Journal of Machine Learning Research 18 (2017) 1-49 Submitted 6/16; Revised 8/17; Published 9/17

Online Learning to Rank with Top-k Feedback

Sougata Chaudhuri [1] SOUGATA@UMICH.EDU
Ambuj Tewari [1,2] TEWRIA@UMICH.EDU
Department of Statistics [1]
Department of Electrical Engineering and Computer Science [2]
University of Michigan
Ann Arbor, MI 48109, USA

Editor: Alexander Rakhlin

Abstract

We consider two settings of online learning to rank where feedback is restricted to top ranked items. The problem is cast as an online game between a learner and sequence of users, over T rounds. In both settings, the learners objective is to present ranked list of items to the users. The learner's performance is judged on the entire ranked list and true relevances of the items. However, the learner receives highly restricted feedback at end of each round, in form of relevances of only the top k ranked items, where $k \ll m$. The first setting is *non-contextual*, where the list of items to be ranked is fixed. The second setting is *contextual*, where lists of items vary, in form of traditional query-document lists. No stochastic assumption is made on the generation process of relevances of items and contexts. We provide efficient ranking strategies for both the settings. The strategies achieve $O(T^{2/3})$ regret, where regret is based on popular ranking measures in first setting and ranking surrogates in second setting. We also provide impossibility results for certain ranking measures and a certain class of surrogates, when feedback is restricted to the top ranked item, i.e. $k = 1$. We empirically demonstrate the performance of our algorithms on simulated and real world data sets.

Keywords: Learning to Rank, Online Learning, Partial Monitoring, Online Bandits, Learning Theory

1. Introduction

Learning to rank (Liu, 2011) is a supervised machine learning problem, where the output space consists of *rankings* of objects. Most learning to rank methods are based on supervised *batch* learning, i.e., rankers are trained on batch data in an offline setting. The accuracy of a ranked list, in comparison to the actual relevance of the documents, is measured by various ranking measures, such as Discounted Cumulative Gain (DCG) (Järvelin and Kekäläinen, 2000), Average Precision (AP) (Baeza-Yates and Ribeiro-Neto, 1999) and others.

Collecting reliable training data can be expensive and time consuming. In certain applications, such as deploying a new web app or developing a custom search engine, collecting large amount of high quality labeled data might be infeasible (Sanderson, 2010). Moreover, a ranker trained from batch data might not be able to satisfy rapidly changing user needs and preferences. Thus, a promising direction of research is development of online ranking

systems, where a ranker is updated on the fly. One type of online ranking models learns from implicit feedback inferred from user clicks on ranked lists (Hofmann et al., 2013). However, there are some potential drawbacks in learning from user clicks. It is possible that the system is designed for explicit ratings but not clicks. Moreover, a clicked item might not actually be relevant to the user and there is also the problem of bias towards top ranked items in inferring feedback from user clicks (Joachims, 2002).

We develop models for online learning of ranking systems, from explicit but *highly restricted* feedback. At a high level, we consider a ranking system which interacts with users over a time horizon, in a sequential manner. At each round, the system presents a ranked list of m items to the user, with the quality of the ranked list judged by the relevance of the items to the user. The relevance of the items, reflecting varying user preferences, is encoded as relevance vectors. The system's objective is to learn from the feedback it receives and update its ranker over time, to satisfy as many users as possible. However, the feedback that the system receives at end of each round is not the full relevance vector, but relevance of only the top k ranked items, where $k \ll m$ (typically, $k = 1$ or 2). We consider two problem settings under the general framework: *non-contextual* and *contextual*. In the first setting, we assume that the set of items to be ranked are fixed (i.e., there are no context on items), with the relevance vectors varying according to users' preferences. In the second setting, we assume that set of items vary, as traditional query-document lists. We highlight two motivating examples for such feedback model, encompassing *privacy* concerns and *economic and user-burden* constraints.

Privacy Concerns: Assume that a medical company wants to build an app to suggest activities (take a walk, meditate, watch relaxing videos, etc.) that can lead to reduction of stress in a certain highly stressed segment of the population. The activities do not have contextual representation and are fixed over time. Not all activities are likely to be equally suitable for everyone under all conditions since the effects of the activities vary depending on the user attributes like age & gender and on the context such as time of day & day of week. The user has liberty to browse through all the suggested activities, and the company would like the user to rate every activity (may be on an $1-5$ scale), reflecting the relevances, so that it can keep refining its ranking strategy. However, in practice, though the user may scan through all suggested activities and have a rough idea about how relevant each one is to her; she is unlikely to give feedback on the usefulness (relevance) of every activity due to privacy concerns and cognitive burden. Hence, in exchange of the user using the app, the company only asks for careful rating of the top 1 or 2 ranked activities. The apps performance would still be based on the full ranked list, compared to the implicit relevance vector that the user generates, but it gets feedback on the relevances of only top 1 or 2 ranked activities.

Economic Constraints: Assume that a small retail company wants to build an app that produces a ranked list of suggestions to a user query, for categories of different products. The app develops features representing the categories, and thus, the interaction with the users happens in a traditional query-documents lists framework (user query and retrieved activities are jointly represented through a feature matrix). Different categories are likely to have varying relevance to different users, depending on user characteristics such as age, gender, etc. Like in the first example, the user has liberty to browse through all the suggestions but she will likely feel too burdened to give carefully considered rating on

each suggestion, unless the company provides some economic incentives to do so. Though the company needs high quality feedback on each suggestion to keep refining the ranking strategy, it cannot afford to give incentives due to budget constraints. Similar to the first example, the company only asks, and possibly pays, for rating on the top 1 or 2 ranked suggestions, in lieu of using the app, but its performance is judged on the full ranked list and implicit relevance vector.

We cast the online learning to rank problem as an online game between a learner and an adversary, played over time horizon T. That is, we do not make any *stochastic* assumption on the relevance vector generation process or the context (features) generation process (in the second problem setting). The adversary is considered to be *oblivious*, i.e., an adversary who generates moves without knowledge of the learner's algorithm. We separately discuss the two problem settings, and our contributions in each, in greater details.

1.1 Non-contextual setting

Existing work loosely related to ranking of a fixed set of items to satisfy diverse user preferences (Radlinski et al., 2008, 2009; Agrawal et al., 2009; Wen et al., 2014) has focused on learning an optimal ranking of a subset of items, to be presented to an user, with performance judged by a simple 0-1 loss. The loss in a round is 0 if among the top k (out of m) items presented to a user, the user finds at least one relevant item. All of the work falls under the framework of *online bandit* learning. In contrast, our model focuses on optimal ranking of the entire list of items, where the performance of the system is judged by practical ranking measures like DCG and AP. The challenge is to decide when and how efficient learning is possible with the highly restricted feedback model. Theoretically, the top k feedback model is neither full-feedback nor bandit-feedback since not even the loss (quantified by some ranking measure) at each round is revealed to the learner. The appropriate framework to study the problem is that of *partial monitoring* (Cesa-Bianchi et al., 2006). A very recent paper shows another practical application of partial monitoring in the stochastic setting (Lin et al., 2014). Recent advances in the classification of partial monitoring games tell us that the minimax regret, in an adversarial setting, is governed by a (two-class) property of the loss and feedback functions, called *global observability* and *local observability* (Bartok et al., 2014; Foster and Rakhlin, 2012).

Our contributions: We instantiate these general observability notions for our problem with top 1 ($k = 1$) feedback. We prove that, for some ranking measures, namely PairwiseLoss (Duchi et al., 2010), DCG and Precision@n (Liu et al., 2007), global observability holds. This immediately shows that the upper bound on regret scales as $O(T^{2/3})$. Specifically for PairwiseLoss and DCG, we further prove that local observability fails, when restricted to the top 1 feedback case, illustrating that their *minimax* regret scales as $\Theta(T^{2/3})$. However, the generic algorithm that enjoys $O(T^{2/3})$ regret for globally observable games necessarily maintains explicit weights on each action in learner's action set. It is impractical in our case to do so, since the learner's action set is the exponentially large set of $m!$ rankings over m objects. We propose a generic algorithm for learning with top k feedback, which uses blocking and a black-box full information algorithm. Specifically, we instantiate the black box algorithm with Follow The Perturbed Leader (FTPL) strategy, which leads to an efficient algorithm achieving $O(T^{2/3})$ regret bound for PairwiseLoss, DCG and

Precision@n, with $O(m \log m)$ time spent per step. Moreover, the regret of our efficient algorithm has a logarithmic dependence on number of learner's actions (i.e., polynomial dependence on m), whereas the generic algorithm has a linear dependence on number of actions (i.e., exponential dependence on m).

For several measures, their *normalized* versions are also considered. For example, the normalized versions of PairwiseLoss, DCG and Precision@n are called AUC (Cortes and Mohri, 2004), NDCG (Järvelin and Kekäläinen, 2002) and AP respectively. We show an unexpected result for the normalized versions: *they do not* admit sub-linear regret algorithms with top 1 feedback. This is despite the fact that the opposite is true for their unnormalized counterparts. Intuitively, the normalization makes it hard to construct an unbiased estimator of the (unobserved) relevance vectors. We are able to translate this intuitive hurdle into a provable impossibility.

We also present some preliminary experiments on simulated data sets to explore the performance of our efficient algorithm and compare its regret to its full information counterpart.

1.2 Contextual Setting

The requirement of having a fixed set of items to rank, in the first part of our work, somewhat limits practical applicability. In fact, in the classic multi-armed bandit problem, while non-contextual bandits have received a lot of attention, the authors Langford and Zhang (2008) mention that "settings with no context information are rare in practice". The second part of our work introduces context, by combining query-level ranking with the explicit but restricted feedback model. At each round, the adversary generates a document list of length m, pertaining to a query. The learner sees the list and produces a real valued score vector to rank the documents. We assume that the ranking is generated by sorting the score vector in descending order of its entries. The adversary then generates a relevance vector but, like in the non-contextual setting, the learner gets to see the relevance of only the top k items of the ranked list. The learner's loss in each round, based on the learner's score vector and the *full* relevance vector, is measured by some continuous ranking surrogates. We focus on continuous surrogates, e.g., the cross entropy surrogate in ListNet (Cao et al., 2007) and hinge surrogate in RankSVM (Joachims, 2002), instead of discontinuous ranking measures like DCG, or AP, because the latter lead to intractable optimization problems in the query-documents setting. Just like in the non-contextual setting, we note that the top k feedback model is neither full feedback nor bandit feedback models. The problem is an instance of partial monitoring, *extended to a setting with side information* (documents list) and an *infinite set of learner's moves* (all real valued score vectors). For such an extension of partial monitoring there exists no generic theoretical or algorithmic framework to the best of our knowledge.

