : V \mapsto \mathbb{R}$
set $U = V$
initialize $\pi(u) = \perp$ for all $u \in V$
for $i = 1..n(= |V|)$ **do**
  choose random $u \in U$ with $\Pr[u] \propto e^{w(u)}$
  set $\pi(u) = i$
  remove u from U
end for
return π

$\sum_{u \neq v} [u, v]_{\pi_t} [v, u]_{\sigma_t}$. Optimizing over this loss summed over $t = 1, \ldots, T$ is NP-Hard even in the offline setting, as shown by Dwork et al. (2001), while our problem is easy to solve offline by simple sorting. Our problem is, however, different, and is not simply a special case of the setting of Yasutake et al. (2012).

A naïve, obvious approach to the problem of predicting rankings, which we state for the purpose of self containment, is by viewing each permutation as one of $n!$ actions, and "tracking" the best permutation using Hedge algorithm. Such schemes (Freund & Schapire (1995); Littlestone & Warmuth (1994)) guarantee an expected the optimal loss is maximal. The distribution over the output permutation $\pi_t \in S(V)$ at each time step t arising in Hedge would assign for any ranking π a probability proportional to $\exp\{-\beta L_{t-1}(\pi)\}$, where $L_{t-1}(\pi)$ is the total loss of π over the first $t-1$ steps, and $\beta > 0$ is some learning rate. This distribution is not equivalent to neither $\mathrm{QuickSort}$ nor $\mathrm{PlackettLuce}$, and it is not even clear how to efficiently draw from it for large n. Hence, one of the main contributions of Suehiro et al. (2012) is in obtaining bounds competitive with Hedge, but with an efficient per time-step running time of $O(n^2)$. Their approach relies on a clever representation of the problem as a special case of optimization over the base polytope of a carefully chosen submodular function.

4.1 Online Linear Optimization View

As stated above, the set $S(V)$ is the vertex set of a well known polytope known as the *permutahedron*. Since our problem is, equivalently, that of online minimization of a linear function of the permutahedron, we can use any online linear optimization tools over discrete subsets of a real vector space, such as Kalai & Vempala (2005). In fact, as we shall see below, the permutahedron is only one reasonable embedding. We mention most relevant results in online linear optimization in what follows.

It is easy to see that for any real vector s, minimizing $\pi \cdot s = \sum \pi(u) s(u)$ over $\pi \in S(V)$ is simply done by ordering the elements of V in decreasing s-value $u_0, u_1, \ldots, u_{n-1}$ and setting $\pi(u_i) = i$ for all i. The highly influencial paper of Kalai & Vempala (2005) suggests Follow the Perturbed Leader (FPL) as a general approach for solving such online linear optimization problems. The bound derived there (using the 'multiplicative version' FPL*) yields an expected regret bound of

$$O\left(n^{3/2}\sqrt{\tilde{L}\log n} + n^3 \log n\right) , \tag{4.1}$$

which is polynomially worse than ours. (A closer inspection reveals that, for example, in the k-choice case FPL* gives an expected regret bound of $O\left(n\sqrt{k\tilde{L}\log n} + n^2 k \log n\right)$.)

As we shall see in Section 7, however, it seems that this suboptimal bound is due to the fact that analysis of FPL* should be done more carefully, taking advantage of the structure of rankings and of the loss functions we consider. We further elaborate on this in Section 7.

Continuing our comparison to previous results, Dani et al. (2007) provide for online linear oprimization problems a regret bound of

$$O(n^2\sqrt{Td\log d \log T}) , \tag{4.2}$$

where d (in our case) is the ambient dimension of the set $S(V) \subseteq \mathbb{R}^n$. Clearly $d = \Theta(n)$, hence this bound is worse than ours by a factor of $\Omega(n\sqrt{\log n \log T})$. (Note that $\tilde{L}$ can always be taken to be the trivial bound of $n^2/2$.)