Our contributions: In this setting, first, we propose a general, efficient algorithm for online learning to rank with top k feedback and show that it works in conjunction with a number of ranking surrogates. We characterize the minimum feedback required, i.e., the value of k, for the algorithm to work with a particular surrogate by formally relating the feedback mechanism with the structure of the surrogates. We then apply our general techniques to three convex ranking surrogates and one non-convex surrogate. The

convex surrogates considered are from three major learning to ranking methods: squared loss from a *pointwise* method (Cossock and Zhang, 2008), hinge loss used in the *pairwise* RankSVM (Joachims, 2002) method, and (modified) cross-entropy surrogate used in the *listwise* ListNet (Cao et al., 2007) method. The non-convex surrogate considered is the SmoothDCG surrogate (Chapelle and Wu, 2010). For the three convex surrogates, we establish an $O(T^{2/3})$ regret bound.

The convex surrogates we mentioned above are widely used but are known to fail to be calibrated with respect to NDCG (Ravikumar et al., 2011). Our second contribution is to show that for the entire class of NDCG calibrated surrogates, no online algorithm can have sub-linear (in T) regret with top 1 feedback, i.e., the minimax regret of an online game for any NDCG calibrated surrogate is $\Omega(T)$. The proof for this result relies on exploiting a connection between the construction of optimal adversary strategies for hopeless *finite action* partial monitoring games (Piccolboni and Schindelhauer, 2001) and the structure of NDCG calibrated surrogates. We only focus on NDCG calibrated surrogates for the *impossibility* results since no (convex) surrogate can be calibrated for AP and ERR (Calauzenes et al., 2012). This impossibility result is the first of its kind for a natural partial monitoring problem with side information when the learner's action space is infinite. Note, however, that there does exist work on partial monitoring problems with continuous learner actions, but without side information (Kleinberg and Leighton, 2003; Cesa-Bianchi et al., 2006), and vice versa (Bartók and Szepesvári, 2012; Gentile and Orabona, 2014).

We apply our algorithms on benchmark ranking data sets, demonstrating the ability to efficiently learn a ranking function in an online fashion, from highly restricted feedback.

The rest of the paper is divided into the following sections. Section 2 and its subsections detail the notations, definitions and technicalities associated with online ranking with restricted feedback in the non-contextual setting. Section 3 and its subsections detail the notations, definitions and technicalities associated with online ranking with restricted feedback in the contextual setting. Section 4 demonstrates the performance of our algorithms on simulated and commercial data sets. Section 5 discusses open questions and future directions of research.

2. Online Ranking with Restricted Feedback- Non Contextual Setting

All proofs not in main text are in Appendix A.

2.1 Notation and Preliminaries

The fixed m items to be ranked are numbered $\{1, 2, \ldots, m\}$. A permutation σ gives a mapping from ranks to items and its inverse σ^{-1} gives a mapping from items to ranks. Thus, $\sigma^{-1}(i) = j$ means item i is placed at position j while $\sigma(i) = j$ means item j is placed at position i. The supervision is in form of a relevance vector $R = \{0, 1, \ldots, n\}^m$, representing relevance of each document to the query. If $n = 1$, the relevance vector is binary graded. For $n > 1$, relevance vector is multi-graded. $R(i)$ denotes ith component of R, i.e., relevance of item i. The subscript t is exclusively used to denote time t. We denote $\{1, \ldots, n\}$ by $[n]$. The learner can choose from $m!$ actions (permutations) whereas nature/adversary can choose from $(n + 1)^m$ outcomes (when there are n relevance levels, $n \geq 1$). We sometimes refer to the learner's ith action (in some fixed ordering of $m!$ available

actions) as σ_i (resp. adversary's ith action as R_i). Note that σ_i^{-1} simply means that we are viewing permutation σ_i as mapping from items to ranks. Also, a vector can be row or column vector depending on context.

At round t, the learner outputs a permutation (ranking) σ_t of the objects (possibly using some internal randomization, based on feedback history so far), and simultaneously, adversary generates relevance vector R_t. The quality of σ_t is judged against R_t by some ranking measure RL. *Crucially, only the relevance of the top k ranked objects are revealed to the learner at end of round t.* Thus, the learner gets to know neither R_t (full information problem) nor $RL(\sigma_t, R_t)$ (bandit problem). The objective of the learner is to minimize the expected regret with respect to best permutation in hindsight:

$$\mathbb{E}_{\sigma_1,\ldots,\sigma_T}\left[\sum_{t=1}^{T} RL(\sigma_t, R_t)\right] - \min_{\sigma} \sum_{t=1}^{T} RL(\sigma, R_t). \tag{1}$$

When RL is a gain, not loss, we need to negate the quantity above. The worst-case regret of a learner strategy is its maximal regret over all possible choices of $R_1, \ldots, R_T$. The **minimax regret** is the minimal worst-case regret over all learner strategies.

2.2 Ranking Measures

We consider ranking measures which can be expressed in the form $f(\sigma) \cdot R$, where the function $f : \mathbb{R}^m \to \mathbb{R}^m$ is composed of m copies of a univariate, monotonic, scalar valued function. Thus, $f(\sigma) = [f^s(\sigma^{-1}(1)), f^s(\sigma^{-1}(2)), \ldots, f^s(\sigma^{-1}(m))]$, where $f^s : \mathbb{R} \to \mathbb{R}$. Monotonic (increasing) means $f^s(\sigma^{-1}(i)) \geq f^s(\sigma^{-1}(j))$, whenever $\sigma^{-1}(i) > \sigma^{-1}(j)$. Monotonic (decreasing) is defined similarly. The following popular ranking measures can be expressed in the form $f(\sigma) \cdot R$.

PairwiseLoss & SumLoss: PairwiseLoss is restricted to binary relevance vectors and defined as:

$$PL(\sigma, R) = \sum_{i=1}^{m}\sum_{j=1}^{m} \mathbb{1}(\sigma^{-1}(i) < \sigma^{-1}(j))\mathbb{1}(R(i) < R(j))$$

PairwiseLoss cannot be directly expressed in the form of $f(\sigma) \cdot R$. Instead, we consider **SumLoss**, defined as:

$$SumLoss(\sigma, R) = \sum_{i=1}^{m} \sigma^{-1}(i)\ R(i)$$

SumLoss has the form $f(\sigma) \cdot R$, where $f(\sigma) = \sigma^{-1}$. It has been shown by Ailon (2014) that regret under the two measures are equal:

$$\sum_{t=1}^{T} PL(\sigma_t, R_t) - \sum_{t=1}^{T} PL(\sigma, R_t) = \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) - \sum_{t=1}^{T} SumLoss(\sigma, R_t). \tag{2}$$

Discounted Cumulative Gain: DCG is a gain function which admits non-binary relevance vectors and is defined as:

$$DCG(\sigma, R) = \sum_{i=1}^{m} \frac{2^{R(i)} - 1}{\log_2(1 + \sigma^{-1}(i))}$$

and becomes $\sum_{i=1}^{m} \frac{R(i)}{\log_2(1+\sigma^{-1}(i))}$ for $R(i) \in \{0,1\}$. Thus, for binary relevance, $DCG(\sigma, R)$ has the form $f(\sigma) \cdot R$, where $f(\sigma) = [\frac{1}{\log_2(1+\sigma^{-1}(1))}, \frac{1}{\log_2(1+\sigma^{-1}(2))}, \ldots, \frac{1}{\log_2(1+\sigma^{-1}(m))}]$.

Precision@n Gain: Precision@n is a gain function restricted to binary relevance and is defined as

$$Precision@n(\sigma, R) = \sum_{i=1}^{m} \mathbb{1}(\sigma^{-1}(i) \leq n)\ R(i)$$

Precision@n can be written as $f(\sigma) \cdot R$ where $f(\sigma) = [\mathbb{1}(\sigma^{-1}(1) < n), \ldots, \mathbb{1}(\sigma^{-1}(m) < n)]$. It should be noted that for $n = k$ (i.e., when feedback is on top n items), feedback is actually the same as full information feedback, for which efficient algorithms already exist (Kalai and Vempala, 2005).

Normalized measures are not of the form $\mathbf{f}(\sigma) \cdot \mathbf{R}$: PairwiseLoss, DCG and Precision@n are unnormalized versions of popular ranking measures, namely, Area Under Curve (AUC), Normalized Discounted Cumulative Gain (NDCG) and Average Precision (AP) respectively. None of these can be expressed in the form $f(\sigma) \cdot R$.

NDCG: NDCG is a gain function, admits non-binary relevance and is defined as:

$$NDCG(\sigma, R) = \frac{1}{Z(R)} \sum_{i=1}^{m} \frac{2^{R(i)} - 1}{\log_2(1 + \sigma^{-1}(i))}$$

and becomes $\frac{1}{Z(R)} \sum_{i=1}^{m} \frac{R(i)}{\log_2(1+\sigma^{-1}(i))}$ for $R(i) \in \{0,1\}$. Here $Z(R) = \max_{\sigma} \sum_{i=1}^{m} \frac{2^{R(i)}-1}{\log_2(1+\sigma^{-1}(i))}$ is the normalizing factor ($Z(R) = \max_{\sigma} \sum_{i=1}^{m} \frac{R(i)}{\log_2(1+\sigma^{-1}(i))}$ for binary relevance). It can be clearly seen that $NDCG(\sigma, R) = f(\sigma) \cdot g(R)$, where $f(\sigma)$ is same as in DCG but $g(R) = \frac{R}{Z(R)}$ is non-linear in R.

AP: Average Precision is a gain function, restricted to binary relevance and is defined as:

$$AP(\sigma, R) = \frac{1}{\|R\|_1} \sum_{i=1}^{m} \frac{\sum_{j \leq i} \mathbb{1}(R(\sigma(j)) = 1)}{i} \mathbb{1}(R(\sigma(i) = 1))$$

It can be clearly seen that AP cannot be expressed in the form $f(\sigma) \cdot R$.

AUC: AUC is a loss function, restricted to binary relevance and is defined as:

$$AUC(\sigma, R) = \frac{1}{N(R)} \sum_{i=1}^{m} \sum_{j=1}^{m} \mathbb{1}(\sigma^{-1}(i) < \sigma^{-1}(j)) \mathbb{1}(R(i) < R(j))$$

where $N(R) = (\sum_{i=1}^{m} \mathbb{1}(R(i) = 1)) \cdot (m - \sum_{i=1}^{m} \mathbb{1}(R(i) = 1))$. It can be clearly seen that AUC cannot be expressed in the form $f(\sigma) \cdot R$.

Note: We will develop our subsequent theory and algorithms for binary valued relevance vectors, and show how they can be extended to multi-graded vectors when ranking measure is DCG/NDCG.