A less efficient embedding can be done in $\mathbb{R}^{n^2} \equiv \mathbb{R}^{V \times [n]}$ using the Birkhoff-vonNeumann polytope, as follows. Given $\pi \in S(V)$, we define the matrix $A_\pi \in \mathbb{R}^{n^2}$ by

$$A_\pi(u, i) = \begin{cases} i & \pi(u) = i \\ 0 & \text{otherwise} \end{cases} .$$

For an indicator function $s : V \mapsto \{0, 1\}$ we define the embedding $C_s \in \mathbb{R}^{n^2}$ by

$$C_s(u, i) = s(u) .$$

It is clear that $\ell(\pi_t, s_t)$ is equivalently given by $A_{\pi_t} \bullet C_{s_t} := \sum_{u,i} A_{\pi_t}(u, i) C_{s_t}(u, i)$. Using the analysis of FPL* again gives an expected regret bound of

$$O\left(n^2\sqrt{\tilde{L}\log n} + n^4 \log n\right) .$$

Hence this embedding clearly does not help FPL*. Interestingly, recent work of Helmbold & Warmuth (2009) who studied linear optimization over the Birkhoff-vonNeumann polytope *does* offer an improvement over FPL*. Its expected regret bound is (1.1), which is worse than ours by a factor of $\Omega(\sqrt{\log n})$. Their algorithm is also more complicated.

The Single Choice Setting as a Bandit Problem It is worth noting that in the single choice case, given π_t and $\ell(\pi_t, s_t)$ it is possible to recover s_t exactly. This means that we can study the game in the single choice case in the *bandit setting*, where the algorithm only observes the loss at each step.[6] We should practice however due caution with this comparison because standard bandit algorithms work in more difficult environments where only a noisy estimate of the loss vector can be used. The comparison here is presented for the sake of completeness.

The algorithm CombBand of Cesa-Bianchi & Lugosi (2012) (building on the methodology of Dani et al. (2007)) which employs the Birkhoff-vonNeumann embedding achieves an expected regret bound of $O(n^{2.5}\sqrt{T})$, which is worse by a factor of $\Theta(\sqrt{n})$ than our approach.[7] Also, their algorithm is quite complicated, relying on permanent approximations.

We also mention the online linear optimization approach in the bandit setting of Abernethy et al. (2008) in case the search is in a convex polytope. The expected regret for our problem in the single choice setting using their approach is $O(n^2 d\sqrt{\theta(n)T})$, where d is the ambient dimension of the polytope, and $\theta(n)$ is a number that can be bounded by the number of its facets (by Hazan (2013)). Using the permutahedron embedding (in $\mathbb{R}^n$), $d = n-1$ and $\theta(n) = 2^n$. Using the Birkhoff-vonNeumann polytope we have $d = \Theta(n^2)$ and $\theta(n) = \Theta(n)$. For both embeddings and for all cases we study, the bound is at least polynomially worse than ours.

Comparison of Lower Bounds Our lower bound (Theorem 3.2) is a strict refinement of a previous lower bound of Helmbold & Warmuth (2009), who present a lower bound for a worse case of a more general online ranking problem. Their worst case is not an instance of our problem, and their lower bound is asymptotically higher than our upper bound (3.1), and hence not informative for us.

5 Proof of Theorem 3.1

5.1 Regret Performance

In order to prove the regret bound (3.1) for Algorithm 1 with both $\mathrm{SortProc} = \mathrm{QuickSort}$ and $\mathrm{SortProc} = \mathrm{PlackettLuce}$, we start with a simple lemma.

Lemma 5.1. *The random ranking π returned by $\mathrm{SortProc}(V, w)$ satisfies that for any given pair of distinct elements $u, v \in V$, the probability of the event $u \prec_\pi v$ equals $e^{w(u)}/(e^{w(u)} + e^{w(v)})$, for both $\mathrm{SortProc} = \mathrm{QuickSort}$ and $\mathrm{SortProc} = \mathrm{PlackettLuce}$.*

The proof for case QuickSort uses techniques from e.g. Ailon et al. (2008).

Proof. For the case $\mathrm{SortProc} = \mathrm{QuickSort}$, the internal order between u and v can be determined in one of two ways. (i) The element u (resp. v) is chosen as pivot in some recursive call, in which v (resp. u) is part of the input. Denote this event $E_{\{u,v\}}$. (ii) Some element $p \notin \{u, v\}$ is chosen as pivot in a recursive call in which both v and u are part of the input, and in this recursive call the elements u and v are separated (one goes to the left recursion, the other to the right one). Denote this event $E_{p;\{u,v\}}$.

It is clear that the collection of events $\{E_{\{u,v\}}\} \cup \{E_{p;\{u,v\}} : p \in V \setminus \{u, v\}\}$ is a disjoint cover of the probability space of QuickSort. If π is the (random) output, then it is clear from the algorithm that

$$\Pr[u \prec_\pi v | E_{\{u,v\}}] = e^{w(u)}/(e^{w(u)} + e^{w(v)}) .$$

It is also clear, using Bayes rule, that for all $p \notin \{u, v\}$,

$$\begin{aligned} & \Pr[u \prec_\pi v | E_{p;\{u,v\}}] \\ = & \frac{\frac{e^{w(u)}}{e^{w(u)}+e^{w(p)}} \frac{e^{w(p)}}{e^{w(p)}+e^{w(v)}}}{\frac{e^{w(u)}}{e^{w(u)}+e^{w(p)}} \frac{e^{w(p)}}{e^{w(p)}+e^{w(v)}} + \frac{e^{w(v)}}{e^{w(v)}+e^{w(p)}} \frac{e^{w(p)}}{e^{w(p)}+e^{w(u)}}} \\ = & e^{w(u)}/(e^{w(u)} + e^{w(v)}) , \end{aligned}$$

as required. For the case $\mathrm{SortProc} = \mathrm{PlackettLuce}$, for any subset $X \subseteq V$ containing u and v, let F_X denote the event that, when the first of u, v is chosen in Line 3, the value of U (in the main loop) equals X. It is clear that $\{F_X\}$ is a disjoint cover of the probability space of the algorithm. If π now denotes the output of PlackettLuce, then the proof is completed by noticing that for any X, $\Pr[u \prec_\pi v | F_X] = e^{w(u)}/(e^{w(u)} + e^{w(v)})$. □

The conclusion from the lemma is, as we show now, that for each pair $\{u, v\} \subseteq V$ the algorithm plays Hedge over the set of two possible actions, namely $u \prec v$ and $v \prec u$. We now make this precise. For each ordered pair (u, v) of two distinct elements in V, let $\phi_t(u, v) = e^{-\eta \sum_{t'=1}^{t} [s_{t'}(v) - s_{t'}(u)]_+}$. We also let $\phi_0(u, v) = 1$. On one hand, we have

$$\begin{aligned} \sum_{\{u,v\}} \log \frac{\phi_T(u,v) + \phi_T(v,u)}{\phi_0(u,v) + \phi_0(v,u)} & \\ \geq & \sum_{u,v:u \prec_{\pi^*} v} \log \phi_T(u,v) - \binom{n}{2} \log 2 \\ = & -\eta L_T(\pi^*) - \binom{n}{2} \log 2 . \end{aligned} \tag{5.1}$$

[6] Note that generally the bandit setting is more difficult than the full-information setting, where the loss of all actions are known to the algorithm. The fact that the two are equivalent in the single choice case is a special property of the problem.

[7] This is not explicitly stated in their work, and requires plugging in various calculations (which they provide) in the bound provided in their main theorem.

On the other hand,

$$\begin{aligned}\sum_{\{u,v\}} \log \frac{\phi_T(u,v)+\phi_T(v,u)}{\phi_0(u,v)+\phi_0(v,u)} &= \sum_{\{u,v\}}\sum_{t=1}^{T}\log\frac{\phi_t(u,v)+\phi_t(v,u)}{\phi_{t-1}(u,v)+\phi_{t-1}(v,u)} \\ &= \sum_{\{u,v\}}\sum_{t=1}^{T}\log\left(\frac{\phi_{t-1}(u,v)e^{-\eta[s_t(v)-s_t(u)]_+}}{\phi_{t-1}(u,v)+\phi_{t-1}(v,u)} + \frac{\phi_{t-1}(v,u)e^{-\eta[s_t(u)-s_t(v)]_+}}{\phi_{t-1}(u,v)+\phi_{t-1}(v,u)}\right)\end{aligned} \tag{5.2}$$

It is now easily verified that for any u, v,

$$\begin{aligned}&\frac{\phi_{t-1}(u,v)}{\phi_{t-1}(u,v)+\phi_{t-1}(v,u)} \\ =\ & \frac{1}{1+e^{\eta\sum_{t'=1}^{t-1}([s_{t'}(v)-s_{t'}(u)]_+-[s_{t'}(u)-s_{t'}(v)]_+)}} \\ =\ & \frac{1}{1+e^{\eta\sum_{t'=1}^{t-1}(s_{t'}(v)-s_{t'}(u))}} = \frac{1}{1+e^{w_{t-1}(v)-w_{t-1}(u)}} \\ =\ & \frac{e^{w_{t-1}(u)}}{e^{w_{t-1}(u)}+e^{w_{t-1}(v)}}\ .\end{aligned} \tag{5.3}$$