2.3 Relevant Definitions from Partial Monitoring

We develop all results in context of SumLoss, with binary relevance vector. We then extend the results to other ranking measures. Our main results on regret bounds build on some

Table 1: Loss matrix L for $m = 3$

Objects	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
123	000	001	010	011	100	101	110	111
$\sigma_1 = 123$	0	3	2	5	1	4	3	6
$\sigma_2 = 132$	0	2	3	5	1	3	4	6
$\sigma_3 = 213$	0	3	1	4	2	5	3	6
$\sigma_4 = 231$	0	1	3	4	2	3	5	6
$\sigma_5 = 312$	0	2	1	3	3	5	4	6
$\sigma_6 = 321$	0	1	2	3	3	4	5	6

Table 2: Feedback matrix H for $m = 3$

Objects	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
123	000	001	010	011	100	101	110	111
$\sigma_1 = 123$	0	0	0	0	1	1	1	1
$\sigma_2 = 132$	0	0	0	0	1	1	1	1
$\sigma_3 = 213$	0	0	1	1	0	0	1	1
$\sigma_4 = 231$	0	1	0	1	0	1	0	1
$\sigma_5 = 312$	0	0	1	1	0	0	1	1
$\sigma_6 = 321$	0	1	0	1	0	1	0	1

of the theory for abstract partial monitoring games developed by Bartok et al. (2014) and Foster and Rakhlin (2012). For ease of understanding, we reproduce the relevant notations and definitions in context of SumLoss. *We will specifically mention when we derive results for top k feedback, with general k, and when we restrict to top* 1 *feedback.*

Loss and Feedback Matrices: The online learning game with the SumLoss measure and top 1 feedback can be expressed in form of a pair of *loss matrix* and *feedback matrix*. The *loss matrix* L is an $m! \times 2^m$ dimensional matrix, with rows indicating the learner's actions (permutations) and columns representing adversary's actions (relevance vectors). The entry in cell (i, j) of L indicates loss suffered when learner plays action i (i.e., σ_i) and adversary plays action j (i.e., R_j), that is, $L_{i,j} = \sigma_i^{-1} \cdot R_j = \sum_{k=1}^{m} \sigma_i^{-1}(k) R_j(k)$. The *feedback matrix* H has same dimension as *loss matrix*, with (i, j) entry being the relevance of top ranked object, i.e., $H_{i,j} = R_j(\sigma_i(1))$. When the learner plays action σ_i and adversary plays action R_j, the true loss is $L_{i,j}$, while the feedback received is $H_{i,j}$.

Table 1 and 2 illustrate the matrices, with number of objects $m = 3$. In both the tables, the permutations indicate rank of each object and relevance vector indicates relevance of each object. For example, $\sigma_5 = 312$ means object 1 is ranked 3, object 2 is ranked 1 and object 3 is ranked 2. $R_5 = 100$ means object 1 has relevance level 1 and other two objects have relevance level 0. Also, $L_{3,4} = \sigma_3 \cdot R_4 = \sum_{i=1}^{3} \sigma_3^{-1}(i) R_4(i) = 2 \cdot 0 + 1 \cdot 1 + 3 \cdot 1 = 4$; $H_{3,4} = R_4(\sigma_3(1)) = R_4(2) = 1$. Other entries are computed similarly.

Let $\ell_i \in \mathbb{R}^{2^m}$ denote row i of L. Let Δ be the probability simplex in $\mathbb{R}^{2^m}$, i.e., $\Delta = \{p \in \mathbb{R}^{2^m} : \forall\ 1 \leq i \leq 2^m,\ p_i \geq 0,\ \sum p_i = 1\}$. The following definitions, given for abstract problems by Bartok et al. (2014), has been refined to fit our problem context.

Definition 1: Learner action i is called optimal under distribution $p \in \Delta$, if $\ell_i \cdot p \leq \ell_j \cdot p$, for all other learner actions $1 \leq j \leq m!,\ j \neq i$. For every action $i \in [m!]$, probability cell of i is defined as $C_i = \{p \in \Delta : \text{action } i \text{ is optimal under } p\}$. If a non-empty cell C_i is $2^m - 1$ dimensional (i.e, elements in C_i are defined by only 1 equality constraint), then associated action i is called *Pareto-optimal*.

Note that since entries in H are relevance levels of objects, there can be maximum of 2 distinct elements in each row of H, i.e., 0 or 1 (assuming binary relevance).

Definition 2: The *signal matrix* S_i, associated with learner's action σ_i, is a matrix with 2 rows and 2^m columns, with each entry 0 or 1, i.e., $S_i \in \{0,1\}^{2 \times 2^m}$. The entries of ℓth column of S_i are respectively: $(S_i)_{1,\ell} = \mathbb{1}(H_{i,\ell} = 0)$ and $(S_i)_{2,\ell} = \mathbb{1}(H_{i,\ell} = 1)$.

Note that by definitions of signal and feedback matrices, the 2nd row of S_i (2nd column of $S_i^\top$)) is precisely the ith row of H. The 1st row of S_i (1st column of $S_i^\top$)) is the (boolean) complement of ith row of H.

2.4 Minimax Regret for SumLoss

The minimax regret for SumLoss, restricted to top 1 feedback, will be established by showing that: a) SumLoss satisfies *global observability*, and b) it does not satisfy *local observability*.

2.4.1 Global Observability

Definition 3: The condition of *global observability* holds, w.r.t. loss matrix L and feedback matrix H, if for every pair of learner's actions $\{\sigma_i, \sigma_j\}$, it is true that $\ell_i - \ell_j \in \oplus_{k \in [m!]} Col(S_k^\top)$, where Col refers to column space (i.e., the vector difference belongs in the column span).

The global observability condition states that the (vector) loss difference between any pair of learner's actions has to belong to the vector space spanned by columns of (transposed) signal matrices corresponding to all possible learner's actions. We derive the following theorem on global observability for *SumLoss*.

Theorem 1. *The global observability condition, as per Definition 3, holds w.r.t. loss matrix L and feedback matrix H defined for SumLoss, for any $m \geq 1$.*

Proof. For any σ_a (learner's action) and R_b (adversary's action), we have (please see the end of the proof for explanation of notations and equalities):

$$L_{a,b} = \sigma_a^{-1} \cdot R_b = \sum_{i=1}^{m} \sigma_a^{-1}(i) R_b(i) \stackrel{1}{=} \sum_{j=1}^{m} j\ R_b(\sigma_a(j)) \stackrel{2}{=} \sum_{j=1}^{m} j\ R_b(\tilde{\sigma}_{j(a)}(1)) \stackrel{3}{=} \sum_{j=1}^{m} j\ (S^\top_{\tilde{\sigma}_{j(a)}})_{R_b,2}.$$

Thus, we have

$$\ell_a = [L_{a,1}, L_{a,2}, \ldots, L_{a,2^m}] =$$
$$[\sum_{j=1}^{m} j\ (S^\top_{\tilde{\sigma}_{j(a)}})_{R_1,2}, \sum_{j=1}^{m} j\ (S^\top_{\tilde{\sigma}_{j(a)}})_{R_2,2}, .., \sum_{j=1}^{m} j\ (S^\top_{\tilde{\sigma}_{j(a)}})_{R_{2^m},2}] \stackrel{4}{=} \sum_{j=1}^{m} j\ (S^\top_{\tilde{\sigma}_{j(a)}})_{:,2}\ (:,2 \text{ indicates 2nd column}).$$

Equality 4 shows that ℓ_a is in the column span of m of the $m!$ possible (transposed) signal matrices, specifically in the span of the 2nd columns of those (transposed) m matrices. Hence, for all actions σ_a, it holds that $\ell_a \in \oplus_{k\in[m!]} Col(S_k^\top)$. This implies that $\ell_a - \ell_b \in \oplus_{k\in[m!]} Col(S_k^\top),\ \forall\ \sigma_a, \sigma_b$.

1. Equality 1 holds because $\sigma_a^{-1}(i) = j \Rightarrow i = \sigma_a(j)$.

2. Equality 2 holds because of the following reason. For any permutation σ_a and for every $j \in [m]$, $\exists$ a permutation $\tilde{\sigma}_{j(a)}$, s.t. the object which is assigned rank j by σ_a is the same object assigned rank 1 by $\tilde{\sigma}_{j(a)}$, i.e., $\sigma_a(j) = \tilde{\sigma}_{j(a)}(1)$.

3. In Equality 3, $(S^\top_{\tilde{\sigma}^{-1}_{j(a)}})_{R_b,2}$ indicates the R_bth row and 2nd column of (transposed) signal matrix $S_{\tilde{\sigma}_{j(a)}}$, corresponding to learner action $\tilde{\sigma}_{j(a)}$. Equality 3 holds because $R_b(\tilde{\sigma}_{j(a)}(1))$ is the entry in the row corresponding to action $\tilde{\sigma}_{j(a)}$ and column corresponding to action R_b of H (see Definition 2).

4. Equality 4 holds from the observation that for a particular j, $[(S^\top_{\tilde{\sigma}_{j(a)}})_{R_1,2}, (S^\top_{\tilde{\sigma}_{j(a)}})_{R_2,2}, \ldots, (S^\top_{\tilde{\sigma}_{j(a)}})_{R_{2^m},2}]$ forms the 2nd column of $(S^\top_{\tilde{\sigma}_{j(a)}})$, i.e., $(S^\top_{\tilde{\sigma}_{j(a)}})_{:,2}$. □

2.4.2 Local Observability

Definition 4: Two Pareto-optimal (learner's) actions i and j are called *neighboring actions* if $C_i \cap C_j$ is a $(2^m - 2)$ dimensional polytope (where C_i is probability cell of action σ_i). The *neighborhood action set* of two neighboring (learner's) actions i and j is defined as $N^+_{i,j} = \{k \in [m!] : C_i \cap C_j \subseteq C_k\}$.

Definition 5: A pair of neighboring (learner's) actions i and j is said to be locally observable if $\ell_i - \ell_j \in \oplus_{k\in N^+_{i,j}} Col(S_k^\top)$. The condition of *local observability* holds if every pair of neighboring (learner's) actions is locally observable.

We now show that local observability condition fails for L, H under SumLoss. First, we present the following two lemmas characterizing Pareto-optimal actions and neighboring actions for SumLoss.