Plugging (5.3) in (5.2) and using Lemma 5.1, we conclude

$$\begin{aligned}\sum_{\{u,v\}} \log \frac{\phi_T(u,v)+\phi_T(v,u)}{\phi_0(u,v)+\phi_0(v,u)} &= \sum_{\{u,,v\}}\sum_{t=1}^{T}\log\mathbb{E}\left[e^{-\eta\ell(\pi_t,s_t,u,v)}\right] \\ &\le \sum_{\{u,v\}}\sum_{t=1}^{T}\log\mathbb{E}\left[1+\ell(\pi_t,s_t,u,v)(e^{-\eta}-1)\right] \\ &\le (e^{-\eta}-1)\sum_{\{u,v\}}\sum_{t=1}^{T}\mathbb{E}\left[\ell(\pi_t,s_t,u,v)\right] \\ &= (e^{-\eta}-1)\mathbb{E}[L_T]\ ,\end{aligned} \tag{5.4}$$

where we used the fact that $e^{-\eta x} \le 1 + x(e^{-\eta} - 1)$ for all $0 \le x \le 1$, and that $\log(1 + x) \le x$ for all x. Combining (5.4) with (5.1), we get

$$\mathbb{E}[L_T] \le \frac{\eta L_T(\pi^*) + \binom{n}{2}\log 2}{1-e^{-\eta}}\ .$$

Using a simple analytical technique from Freund & Schapire (1995), it can be shown that by choosing $\eta = \ln\left(1+\sqrt{n^2(\log 2)/\tilde{L}}\right)$ we get the required result.

5.2 Running Time

It is well known that QuickSort is an algorithm of expected running time of $O(n \log n)$ when used as a simple sorting algorithm. That it has the same run time bound in our case (in which the output is random) has been proven, for example by Ailon & Mohri (2010).

As for PlackettLuce, it is well known (especially among economists who specialize in *discrete choice* theory) that the PlackettLuce noisy sorting algorithm's output is distributed identically to the random process of adding to each coordinate of w (see notation in Algorithm 3) an iid noise variable following an *extreme value of type I* law, and then sorting (using any $O(n \log n)$-time algorithm) the elements in decreasing perturbed-w value.[8] (The extreme value of type I distribution has a cdf of $F(x) = e^{-e^{-x}}$.) A proof of this classic result can be found, for example, in the work of Yellott (1977) (see also the book of Marden (1995)).

6 Proof of Theorem 3.2

We provide a proof for the single choice case in this extended abstract, and include notes for the k-choice case within the proof.

Fix n and V of size n, and assume $T \ge 2n$. Assume the adversary chooses a sequence $u_1, \ldots, u_T$ of single elements so that each element u_i is chosen independently and uniformly at random from V. [For general k, we will select subsets $U_1, \ldots, U_T$ of size k at each step, uniformly at random from the space of such subsets]. For each $u \in V$, let $f(u)$ denote the frequency of u in the sequence, namely $f(u) = |\{i : u_i = u\}|$. Clearly, the minimizer π^* of $L_T(\pi)$ can be taken to be any ranking π satisfying $f(\pi^{-1}(1)) \ge f(\pi^{-1}(2)) \ge \cdots \ge f(\pi^{-1}(n))$. For ease of notation we let $u^j = \pi^{*-1}(j)$, namely the element in position j in π^*. The cost $L_T(\pi^*)$ is given by $L_T(\pi^*) = \sum_{j=1|}^{n} f(u^j)(j-1)$. For any number $x \in [0, T]$, let $m(x) = |\{u \in V : f(u) \ge x\}|$, namely, the number of elements with frequency at least x. Changing order of summation, $L_T(\pi^*)$ can also be written as

$L_T(\pi^*) = \sum_{x=1}^{T}(0 + 1 + 2 + \cdots + (m(x) - 1)) = \frac{1}{2}\sum_{x=1}^{T} m(x)(m(x) - 1)$. This, in turn, equals $\frac{1}{2}\sum_{x=1}^{T}\sum_{u\neq v}\mathbf{1}_{f(u)\ge x}\mathbf{1}_{f(v)\ge x}$.