Lemma 2. *For SumLoss, each of learner's action σ_i is Pareto-optimal, where Pareto-optimality has been defined in Definition 1.*

Proof. For any $p \in \Delta$, we have $\ell_i \cdot p = \sum_{j=1}^{2^m} p_j\ (\sigma_i^{-1} \cdot R_j) = \sigma_i^{-1} \cdot (\sum_{j=1}^{2^m} p_j R_j) = \sigma_i^{-1} \cdot \mathbb{E}[R]$, where the expectation is taken w.r.t. p. $l_i \cdot p$ is minimized when ranking of objects according to σ_i and expected relevance of objects are in opposite order. That is, the object with highest expected relevance is ranked 1 and so on. Formally, $l_i \cdot p$ is minimized when $\mathbb{E}[R(\sigma_i(1))] \geq \mathbb{E}[R(\sigma_i(2))] \geq \ldots \geq \mathbb{E}[R(\sigma_i(m))]$.

Thus, for action σ_i, probability cell is defined as $C_i = \{p \in \Delta : \sum_{j=1}^{2^m} p_j = 1,\ \mathbb{E}[R(\sigma_i(1))] \geq \mathbb{E}[R(\sigma_i(2))] \geq \ldots \geq \mathbb{E}[R(\sigma_i(m))]\}$. Note that, $p \in C_i$ iff action i is optimal w.r.t. p. Since C_i is obviously non-empty and it has only 1 equality constraint (hence $2^m - 1$ dimensional), action i is Pareto optimal.

The above holds true for all learner's actions σ_i. □

Lemma 3. *A pair of learner's actions $\{\sigma_i,\ \sigma_j\}$ is a neighboring actions pair, if there is exactly one pair of objects, numbered $\{a, b\}$, whose positions differ in σ_i and σ_j. Moreover, a needs to be placed just before b in σ_i and b needs to placed just before a in σ_j.*

Proof. From Lemma 2, we know that every one of learner's actions is Pareto-optimal and C_i, associated with action σ_i, has structure $C_i = \{p \in \Delta : \sum_{j=1}^{2^m} p_j = 1,\ \mathbb{E}[R(\sigma_i(1))] \geq \mathbb{E}[R(\sigma_i(2))] \geq \ldots \geq \mathbb{E}[R(\sigma_i(m))]\}$.

Let $\sigma_i(k) = a,\ \sigma_i(k+1) = b$. Let it also be true that $\sigma_j(k) = b,\ \sigma_j(k+1) = a$ and $\sigma_i(n) = \sigma_j(n),\ \forall n \neq \{k,\ k+1\}$. Thus, objects in $\{\sigma_i, \sigma_j\}$ are same in all places except in a pair of consecutive places where the objects are interchanged.

Then, $C_i \cap C_j = \{p \in \Delta : \sum_{j=1}^{2^m} p_j = 1,\ \mathbb{E}[R(\sigma_i(1)] \geq \ldots \geq \mathbb{E}[R(\sigma_i(k)] = \mathbb{E}[R(\sigma_i(k+1)] \geq \ldots \geq \mathbb{E}[R(\sigma_i(m)]\}$. Hence, there are two equalities in the non-empty set $C_i \cap C_j$ and it is an $(2^m - 2)$ dimensional polytope. Hence condition of Definition 4 holds true and $\{\sigma_i, \sigma_j\}$ are neighboring actions pair.

□

Lemma 2 and 3 are used to establish the following result:

Theorem 4. *The local observability condition, as per Definition 5, fails w.r.t. loss matrix L and feedback matrix H defined for SumLoss, already at $m = 3$.*

Proof. We will explicitly show that local observability condition fails by considering the case when number of objects is $m = 3$. Specifically, action pair $\{\sigma_1,\ \sigma_2\}$, in Table 1 are neighboring actions, using Lemma 3 . Now every other action $\{\sigma_3, \sigma_4, \sigma_5, \sigma_6\}$ either places object 2 at top or object 3 at top. It is obvious that the set of probabilities for which $E[R(1)] \geq E[R(2)] = E[R(3)]$ cannot be a subset of any C_3, C_4, C_5, C_6. From Def. 4, the neighborhood action set of actions $\{\sigma_1, \sigma_2\}$ is precisely σ_1 and σ_2 and contains no other actions. By definition of signal matrices $S_{\sigma_1},\ S_{\sigma_2}$ and entries $\ell_1,\ \ell_2$ in Table 1 and 2, we have,

$$\begin{aligned} S_{\sigma_1} = S_{\sigma_2} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \\ \ell_1 - \ell_2 &= \begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 1 & -1 & 0 \end{bmatrix}. \end{aligned} \tag{3}$$

It is clear that $\ell_1 - \ell_2 \notin Col(S_{\sigma_1}^\top)$. Hence, Definition 5 fails to hold. □

2.5 Minimax Regret Bound

We establish the minimax regret for SumLoss by combining results on global and local observability. First, we get a lower bound by combining our Theorem 4 with Theorem 4 of Bartok et al. (2014).

Corollary 5. *Consider the online game for SumLoss with top-1 feedback and $m = 3$. Then, for every learner's algorithm, there is an adversary strategy generating relevance vectors, such that the expected regret of the learner is $\Omega(T^{2/3})$.*

The fact that the game is globally observable (Theorem 1), combined with Theorem 3.1 in Cesa-Bianchi et al. (2006), gives an algorithm (inspired by the algorithm originally given in Piccolboni and Schindelhauer (2001)) obtaining $O(T^{2/3})$ regret.

Corollary 6. *The algorithm in Figure 1 of Cesa-Bianchi et al. (2006) achieves $O(T^{2/3})$ regret bound for SumLoss.*

However, the algorithm in Cesa-Bianchi et al. (2006) is intractable in our setting since the algorithm necessarily enumerates all the actions of the learner in each round, which is exponential in m in our case ($m!$ to be exact). Moreover, the regret bound of the algorithm also has a linear dependence on the number of actions, which renders the bound useless.

Discussion: The results above establish that the minimax regret for SumLoss, *restricted to top*-1 *feedback*, is $\Theta(T^{2/3})$. Theorem 4 of Bartok et al. (2014) says the following: A partial monitoring game which is both globally and locally observable has minimax regret $\Theta(T^{1/2})$, while a game which is globally observable but *not* locally observable has minimax regret $\Theta(T^{2/3})$. In Theorem 1, we proved global observability, when feedback is restricted to relevance of top ranked item. The global observability result automatically extends to feedback on top k items, for $k > 1$. This is because for top k feedback, with $k > 1$, the learner receives strictly greater information at end of each round than top 1 feedback (for example, the learner can just throw away relevance feedback on items ranked 2nd onwards). So, with top k feedback, for general k, the game will remain at least globally observable. In fact, our algorithm in the next section will achieve $O(T^{2/3})$ regret bound for SumLoss with top k feedback, $k \geq 1$. However, the same is not true for failure of local observability. Feedback on more than top ranked item can make the game strictly easier for the learner and may make local observability condition hold, for some $k > 1$. In fact, for $k = m$ (full feedback), the game will be a simple full information game (disregarding computational complexity), and hence locally observable.

2.6 Algorithm for Obtaining Minimax Regret under SumLoss with Top k Feedback

We first provide a general algorithmic framework for getting an $O(T^{2/3})$ regret bound for SumLoss, with feedback on top k ranked items per round, for $k \geq 1$. We then instantiate a specific algorithm, which spends $O(m \log m)$ time per round (thus, highly efficient) and obtains a regret of rate $O(poly(m)\ T^{2/3})$.

2.6.1 General Algorithmic Framework

Our algorithm combines *blocking* with a randomized *full information* algorithm. We first divide time horizon T into blocks (referred to as blocking). Within each block, we allot a small number of rounds for pure *exploration*, which allows us to estimate the average of the full relevance vectors generated by the adversary in that block. The estimated average vector is cumulated over blocks and then fed to a full information algorithm for the next block. The randomized full information algorithm *exploits* the information received at the beginning of the block to maintain distribution over permutations (learner's actions). In each round in the new block, actions are chosen according to the distribution and presented to the user.

The *key property* of the randomized full information algorithm is this: the algorithm should have an expected regret rate of $O(C\sqrt{T})$, where the regret is the difference between cumulative loss of the algorithm and cumulative loss of best action in hindsight, over T rounds, and C is a parameter specific to the full information algorithm.

Our algorithm is motivated by the reduction from bandit-feedback to full feedback scheme given in Blum and Mansour (2007). However, the reduction *cannot be directly applied to our problem*, because we are not in the bandit setting and hence do not know loss of any action. Further, the algorithm of Blum and Mansour (2007) necessarily spends N rounds per block to try out *each* of the N available actions — this is impractical in our setting since $N = m!$.

Algorithm 1 describes our approach. A key aspect is the formation of estimate of average relevance vector of a block (line 16), for which we have the following lemma:

Lemma 7. *Let the average of (full) relevance vectors over the time period $\{1, 2, \ldots, t\}$ be denoted as $R^{avg}_{1:t}$, that is, $R^{avg}_{1:t} = \sum_{n=1}^{t} \frac{R_n}{t} \in \mathbb{R}^m$. Let $\{i_1, i_2, \ldots, i_{\lceil m/k \rceil}\}$ be $\lceil m/k \rceil$ arbitrary time points, chosen uniformly at random, without replacement, from $\{1, \ldots, t\}$. At time point i_j, only k distinct components of relevance vector R_{i_j}, i.e., $\{R_{i_j}(k \cdot (j-1)+1), R_{i_j}(k \cdot (j-1)+2), \ldots, R_{i_j}(k \cdot j)\}$, become known, $\forall j \in \{1, \ldots, \lceil m/k \rceil\}$ (for $j = \lceil m/k \rceil$, there might be less than k components available). Then the vector formed from the m revealed components, i.e. $\hat{R}_t = [R_{i_j}(k \cdot (j-1)+1), R_{i_j}(k \cdot (j-1)+2), \ldots, R_{i_j}(k \cdot j)]_{\{j=1,2,\ldots,\lceil m/k \rceil\}}$ is an unbiased estimator of $R^{avg}_{1:t}$.*

Proof. We can write $\hat{R}_t = \sum_{j=1}^{\lceil m/k \rceil} \sum_{\ell=1}^{k} R_{i_j}(k \cdot (j-1)+\ell) e_{k \cdot (j-1)+\ell}$, where e_i is the m dimensional standard basis vector along coordinate j. Then, taking expectation over the randomly chosen time points, we have: $E_{i_1,\ldots,i_{\lceil m/k \rceil}}(\hat{R}_t) = \sum_{j=1}^{\lceil m/k \rceil} E_{i_j}[\sum_{\ell=1}^{k} R_{i_j}(k \cdot (j-1)+\ell) e_{k \cdot (j-1)+\ell}] = \sum_{j=1}^{m} \sum_{\ell=1}^{k} \sum_{n=1}^{t} \frac{R_n(k \cdot (j-1)+\ell) e_{k \cdot (j-1)+\ell}}{t} = R^{avg}_{1:t}$. □

Suppose we have a full information algorithm whose regret in T rounds is upper bounded by $C\sqrt{T}$ for some constant C and let C^I be the maximum loss that the learner can suffer in a round. Note that C^I depends on the loss used and on the range of the relevance scores. We have the following regret bound, obtained from application of Algorithm 1 on SumLoss with top k feedback.

Theorem 8. *Let C, C^I be the constants defined above. The expected regret under SumLoss, obtained by applying Algorithm 1, with relevance feedback on top k ranked items per round ($k \geq 1$), and the expectation being taken over randomized learner's actions σ_t, is*

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] - \min_{\sigma} \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) \leq C^I \lceil m/k \rceil K + C\frac{T}{\sqrt{K}}. \quad (4)$$

Optimizing over block size K, the final regret bound is:

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] - \min_{\sigma} \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) \leq 2(C^I)^{1/3} C^{2/3} \lceil m/k \rceil^{1/3} T^{2/3}. \quad (5)$$

2.6.2 Computationally Efficient Algorithm with FTPL

We instantiate our general algorithm with *Follow The Perturbed Leader* (FTPL) full information algorithm (Kalai and Vempala, 2005). The following choices need to be made in Algorithm 1 to implement FTPL as the full information algorithm:

Algorithm 1 RankingwithTop-kFeedback(RTop-kF)- Non Contextual

1: T = Time horizon, K = No. of (equal sized) blocks, FI= randomized full information algorithm.
2: Time horizon divided into equal sized blocks $\{B_1, \ldots, B_K\}$, where $B_i = \{(i-1)(T/K)+1, \ldots, i(T/K)\}$.
3: Initialize $\hat{s}_0 = \mathbf{0} \in \mathbb{R}^m$. Initialize any other parameter specific to FI.
4: **For** $i = 1, \ldots, K$
5: Select $\lceil m/k \rceil$ time points $\{i_1, \ldots, i_{\lceil m/k \rceil}\}$ from block B_i, uniformly at random, without replacement.
6: Divide the m items into $\lceil m/k \rceil$ cells, with k distinct items in each cell.[1]
7: **For** $t \in B_i$
8: **If** $t = i_j \in \{i_1, \ldots, i_{\lceil m/k \rceil}\}$
9: **Exploration round**:
10: Output any permutation σ_t which places items of jth cell in top k positions (in any order).
11: Receive feedback as relevance of top k items of σ_t (i.e., items of jth cell).
12: **Else**
13: **Exploitation round**:
14: Feed $\hat{s}_{i-1}$ to the randomized full information algorithm FI and output σ_t according to FI.
15: **end for**
16: Set $\hat{R}_i \in \mathbb{R}^m$ as vector of relevances of the m items collected during exploration rounds.
17: Update $\hat{s}_i = \hat{s}_{i-1} + \hat{R}_i$.
18: **end for**

Initialization of parameters: In line 3 of the algorithm, the parameter specific to FTPL is randomization parameter $\epsilon \in \mathbb{R}$.

Exploitation round: σ_t, during exploitation, is sampled by FTPL as follows: sample $p_t \in [0, 1/\epsilon]^m$ from the product of uniform distribution in each dimension. Output permutation $\sigma_t = M(\hat{s}_{i-1} + p_t)$ where $M(y) = \operatorname{argmin}_{\sigma} \sigma^{-1} \cdot y$.

Discussion: The key reason for using FTPL as the full information algorithm is that the structure of our problem allows the permutation σ_t to be chosen during exploitation round via a simple sorting operation on m objects. This leads to an easily implementable algorithm which spends only $O(m \log m)$ time per round (sorting is in fact the most expensive step in the algorithm). The reason that the simple sorting operation does the trick is the following: FTPL only *implicitly* maintains a distribution over $m!$ actions (permutations) at beginning of each round. Instead of having an explicit probability distribution over each action and sampling from it, FTPL mimics sampling from a distribution over actions by randomly perturbing the information vector received so far (say $\hat{s}_{i-1}$ in block B_i) and then sorting the items by perturbed score. The random perturbation puts an implicit weight on each of the $m!$ actions and sorting is basically sampling according to the weights. This is an advantage over general full information algorithms based on exponential weights, which maintain explicit weight on actions and samples from it. We also note that we are using FTPL with uniform distribution over a hypercube as our perturbation distribution. Other choices, such as the Gaussian distribution, can lead to slightly better dependence on the dimension m (see the discussion following the proof of Corollary 9 in the appendix). Our bounds, therefore, do not necessarily yield the optimal dependence on m.

[1] For example, assume $m = 7$ and $k = 2$. Then place items $(1, 2)$ in cell 1, items $(3, 4)$ in cell 2, items $(5, 6)$ in cell 3 and item 7 in cell 4.

We have the following corollary:

Corollary 9. *The expected regret of SumLoss, obtained by applying Algorithm 1, with FTPL full information algorithm and feedback on top k ranked items at end of each round ($k \geq 1$), and $K = O\left(\frac{m^{1/3}T^{2/3}}{\lceil m/k \rceil^{2/3}}\right)$, $\epsilon = O(\frac{1}{\sqrt{mK}})$, is:*

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] - \min_{\sigma} \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) \leq O(m^{7/3}\lceil m/k \rceil^{1/3} T^{2/3}). \quad (6)$$

where $O(\cdot)$ hides some numeric constants.

Assuming that $\lceil m/k \rceil \sim m/k$, the regret rate in Corollary 9 is $O\left(\frac{m^{8/3}T^{2/3}}{k^{1/3}}\right)$

2.7 Regret Bounds for PairwiseLoss, DCG and Precision@n

PairwiseLoss: As we saw in Eq. 2, the regret of SumLoss is same as regret of PairwiseLoss. Thus, SumLoss in Corollary 9 can be replaced by PairwiseLoss to get exactly same result.

DCG: All the results of SumLoss can be extended to DCG (see Appendix A). Moreover, the results can be extended even for multi-graded relevance vectors. Thus, the minimax regret under DCG, *restricted to feedback on top ranked item*, even when the adversary can play multi-graded relevance vectors, is $\Theta(T^{2/3})$.

The main differences between SumLoss and DCG are the following. The former is a loss function; the latter is a gain function. Also, for DCG, $f(\sigma) \neq \sigma^{-1}$ (see definition in Sec.2.2) and when relevance is multi-graded, DCG cannot be expressed as $f(\sigma) \cdot R$, as clear from definition. Nevertheless, DCG can be expressed as $f(\sigma) \cdot g(R)$, , where $g(R) = [g^s(R(1)), g^s(R(2)), \ldots, g^s(R(m))]$, $g^s(i) = 2^i - 1$ is constructed from univariate, monotonic, scalar valued functions ($g(R) = R$ for binary graded relevance vectors). Thus, Algorithm 1 can be applied (with slight variation), with FTPL full information algorithm and top k feedback, to achieve regret of $O(T^{2/3})$ [2]. The slight variation is that during *exploration* rounds, when relevance feedback is collected to form the estimator at end of the block, the relevances should be transformed by function $g^s(\cdot)$.. The estimate is then constructed in the transformed space and fed to the full information algorithm. In the *exploitation* round, the selection of σ_t remains exactly same as in SumLoss, i.e., $\sigma_t = M(\hat{s}_{i-1} + p_t)$ where $M(y) = \operatorname{argmin}_{\sigma} \sigma^{-1} \cdot y$. This is because $\operatorname{argmax}_{\sigma} f(\sigma) \cdot y = \operatorname{argmin}_{\sigma} \sigma^{-1} \cdot y$, by definition of $f(\sigma)$ in DCG.

Let relevance vectors chosen by adversary have $n+1$ grades, i.e., $R \in \{0, 1, \ldots, n\}^m$. In practice, n is almost always less than 5. We have the following corollary:

Corollary 10. *The expected regret of DCG, obtained by applying Algorithm 1, with FTPL full information algorithm and feedback on top k ranked items at end of each round ($k \geq 1$),*

[2] Note that there is no problem in applying Lemma 7 to DCG. This is because we are trying to estimate $\sum_{n=1}^{t} \frac{g(R_n)}{t}$, which can be estimated by considering the relevance feedbacks themselves transformed by $g(\cdot)$

and $K = O\left(\frac{m^{1/3}T^{2/3}}{\lceil m/k \rceil^{2/3}}\right)$, $\epsilon = O(\frac{1}{(2^n-1)^2\sqrt{mK}})$, *is:*

$$\max_{\sigma} \sum_{t=1}^{T} DCG(\sigma_t, R_t) - \mathbb{E}\left[\sum_{t=1}^{T} DCG(\sigma_t, R_t)\right] \leq O((2^n-1)m^{4/3}\lceil m/k \rceil^{1/3}T^{2/3}). \quad (7)$$

Assuming that $\lceil m/k \rceil \sim m/k$, the regret rate in Corollary 10 is $O\left(\frac{(2^n-1)m^{5/3}T^{2/3}}{k^{1/3}}\right)$.

Precision@n: Since Precision@$n = f(\sigma) \cdot R$, the global observability property of SumLoss can be easily extended to it and Algorithm 1 can be applied, with FTPL full information algorithm and top k feedback, to achieve regret of $O(T^{2/3})$. In the *exploitation* round, the selection of σ_t remains exactly same as in SumLoss, i.e., $\sigma_t = M(\hat{s}_{i-1} + p_t)$ where $M(y) = \operatorname{argmin}_{\sigma} \sigma^{-1} \cdot y$.

However, the local observability property of SumLoss does not extend to Precision@n. The reason is that while $f(\cdot)$ of SumLoss is strictly monotonic, $f(\cdot)$ of Precision@n is monotonic but not strict. Precision@n depends only on the objects in the top n positions of the ranked list, *irrespective of the order*. A careful review shows that Lemma 3 fails to extend to the case of Precision@n, due to lack of strict monotonicity. Thus, we cannot define the neighboring action set of the Pareto optimal action pairs, and hence cannot prove or disprove local observability.

We have the following corollary:

Corollary 11. *The expected regret of Precision@n, obtained by applying Algorithm 1, with FTPL full information algorithm and feedback on top k ranked items at end of each round* ($k \geq 1$), *and* $K = O\left(\frac{m^{1/3}T^{2/3}}{\lceil m/k \rceil^{2/3}}\right)$, $\epsilon = O(\frac{1}{\sqrt{mK}})$, *is:*

$$\max_{\sigma} \sum_{t=1}^{T} Precision@n(\sigma_t, R_t) - \mathbb{E}\left[\sum_{t=1}^{T} Precision@n(\sigma_t, R_t)\right] \leq O(n\ m^{1/3}\lceil m/k \rceil^{1/3}T^{2/3}). \quad (8)$$

Assuming that $\lceil m/k \rceil \sim m/k$, the regret rate in Corollary 11 is $O\left(\frac{n\ m^{2/3}T^{2/3}}{k^{1/3}}\right)$.