By linearity of expectation, $\mathbb{E}[L_T(\pi^*)] = \frac{1}{2}\sum_{x=1}^{T}\sum_{u\neq v}\mathbb{E}[\mathbf{1}_{f(u)\ge x}\mathbf{1}_{f(v)\ge x}]$. This clearly equals $\frac{1}{2}n(n-1)\sum_{x=1}^{T}\mathbb{E}[\mathbf{1}_{f(u^*)\ge x}\mathbf{1}_{f(v^*)\ge x}]$, where u^*, v^* are any two fixed, distinct elements of V. Note that $f(u)$ is distributed $B(T, 1/n)$ for any $u \in V$, where $B(N, p)$ denotes Binomial with N trials and probability p of success. In what follows we let $X_{N,p}$ be a random variable distributed $B(N, p)$. Let $\mu = T/n$ be the expectation of $X_{T,1/n}$, and let $\sigma = \sqrt{T(n-1)}/n$ be its standard deviation. [For general k, instead, we have moments of a the binomial with n trials and probability k/n of success.] We will assume for simplicity that μ is an integer (although

[8] Also often known as the *Gumbel* distribution.

this requirement can be easily removed). We will fix an integer $j > 0$ that will be chosen later. We split the last expression as $\mathbb{E}[L_T(\pi^*)] = \alpha + \beta + \gamma$, where

$$\begin{aligned}\alpha &= \frac{1}{2}n(n-1)\sum_{x=1}^{\mu-\lfloor j\sigma\rfloor-1}\mathbb{E}[\mathbf{1}_{f(u^*)\geq x}\mathbf{1}_{f(v^*)\geq x}]\\ \beta &= \frac{1}{2}n(n-1)\sum_{x=\mu-\lfloor j\sigma\rfloor}^{\mu+\lfloor j\sigma\rfloor}\mathbb{E}[\mathbf{1}_{f(u^*)\geq x}\mathbf{1}_{f(v^*)\geq x}]\\ \gamma &= \frac{1}{2}n(n-1)\sum_{x=\mu+\lfloor j\sigma\rfloor+1}^{T}\mathbb{E}[\mathbf{1}_{f(u^*)\geq x}\mathbf{1}_{f(v^*)\geq x}]\,.\end{aligned}$$

Before we bound α, β, γ, first note that for any x, the random variable $(f(u^*)|f(v^*) = x)$ is distributed $B(T - x, 1/(n-1))$. Also, for any x the function $g(x') = \Pr[f(u^*) \geq x | f(v^*) = x']$ is monotonically decreasing in x'. Hence, for any $1 \leq x \leq T$,

$$\begin{aligned}E\quad&[\mathbf{1}_{f(u^*)\geq x}\mathbf{1}_{f(v^*)\geq x}] \qquad (6.1)\\ =\quad&\sum_{x'=x}^{T}\Pr[f(v^*)=x']\cdot\Pr[f(u^*)\geq x|f(v^*)=x']\\ \leq\quad&\sum_{x'=x}^{T}\Pr[f(v^*)=x']\cdot\Pr[f(u^*)\geq x|f(v^*)=x]\\ =\quad&\Pr[f(v^*)\geq x]\cdot\Pr[f(u^*)\geq x|f(v^*)=x]\\ =\quad&\Pr[X_{T,1/n}\geq x]\cdot\Pr[X_{T-x,1/(n-1)}\geq x] \qquad (6.2)\end{aligned}$$

Bounding γ: We use Chernoff bound, stating that for any integer N and probability p,

$$\begin{aligned}&\forall x\in[Np,2Np],\\ &\qquad\Pr[X_{N,p}\geq x] \leq \exp\left\{\frac{-(x-Np)^2}{(3Np)}\right\} \qquad (6.3)\\ &\forall x > 2Np,\\ &\qquad\Pr[X_{N,p}\geq x] \leq \Pr[X_{N,p}\geq 2NP]\,. \qquad (6.4)\end{aligned}$$

Plugging (6.2) in the definition of γ and using (6.3-6.4), we conclude that there exists global integers j, n_0 and a polynomial P such that for all $n \geq n_0$ and $T \geq P(n)$,

$$\gamma \leq 0.001\cdot n(n-1)\sqrt{T/n} \leq 0.001\cdot n^{3/2}\sqrt{T}\,. \qquad (6.5)$$