2.8 Non-Existence of Sublinear Regret Bounds for NDCG, AP and AUC

As stated in Sec. 2.2, NDCG, AP and AUC are normalized versions of measures DCG, Precision@n and PairwiseLoss. We have the following lemma for all these normalized ranking measures.

Lemma 12. *The global observability condition, as per Definition 1, fails for NDCG, AP and AUC, when feedback is restricted to top ranked item.*

Combining the above lemma with Theorem 2 of Bartok et al. (2014), we conclude that there *cannot exist any algorithm which has sub-linear regret for any of the following measures: NDCG, AP or AUC, when restricted to top* 1 *feedback.*

Theorem 13. *There exists an online game, for NDCG with top-1 feedback, such that for every learner's algorithm, there is an adversary strategy generating relevance vectors, such that the expected regret of the learner is $\Omega(T)$. Furthermore, the same lower bound holds if NDCG is replaced by AP or AUC.*

3. Online Ranking with Restricted Feedback- Contextual Setting

All proofs not in the main text are in Appendix B.

3.1 Problem Setting and Learning to Rank Algorithm

First, we introduce some additional notations to Section 2.1. In the contextual setting, each query and associated items (documents) are represented jointly as a feature matrix. Each feature matrix, $X \in \mathbb{R}^{m \times d}$, consists of a list of m documents, each represented as a feature vector in $\mathbb{R}^d$. The feature matrices are considered side-information (context) and represents varying items, as opposed to the fixed set of items in the first part of our work. $X_{i:}$ denotes ith row of X. We assume feature vectors representing documents are bounded by R_D in ℓ_2 norm. The relevance vectors are same as before.

As per traditional learning to rank setting with query-document matrices, documents are ranked by a ranking function. The prevalent technique is to represent a ranking function as a scoring function and get ranking by sorting scores in descending order. A linear scoring function produces score vector as $f_w(X) = Xw = s^w \in \mathbb{R}^m$, with $w \in \mathbb{R}^d$. Here, $s^w(i)$ represents score of ith document (s^w points to score s being generated by using parameter w). We assume that ranking parameter space is bounded in ℓ_2 norm, i.e, $\|w\|_2 \leq U$, $\forall\ w$. $\pi_s = \text{argsort}(s)$ is the permutation induced by sorting score vector s in descending order. As a reminder, a permutation π gives a mapping from ranks to documents and π^{-1} gives a mapping from documents to ranks.

Performance of ranking functions are judged, based on the rankings obtained from score vectors, by ranking measures like DCG, AP and others. However, the measures themselves are discontinuous in the score vector produced by the ranking function, leading to intractable optimization problems. Thus, most learning to rank methods are based on minimizing *surrogate* losses, which can be optimized efficiently. A surrogate ϕ takes in a score vector s and relevance vector R and produces a real number, i.e., $\phi : \mathbb{R}^m \times \{0, 1, \ldots, n\}^m \mapsto \mathbb{R}$. $\phi(\cdot, \cdot)$ is said to be convex if it is convex in its first argument, for any value of the second argument. Ranking surrogates are designed in such a way that the ranking function learnt by optimizing the surrogates has good performance with respect to ranking measures.

Formal problem setting: We formalize the problem as a game being played between a learner and an adversary over T rounds. The learner's action set is the uncountably infinite set of score vectors in $\mathbb{R}^m$ and the adversary's action set is all possible relevance vectors, i.e., $(n+1)^m$ possible vectors. At round t, the adversary generates a list of documents, represented by a matrix $X_t \in \mathbb{R}^{m \times d}$, pertaining to a query (the document list is considered as side information). The learner receives X_t, produces a score vector $\tilde{s}_t \in \mathbb{R}^m$ and ranks the documents by sorting according to score vector. The adversary then generates a relevance vector R_t but only reveals the relevances of top k ranked documents to the learner. The learner uses the feedback to choose its action for the next round (updates an internal scoring function). The learner suffers a loss as measured in terms of a surrogate ϕ, i.e, $\phi(\tilde{s}_t, R_t)$.

As is standard in online learning setting, the learner's performance is measured in terms of its expected regret:

$$\mathbb{E}\left[\sum_{t=1}^{T}\phi(\tilde{s}_t, R_t)\right] - \min_{\|w\|_2\leq U}\sum_{t=1}^{T}\phi(X_t w, R_t),$$

where the expectation is taken w.r.t. to randomization of learner's strategy and $X_t w = s_t^w$ is the score produced by the linear function parameterized by w.

Algorithm 2 Ranking with Top-k Feedback (RTop-kF)- Contextual

1: Exploration parameter $\gamma \in (0, \frac{1}{2})$, learning parameter $\eta > 0$, ranking parameter $w_1 = \mathbf{0} \in \mathbb{R}^d$
2: **For** $t = 1$ to T
3: Receive X_t (document list pertaining to query q_t)
4: Construct score vector $s_t^{w_t} = X_t w_t$ and get permutation $\sigma_t = \text{argsort}(s_t^{w_t})$
5: $\mathbb{Q}_t(s) = (1-\gamma)\delta(s - s_t^{w_t}) + \gamma\text{Uniform}([0,1]^m)$ (δ is the Dirac Delta function).
6: Sample $\tilde{s}_t \sim \mathbb{Q}_t$ and output the ranked list $\tilde{\sigma}_t = \text{argsort}(\tilde{s}_t)$
(Effectively[3], it means $\tilde{\sigma}_t$ is drawn from $\mathbb{P}_t(\sigma) = (1-\gamma)\mathbb{1}(\sigma = \sigma_t) + \frac{\gamma}{m!}$)
7: Receive relevance feedback on top-k items, i.e., $(R_t(\tilde{\sigma}_t(1)), \ldots, R_t(\tilde{\sigma}_t(k)))$
8: Suffer loss $\phi(\tilde{s}_t, R_t)$ (Neither loss nor R_t revealed to learner)
9: Construct $\tilde{z}_t$, an unbiased estimator of gradient $\nabla_{w=w_t}\phi(X_t w, R_t)$, from top-$k$ feedback.
10: Update $w = w_t - \eta\tilde{z}_t$
11: $w_{t+1} = \min\{1, \frac{\text{U}}{\|w\|_2}\}w$ (Projection onto Euclidean ball of radius U).
12: **End For**

Relation between feedback and structure of surrogates: Algorithm 2 is our general algorithm for learning a ranking function, online, from partial feedback. The key step in Algorithm 2 is the construction of the unbiased estimator $\tilde{z}_t$ of the surrogate gradient $\nabla_{w=w_t}\phi(X_t w, R_t)$. The information present for the construction process, at end of round t, is the random score vector $\tilde{s}_t$ (and associated permutation $\tilde{\sigma}_t$) and relevance of top-k items of $\tilde{\sigma}_t$, i.e., $\{R_t(\tilde{\sigma}_t(1)), \ldots, R_t(\tilde{\sigma}_t(k)\}$. Let $\mathbb{E}_t[\cdot]$ be the expectation operator w.r.t. to randomization at round t, conditioned on $(w_1, \ldots, w_t)$. Then $\tilde{z}_t$ being an unbiased estimator of gradient of surrogate, w.r.t w_t, means the following: $\mathbb{E}_t[\tilde{z}_t] = \nabla_{w=w_t}\phi(X_t w, R_t)$. We note that conditioned on the past, the score vector $s_t^{w_t} = X_t w_t$ is deterministic. We start with a general result relating feedback to the construction of unbiased estimator of a vector valued function. Let S_m be the set of $m!$ permutations of $[m]$. Let $\mathbb{P}$ denote a probability distribution on S_m, i.e, $\sum_{\sigma\in S_m}\mathbb{P}(\sigma) = 1$. For a distinct set of indices $(j_1, j_2, \ldots, j_k) \subseteq [m]$, we denote $p(j_i, j_2, \ldots, j_k)$ as the the sum of probability of permutations whose first k objects match objects $(j_1, \ldots, j_k)$, in order. Formally,

$$p(j_1, \ldots, j_k) = \sum_{\pi\in S_m}\mathbb{P}(\pi)\mathbb{1}(\pi(1) = j_1, \ldots, \pi(k) = j_k). \tag{9}$$

We have the following lemma relating feedback and structure of surrogates:

[3] $\tilde{s}_t \sim s_t^{w_t} \implies \tilde{\sigma}_t \sim \sigma_t$; otherwise $\tilde{\sigma}_t$ is any other permutation.

Lemma 14. *Let $F : \mathbb{R}^m \mapsto \mathbb{R}^a$ be a vector valued function, where $m \geq 1$, $a \geq 1$. For a fixed $x \in \mathbb{R}^m$, let k entries of x be observed at random. That is, for a fixed probability distribution $\mathbb{P}$ and some random $\sigma \sim \mathbb{P}(S_m)$, observed tuple is $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$. A necessary condition for existence of an unbiased estimator of $F(x)$, that can be constructed from $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$, is that it should be possible to decompose $F(x)$ over k (or less) coordinates of x at a time. That is, $F(x)$ should have the structure:*

$$F(x) = \sum_{(i_1, i_2, \ldots, i_\ell) \in {}^mP_\ell} h_{i_1, i_2, \ldots, i_\ell}(x_{i_1}, x_{i_2}, \ldots, x_{i_\ell}) \tag{10}$$

where $\ell \leq k$, ${}^mP_\ell$ is ℓ permutations of m and $h : \mathbb{R}^\ell \mapsto \mathbb{R}^a$ (the subscripts in h are used to denote possibly different functions in the decomposition structure). Moreover, when $F(x)$ can be written in form of Eq 10 , with $\ell = k$, an unbiased estimator of $F(x)$, based on $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$, is,

$$g(\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}) = \frac{\sum_{(j_1, j_2, \ldots, j_k) \in S_k} h_{\sigma(j_1), \ldots, \sigma(j_k)}(x_{\sigma(j_1)}, \ldots, x_{\sigma(j_k)})}{\sum_{(j_1, \ldots, j_k) \in S_k} p(\sigma(j_1), \ldots, \sigma(j_k))} \tag{11}$$

where S_k is the set of $k!$ permutations of $[k]$ and $p(\sigma(1), \ldots, \sigma(k))$ is as in Eq 9 .