Bounding β: Using the same j as just chosen, possibly increasing n_0 and applying the central limit theorem, we conclude that there exists a function h such that for all $n \geq n_0$ and $T \geq h(n)$,

$$\beta \leq \frac{1}{2}n(n-1)\left(\sqrt{\frac{T}{n}}+1\right)\sum_{i=-j}^{j}(1-\Phi(i-1/100))^2\,, \qquad (6.6)$$

where Φ is the normal cdf. For notation purposes, let $\Psi(x) = 1 - \Phi(x)$ and $\epsilon = 1/100$. Hence,

$$\begin{aligned}\beta \leq\ &\frac{1}{2}n(n-1)\left(\sqrt{\frac{T}{n}}+1\right)\\ &\times\left(\Phi(-\epsilon)^2+\sum_{i=1}^{j}\left(\Phi(i-\epsilon)^2+\Psi(i+\epsilon)^2\right)\right)\,.\end{aligned}$$

We now make some rough estimates of the normal cdf. The reason for doing these tedious calculations will be made clear shortly. One verifies that $\Phi(-\epsilon) \leq 0.497$, $\Phi(1-\epsilon) \leq 0.839$, $\Phi(2-\epsilon) \leq 0.977$, $\Phi(3-\epsilon) \leq 0.999$, $\Psi(1+\epsilon) \leq 0.157$, $\Psi(2+\epsilon) \leq 0.023$, $\Psi(3+\epsilon) \leq 0.001$. Hence,

$$\begin{aligned}\beta \leq\ &\frac{1}{2}n(n-1)\left(\sqrt{\frac{T}{n}}+1\right)\\ &\times\left(2.929+\sum_{i=4}^{j}\left(\Phi(i-\epsilon)^2+\Psi(i+\epsilon)^2\right)\right)\end{aligned}$$

It is now easy to verify using standard analysis that for all $i \geq 4$,

$$\Phi(i-\epsilon)^2+\Psi(i+\epsilon)^2 \leq 1\,. \qquad (6.7)$$

Therefore,

$$\begin{aligned}\beta &\leq \frac{1}{2}n(n-1)\left(\sqrt{\frac{T}{n}}+1\right)(j-0.07)\\ &\leq \frac{1}{2}n^{3/2}\sqrt{T}(j-0.07)+\frac{1}{2}n^2(j-0.07)\end{aligned}$$

(Note that the crux of the enitre proof is in getting the first summand in the last expression to be $\frac{1}{2}n^{3/2}\sqrt{T}(j-c)$ for some $c > 0$. This is the reason we needed to estimate the normal cdf around small integers, and the inequality (6.7) for larger integers.)

Bounding α is done trivially by using $\mathbb{E}[\mathbf{1}_{f(u^*)\geq x}\mathbf{1}_{f(v^*)\geq x}] \leq 1$. This gives,

$$\begin{aligned}\alpha &\leq \frac{1}{2}n(n-1)\left(\mu-\lfloor j\sigma\rfloor-1\right)\\ &\leq \frac{1}{2}n(n-1)\left(T/n-j\sqrt{T/n}+1\right)\\ &\leq \frac{1}{2}(n-1)T-\frac{1}{2}jn^{3/2}\sqrt{T}+\frac{1}{2}j\sqrt{Tn}+\frac{1}{2}n^2\,.\end{aligned}$$

Combining our bound for α, β, γ, possibly increasing n_0 and the function h, we conclude that there exists a global integer n_0 and a function h such that for all $n \geq n_0$ and $T \geq h(n)$,

$$\mathbb{E}[L_T(\pi^*)] = \alpha+\beta+\gamma \leq \frac{1}{2}(n-1)T - 0.003\cdot n^{3/2}\sqrt{T}\,.$$

On the other hand, we know that for any algorithm, the expected total loss is exactly $\frac{1}{2}T(n-1)$. Indeed, each element u_t in the sequence $u_1, \ldots, u_T$ can be assumed to be randomly drawn after π_t is chosen by the algorithm, hence, the expected loss at time t is exactly $(0+1+\cdots+(n-1))/n = (n-1)/2$. From here the proof conclusion is clear.

7 PlackettLuce, FPL, And Other Ranking Loss Functions

In view of Section 5.2, our main algorithm OnlineRank (Algorithm 1) with the PlackettLuce procedure (Algorithm 3) is, in fact, a type of FPL implementation with respect to the uncertainty distribution chosen to be extreme value of type I. (The QuickSort procedure (Algorithm 2) does not give rise to FPL.)

The reason the analysis of FPL* in the work of Kalai & Vempala (2005) gave rise to the suboptimal bound (4.1) was due to the fact that it viewed the problem in an n dimensional space. A more careful analysis decomposes the problem as $\binom{n}{2}$ separate sub-problems, each corresponding to a pair of elements in V. This analysis also requires setting the learning rate (via the noise shape parameter) differently from the value proposed by Kalai & Vempala (2005). Instead of presenting the details of this particular analysis, we will take advantage of this revised view of FPL* for our problem, and explore a much more general setting.

Consider a setting in which our loss function at time t is defined as $\ell_z(\pi_t, s_t) = \sum_{u \in U} z(\pi_t(u)) \cdot s_t(u)$, where the functional parameter $z : [n] \mapsto \mathbb{R}$ is monotone nondecreasing. This function assigns different importance weights to the n possible positions. So far we studied the linear function $z = z_{\mathrm{LIN}}$ with $z_{\mathrm{LIN}}(i) = i$. Other important functions are, for example z_{NDCG} defined as $z_{\mathrm{NDCG}}(i) = 1/log_2(i+2)$, related to the commonly used NDCG measure from information retrieval Järvelin & Kekäläinen (2002). Each z now gives rise to a linear optimization problem over a "deformed" version of the permutahedron.

Theorem 7.1. *Assume $z(i) = \alpha_0 + \sum_{j=1}^{d} \alpha_j \binom{i}{j}$ for some constant degree $d \geq 1$ and constants $\alpha_1, \ldots, \alpha_d \geq 0$, with $\alpha_d > 0$. Consider the online ranking problem with respect to ℓ_z, as defined above. Let $\tilde{L}$ be an upper bound on $\min_{\pi \in S(V)} \sum_{t=1}^{T} \ell_z(\pi, s_t)$ Let $\xi = \sum_{j=1}^{d} \alpha_j \binom{n}{j+1} j \log j$. Then if the shape parameter ϵ of FPL* of Kalai & Vempala (2005) is taken as $\min\left\{\sqrt{\xi/\tilde{L}}, 1\right\}$, then the expected regret is $O\left(\sqrt{\xi\tilde{L}} + \xi\right)$.*

Note that if, instead, we directly used the analysis of Kalai & Vempala (2005) , then we would take, instead of ξ as in the theorem, $\xi = n\sum_{i=1}^{n} z(i) \log n$. This would give rise to an expected regret which is worse by a factor of $\Theta\left(\sqrt{n\frac{\log n}{\log d}}\right)$, assuming $d, \alpha_1, \ldots, \alpha_d$ are held fixed.

Proof. (Sketch) Let $\mathcal{U}_j$ denote all subsets of V of size exactly j, for $j \in [d+1]$. The idea is, for any fixed $j \in [d+1]$ and $U \in \mathcal{U}_j$, to consider, for the sake of analysis, a virtual online linear optimization problem, defined as follows: Denoting $U = \{u_1, \ldots, u_j\} \subseteq V$, given the score function s_t at time t, an algorithm for the virtual game chooses one of j vectors $v_1^U, \ldots, v_j^U \in \{0,1\}^j$, where $v_j^U(i)$ is 1 if $i = j$ and 0 otherwise. The linear loss in the virtual game, upon an algorithm's choice of some $v_{j'}$ for $j' \in [j]$, is defined as $\sum_{i=1}^{j} s_t(u_i) v_{j'}^U(i)$. In fact, this is an *expert setting*, and the loss is simply $s_t(u_{j'})$.