Illustrative Examples: We provide simple examples to concretely illustrate the abstract functions in Lemma 14. Let $F(\cdot)$ be the identity function, and $x \in \mathbb{R}^m$. Thus, $F(x) = x$ and the function decomposes over $k = 1$ coordinate of x as follows: $F(x) = \sum_{i=1}^m x_i e_i$, where $e_i \in \mathbb{R}^m$ is the standard basis vector along coordinate i. Hence, $h_i(x_i) = x_i e_i$. Based on top-1 feedback, following is an unbiased estimator of $F(x)$: $g(\sigma, x_{\sigma(1)}) = \frac{x_{\sigma(1)} e_{\sigma(1)}}{p(\sigma(1))}$, where $p(\sigma(1)) = \sum_{\pi \in S_m} \mathbb{P}(\pi) \mathbb{1}(\pi(1) = \sigma(1))$. In another example, let $F : \mathbb{R}^3 \mapsto \mathbb{R}^2$ and $x \in \mathbb{R}^3$. Let $F(x) = [x_1 + x_2; x_2 + x_3]^\top$. Then the function decomposes over $k = 1$ coordinate of x as $F(x) = x_1 e_1 + x_2(e_1 + e_2) + x_3 e_2$, where $e_i \in \mathbb{R}^2$. Hence, $h_1(x_1) = x_1 e_1$, $h_2(x_2) = x_2(e_1 + e_2)$ and $h_3(x_3) = x_3 e_2$. An unbiased estimator based on top-1 feedback is: $g(\sigma, x_{\sigma(1)}) = \frac{h_{\sigma(1)}(x_{\sigma(1)})}{p(\sigma(1))}$.

3.2 Unbiased Estimators of Gradients of Surrogates

Algorithm 2 can be implemented for any ranking surrogate as long as an unbiased estimator of the gradient can be constructed from the random feedback. We will use techniques from *online convex optimization* to obtain formal regret guarantees. We will thus construct the unbiased estimator of four major ranking surrogates. Three of them are popular *convex* surrogates, one each from the three major learning to rank methods, i.e., *pointwise*, *pairwise* and *listwise* methods. The fourth one is a popular *non-convex* surrogate.

Shorthand notations: We note that by chain rule, $\nabla_{w=w_t} \phi(X_t w, R_t) = X_t^\top \nabla_{s_t^{w_t}} \phi(s_t^{w_t}, R_t)$, where $s_t^{w_t} = X_t w_t$. Since X_t is deterministic in our setting, we focus on unbiased estimators of $\nabla_{s_t^{w_t}} \phi(s_t^{w_t}, R_t)$ and take a matrix-vector product with X_t. To reduce notational clutter

in our derivations, we drop w from s^w and the subscript t throughout. Thus, in our derivations, $\tilde{z} = \tilde{z}_t$, $X = X_t$, $s = s_t^{w_t}$ (and not $\tilde{s}_t$), $\sigma = \tilde{\sigma}_t$ (and not σ_t), $R = R_t$, e_i is standard basis vector in $\mathbb{R}^m$ along coordinate i and $p(\cdot)$ as in Eq. 9 with $\mathbb{P} = \mathbb{P}_t$ where $\mathbb{P}_t$ is the distribution in round t in Algorithm 2.

3.2.1 Convex Surrogates

Pointwise Method: We will construct the unbiased estimator of the gradient of squared loss (Cossock and Zhang, 2006): $\phi_{sq}(s, R) = \|s - R\|_2^2$. The gradient $\nabla_s \phi_{sq}(s, R)$ is $2(s - R) \in \mathbb{R}^m$. As we have already demonstrated in the example following Lemma 14, *we can construct unbiased estimator of R from top-1 feedback* ($\{\sigma, R(\sigma(1))\}$). Concretely, the unbiased estimator is:

$$\tilde{z} = X^\top \left(2 \left(s - \frac{R(\sigma(1)) e_{\sigma(1)}}{p(\sigma(1))} \right) \right).$$

Pairwise Method: We will construct the unbiased estimator of the gradient of hinge-like surrogate in RankSVM (Joachims, 2002): $\phi_{svm}(s, R) = \sum_{i \neq j=1} \mathbb{1}(R(i) > R(j)) \max(0, 1 + s(j) - s(i))$. The gradient is given by:

$$\nabla_s \phi_{svm}(s, R) = \sum_{i=1}^{m} \sum_{j=1, j \neq i}^{m} \mathbb{1}(R(i) > R(j)) \mathbb{1}(1 + s(j) > s(i))(e_j - e_i) \in \mathbb{R}^m.$$

Since s is a known quantity, from Lemma 14, we can construct $F(R)$ as follows:

$$F(R) = F_s(R) = \sum_{i=1}^{m} \sum_{j=1, j \neq i}^{m} h_{s,i,j}(R(i), R(j)), \;\; h_{s,i,j}(R(i), R(j)) = \mathbb{1}(R(i) > R(j)) \mathbb{1}(1 + s(j) > s(i))(e_j - e_i).$$

Since $F_s(R)$ is decomposable over 2 coordinates of R at a time, *we can construct an unbiased estimator from top-2 feedback* ($\{\sigma, R(\sigma(1)), R(\sigma(2))\}$). The unbiased estimator is:

$$\tilde{z} = X^\top \left(\frac{h_{s,\sigma(1),\sigma(2)}(R(\sigma(1)), R(\sigma(2))) + h_{s,\sigma(2),\sigma(1)}(R(\sigma(2)), R(\sigma(1)))}{p(\sigma(1), \sigma(2)) + p(\sigma(2), \sigma(1))} \right).$$

We note that the unbiased estimator was constructed from top-2 feedback. The following lemma, in conjunction with the necessary condition of Lemma 14 shows that it is the minimum information required to construct the unbiased estimator.

Lemma 15. *The gradient of RankSVM surrogate, i.e., $\phi_{svm}(s, R)$ cannot be decomposed over 1 coordinate of R at a time.*

Listwise Method: Convex surrogates developed for listwise methods of learning to rank are defined over the entire score vector and relevance vector. Gradients of such surrogates cannot usually be decomposed over coordinates of the relevance vector. We will focus on the cross-entropy surrogate used in the highly cited ListNet (Cao et al., 2007) ranking algorithm and show how a very natural modification to the surrogate makes its gradient estimable in our partial feedback setting.

The authors of the ListNet method use a cross-entropy surrogate on two probability distributions on permutations, induced by score and relevance vector respectively. More

formally, the surrogate is defined as follows[4]. Define m maps from $\mathbb{R}^m$ to $\mathbb{R}$ as: $P_j(v) = \exp(v(j))/\sum_{j=1}^m \exp(v(j))$ for $j \in [m]$. Then, for score vector s and relevance vector R, $\phi_{\text{LN}}(s,R) = -\sum_{i=1}^m P_i(R)\log P_i(s)$ and $\nabla_s \phi_{\text{LN}}(s,R) = \sum_{i=1}^m \left(-\frac{\exp(R(i))}{\sum_{j=1}^m \exp(R(j))} + \frac{\exp(s(i))}{\sum_{j=1}^m \exp(s(j))}\right) e_i$. We have the following lemma about the gradient of ϕ_{LN}.

Lemma 16. *The gradient of ListNet surrogate $\phi_{LN}(s,R)$ cannot be decomposed over k, for $k = 1, 2$, coordinates of R at a time.*

In fact, an examination of the proof of the above lemma reveals that decomposability at any $k < m$ does not hold for the gradient of LisNet surrogate, though we only prove it for $k = 1, 2$ (since feedback for top k items with $k > 2$ does not seem practical). Due to Lemma 14, this means that if we want to run Alg. 2 under top-k feedback, a modification of ListNet is needed. We now make such a modification.

We first note that the cross-entropy surrogate of ListNet can be easily obtained from a standard divergence, viz. Kullback-Liebler divergence. Let $p, q \in \mathbb{R}^m$ be 2 probability distributions ($\sum_{i=1}^m p_i = \sum_{i=1}^m q_i = 1$). Then $KL(p,q) = \sum_{i=1}^m p_i \log(p_i) - \sum_{i=1}^m p_i \log(q_i) - \sum_{i=1}^m p_i + \sum_{i=1}^m q_i$. Taking $p_i = P_i(R)$ and $q_i = P_i(s)$, $\forall\ i \in [m]$ (where $P_i(v)$ is as defined in ϕ_{LN}) and noting that $\phi_{\text{LN}}(s,R)$ needs to be minimized w.r.t. s (thus we can ignore the $\sum_{i=1}^m p_i \log(p_i)$ term in $KL(p,q)$), we get the cross entropy surrogate from KL.

Our natural modification now easily follows by considering KL divergence for *un-normalized* vectors (it should be noted that KL divergence is an instance of a Bregman divergence). Define m maps from $\mathbb{R}^m$ to $\mathbb{R}$ as: $P'_j(v) = \exp(v(j))$ for $j \in [m]$. Now define $p_i = P'_i(R)$ and $q_i = P'_i(s)$. Then, the modified surrogate $\phi_{KL}(s,R)$ is:

$$\sum_{i=1}^m e^{R(i)} \log(e^{R(i)}) - \sum_{i=1}^m e^{R(i)} \log(e^{s(i)}) - \sum_{i=1}^m e^{R(i)} + \sum_{i=1}^m e^{s(i)},$$

and $\sum_{i=1}^m (\exp(s(i)) - \exp(R(i)))\, e_i$ is its gradient w.r.t. s. Note that $\phi_{KL}(s,R)$ is non-negative and convex in s. Equating gradient to $\mathbf{0} \in \mathbb{R}^m$, at the minimum point, $s(i) = R(i)$, $\forall\ i \in [m]$. Thus, the sorted order of optimal score vector agrees with sorted order of relevance vector and it is a valid ranking surrogate.

Now, from Lemma 14, we can construct $F(R)$ as follows: $F(R) = F_s(R) = \sum_{i=1}^m h_{s,i}(R(i))$, where $h_{s,i}(R(i)) = (\exp(s(i)) - \exp(R(i)))\, e_i$. Since $F_s(R)$ is decomposable over 1 coordinate of R at a time, *we can construct an unbiased estimator from top-1 feedback* ($\{\sigma, R(\sigma(1))\}$). The unbiased estimator is:

$$\tilde{\mathbf{z}} = X^\top \left(\frac{(\exp(s(\sigma(1))) - \exp(R(\sigma(1))))e_{\sigma(1)}}{p(\sigma(1))} \right)$$

Other Listwise Methods: As we mentioned before, most listwise convex surrogates will not be suitable for Algorithm 2 with top-k feedback. For example, the class of popular listwise surrogates that are developed from structured prediction perspective (Chapelle

[4]The ListNet paper actually defines a family of losses based on probability models for top r documents, with $r \leq m$. We use $r = 1$ in our definition since that is the version implemented in their experimental results.

et al., 2007; Yue et al., 2007) cannot have unbiased estimator of gradients from top-k feedback since they are based on maps from full relevance vectors to full rankings and thus cannot be decomposed over $k = 1$ or 2 coordinates of R. It does not appear they have any natural modification to make them amenable to our approach.