The key observation is that by playing the FPL* strategy of Kalai & Vempala (2005) on the ranking problem over ℓ_z we are, in fact, simulataneously playing FPL* over the virtual games corresponding to all $U \in \cup_{j=1}^{d+1} \mathcal{U}_j$. More precisely, by choosing a ranking π_t, we are simultaneously choosing, for all $j \in [d+1]$ and $U = \{u_1, \ldots, u_j\} \in \mathcal{U}_j$, the same vector that FPL* would have chosen for the virtual game corresponding to U; This vector is $v_{j'}^U$, where j' is such that $u_{j'}$ is *last* in π_t among $u_1, \ldots, u_j$. It now suffices to notice that

$$\ell_z(\pi, s_t) = \alpha_0 + \sum_{j=1}^{d} \alpha_j \sum_{U \in \mathcal{U}_{j+1}} \ell_t^{\mathrm{vir}}(U) \,, \qquad (7.1)$$

where $\ell_t^{\mathrm{vir}}(U)$ is the loss at time t in the virtual game corresponding to U. (Notice, for example, that the contribution of an element $u = \pi^{-1}(i)$ to the RHS of (7.1) over all virtual games corresponding to U such that $v \in U \in \mathcal{U}_{j+1}$ is exactly $\alpha_j \binom{n}{j} s_t(v)$.) A final step of choosing the best shape parameter ϵ concludes the proof. □

8 Future Work

Our main open problem is, what can be generally done in the bandit setting? Is the algorithm CombBand of Cesa-Bianchi & Lugosi (2012) the optimal for the setting studied here?

Theorem 7.1 did not apply to functions such as z_{NDCG}, leaving open the following additional problem: What are the optimal regret bounds over a given loss function ℓ_z for a given monotone nondecreasing z? What is the algorithmic complexity required for achieving this bound?

Acknowledgements The author acknowledges the support of a Marie Curie International Reintegration Grant PIRG07-GA-2010-268403, an Israel Science Foundation grant number 1271/13 and a Jacobs Technion-Cornell Innovation Institute grant. He also wishes to thank Kohei Hatano, Eiji Takimoto and an anonymous reviewer for enlightening comments.

References

Abernethy, Jacob, Hazan, Elad, and Rakhlin, Alexander. Competing in the dark: An efficient algorithm for bandit linear optimization. In *COLT*, pp. 263–274, 2008.

Ailon, Nir and Mohri, Mehryar. Preference-based learning to rank. *Machine Learning*, 80(2-3):189–211, 2010.

Ailon, Nir, Charikar, Moses, and Newman, Alantha. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008.

Cesa-Bianchi, Nicolò and Lugosi, Gábor. Combinatorial bandits. *J. Comput. Syst. Sci.*, 78(5):1404–1422, 2012.

Dani, Varsha, Hayes, Thomas P., and Kakade, Sham. The price of bandit information for online optimization. In *NIPS*, 2007.

Dwork, Cynthia, Kumar, Ravi, Naor, Moni, and Sivakumar, D. Rank aggregation methods for the web. In *Proceedings of the Tenth International Conference on the World Wide Web (WWW10)*, pp. 613–622, Hong Kong, 2001.

Freund, Yoav and Schapire, Robert E. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, pp. 23–37, 1995.

Hazan, Elad. Private communication, 2013.

Helmbold, David P. and Warmuth, Manfred K. Learning permutations with exponential weights. *J. Mach. Learn. Res.*, 10:1705–1736, December 2009. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1577069.1755841.

Järvelin, Kalervo and Kekäläinen, Jaana. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002. ISSN 1046-8188.

Kalai, Adam and Vempala, Santosh. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, October 2005. ISSN 0022-0000. doi: 10.1016/j.jcss.2004.10.016. URL http://dx.doi.org/10.1016/j.jcss.2004.10.016.

Littlestone, Nick and Warmuth, Manfred K. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, February 1994. ISSN 0890-5401. doi: 10.1006/inco.1994.1009. URL http://dx.doi.org/10.1006/inco.1994.1009.

Marden, John I. *Analyzing and Modeling Rank Data*. Chapman & Hall, 1995.

Suehiro, Daiki, Hatano, Kohei, Kijima, Shuji, Takimoto, Eiji, and Nagano, Kiyohito. Online prediction under submodular constraints. In *Algorithmic Learning Theory*, Lecture Notes in Computer Science, pp. 260–274, 2012.

Yasutake, Shota, Hatano, Kohei, Takimoto, Eiji, and Takeda, Masayuki. Online rank aggregation. *Journal of Machine Learning Research - Proceedings Track*, 25: 539–553, 2012.

Yellott, J. The relationship between Luce's choice axiom, Thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15:109–144, 1977.