3.2.2 Non-convex Surrogate

We provide an example of a non-convex surrogate for which Alg. 2 is applicable (however it will not have any regret guarantees due to non-convexity). We choose the SmoothDCG surrogate given in (Chapelle and Wu, 2010), which has been shown to have very competitive empirical performance. SmoothDCG, like ListNet, defines a family of surrogates, based on the cut-off point of DCG (see original paper (Chapelle and Wu, 2010) for details). We consider SmoothDCG@1, which is the smooth version of DCG@1 (i.e., DCG which focuses just on the top-ranked document). The surrogate is defined as: $\phi_{SD}(s, R) = \frac{1}{\sum_{j=1}^{m} \exp(s(j)/\epsilon)} \sum_{i=1}^{m} G(R(i)) \exp(s(i)/\epsilon)$, where ϵ is a (known) smoothing parameter and $G(a) = 2^a - 1$. The gradient of the surrogate is:

$$[\nabla_s \phi_{\mathrm{SD}}(s, R)] = \sum_{i=1}^{m} h_{s,i}(R(i)),$$

$$h_{s,i}(R(i)) = G(R_i) \left(\sum_{j=1}^{m} \frac{1}{\epsilon} [\frac{\exp(s(i)/\epsilon)}{\sum_{j'} \exp(s(j')/\epsilon)} \mathbb{1}_{(i=j)} - \frac{\exp((s(i) + s(j))/\epsilon)}{(\sum_{j'} \exp(s(j')/\epsilon))^2}] e_j \right)$$

Using Lemma 14, we can write $F(R) = F_s(R) = \sum_{i=1}^{m} h_{s,i}(R(i))$ where $h_{s,i}(R(i))$ is defined above. Since $F_s(R)$ is decomposable over 1 coordinate of R at a time, *we can construct an unbiased estimator from top-1 feedback* ($\{\sigma, R(\sigma(1))\}$), with unbiased estimator being:

$$s(\sigma(1))\tilde{z} = X^\top \left(\frac{G(R(\sigma(1)))}{p(\sigma(1))} \sum_{j=1}^{m} \frac{1}{\epsilon} [\frac{\exp(s(\sigma(1))/\epsilon)}{\sum_{j'} \exp(s(j')/\epsilon)} \mathbb{1}_{(\sigma(1)=j)} - \frac{\exp((s(\sigma(1)) + s(j))/\epsilon)}{(\sum_{j'} \exp(s(j')/\epsilon))^2}] e_j \right)$$

3.3 Computational Complexity of Algorithm 2

Three of the four key steps governing the complexity of Algorithm 2, i.e., construction of $\tilde{s}_t$, $\tilde{\sigma}_t$ and sorting can all be done in $O(m \log(m))$ time.The only bottleneck could have been calculations of $p(\tilde{\sigma}_t(1))$ in squared loss, (modified) ListNet loss and SmoothDCG loss, and $p(\tilde{\sigma}_t(1), \tilde{\sigma}_t(2))$ in RankSVM loss, since they involve sum over permutations. However, they have a compact representation, i.e., $p(\tilde{\sigma}_t(1)) = (1 - \gamma + \frac{\gamma}{m}) \mathbb{1}(\tilde{\sigma}_t(1) = \sigma_t(1)) + \frac{\gamma}{m} \mathbb{1}(\tilde{\sigma}_t(1) \neq \sigma_t(1))$ and $p(\tilde{\sigma}_t(1), \tilde{\sigma}_t(2)) = (1 - \gamma + \frac{\gamma}{m(m-1)}) \mathbb{1}(\tilde{\sigma}_t(1) = \sigma_t(1), \tilde{\sigma}_t(2) = \sigma_t(2)) + \frac{\gamma}{m(m-1)} [\sim \mathbb{1}(\tilde{\sigma}_t(1) = \sigma_t(1), \tilde{\sigma}_t(2) = \sigma_t(2))]$. The calculations follow easily due to the nature of $\mathbb{P}_t$ (step-6 in algorithm) which put equal weights on all permutations other than σ_t.

3.4 Regret Bounds

The underlying deterministic part of our algorithm is online gradient descent (OGD) (Zinkevich, 2003). The regret of OGD, run with unbiased estimator of gradient of a **convex**

function, as given in Theorem 3.1 of (Flaxman et al., 2005), in our problem setting is:

$$\mathbb{E}\left[\sum_{t=1}^{T}\phi(X_t w_t, R_t)\right] \leq \min_{w:\|w\|_2\leq U}\sum_{t=1}^{T}\phi(X_t w, R_t) + \frac{U^2}{2\eta} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\|\tilde{z}_t\|_2^2\right] \tag{12}$$

where $\tilde{z}_t$ is unbiased estimator of $\nabla_{w=w_t}\phi(X_t w, R_t)$, conditioned on past events, η is the learning rate and the expectation is taken over all randomness in the algorithm.

However, from the perspective of the loss $\phi(\tilde{s}_t, R_t)$ incurred by Algorithm 2, at each round t, the RHS above is not a valid upper bound. The algorithms plays the score vector suggested by OGD ($\tilde{s}_t = X_t w_t$) with probability $1-\gamma$ (exploitation) and plays a randomly selected score vector (i.e., a draw from the uniform distribution on $[0,1]^m$), with probability γ (exploration). Thus, the expected number of rounds in which the algorithm does not follow the score suggested by OGD is γT, leading to an extra regret of order γT. Thus, we have [5]

$$\mathbb{E}\left[\sum_{t=1}^{T}\phi(\tilde{s}_t, R_t)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T}\phi(X_t w_t, R_t)\right] + O\left(\gamma T\right) \tag{13}$$

We first control $\mathbb{E}_t\|\tilde{z}_t\|_2^2$, for all convex surrogates considered in our problem (we remind that $\tilde{z}_t$ is the estimator of a gradient of a surrogate, calculated at time t. In Sec 3.2.1 , we omitted showing w in s^w and index t). To get bound on $\mathbb{E}_t\|\tilde{z}_t\|_2^2$, we used the following norm relation that holds for any matrix X (Bhaskara and Vijayaraghavan, 2011): $\|X\|_{p\to q} = \sup_{v\neq 0}\frac{\|Xv\|_q}{\|v\|_p}$, where q is the dual exponent of p (i.e., $\frac{1}{q}+\frac{1}{p}=1$), and the following lemma derived from it:

Lemma 17. *For any $1 \leq p \leq \infty$, $\|X^\top\|_{1\to p} = \|X\|_{q\to\infty} = \max_{j=1}^{m}\|X_{j:}\|_p$, where $X_{j:}$ denotes jth row of X and m is the number of rows of matrix.*

We have the following result:

Lemma 18. *For parameter γ in Algorithm 2 , R_D being the bound on ℓ_2 norm of the feature vectors (rows of document matrix X), m being the upper bound on number of documents per query, U being the radius of the Euclidean ball denoting the space of ranking parameters and $R_{\max}$ being the maximum possible relevance value (in practice always ≤ 5), let $C^\phi \in \{C^{sq}, C^{svm}, C^{KL}\}$ be polynomial functions of R_D, m, U, R_{max}, where the degrees of the polynomials depend on the surrogate ($\phi_{sq}, \phi_{svm}, \phi_{KL}$), with no degree ever greater than four. Then we have,*

$$\mathbb{E}_t\left[\|\tilde{z}_t\|_2^2\right] \leq \frac{C^\phi}{\gamma} \tag{14}$$

Plugging Eq. 14 and Eq. 13 in Eq. 12, and optimizing over η and γ, (which gives $\eta = O(T^{-2/3})$ and $\gamma = O(T^{-1/3})$), we get the final regret bound:

[5] The instantaneous loss suffered at each of the exploration round can be maximum of $O(1)$, as long as $\phi(s, R)$ is bounded, $\forall\ s$ and $\forall\ R$. This is true because the score space is ℓ_2 norm bounded, maximum relevance grade is finite in practice and we consider Lipschitz, convex surrogates.

Theorem 19. *For any sequence of instances and labels $(X_t, R_t)_{\{t\in[T]\}}$, applying Algorithm 2 with top-1 feedback for ϕ_{sq} and ϕ_{KL} and top-2 feedback for ϕ_{svm}, will produce the following bound:*

$$\mathbb{E}\left[\sum_{t=1}^{T}\phi(\tilde{s}_t, R_t)\right] - \min_{w:\|w\|_2\leq U}\sum_{t=1}^{T}\phi(X_t w, R_t) \leq C^{\phi} O\left(T^{2/3}\right) \tag{15}$$

where C^{ϕ} is a surrogate dependent function, as described in Lemma 18 , and expectation is taken over underlying randomness of the algorithm, over T rounds.

3.5 Impossibility of Sublinear Regret for NDCG Calibrated Surrogates

Learning to rank methods optimize surrogates to learn a ranking function, even though performance is measured by target measures like NDCG. This is done because direct optimization of the measures lead to NP-hard optimization problems. One of the most desirable properties of any surrogate is *calibration*, i.e., the surrogate should be calibrated w.r.t the target (Bartlett et al., 2006). Intuitively, it means that a function with small expected surrogate loss on unseen data should have small expected target loss on unseen data. We focus on NDCG calibrated surrogates (both convex and non-convex) that have been characterized by Ravikumar et al. (2011). We first state the necessary and sufficient condition for a surrogate to be calibrated w.r.t NDCG. For any score vector s and distribution η on relevance space $\mathcal{Y}$, let $\bar{\phi}(s,\eta) = \mathbb{E}_{R\sim\eta}\phi(s,R)$. Moreover, we define $G(\mathbf{R}) = (G(R_1),\ldots,G(R_m))^{\top}$. $Z(R)$ is defined in Sec 2.2.

Theorem 20. *(Ravikumar et al., 2011, Thm. 6) A surrogate ϕ is NDCG calibrated iff for any distribution η on relevance space $\mathcal{Y}$, there exists an invertible, order preserving map $g:\mathbb{R}^m \mapsto \mathbb{R}^m$ s.t. the unique minimizer $s^*_{\phi}(\eta)$ can be written as*

$$s^*_{\phi}(\eta) = g\left(\mathbb{E}_{R\sim\eta}\left[\frac{G(\mathbf{R})}{Z(R)}\right]\right). \tag{16}$$

Informally, Eq. 16 states that $\text{argsort}(s^*_{\phi}(\eta)) \subseteq \text{argsort}(\mathbb{E}_{R\sim\eta}\left[\frac{G(\mathbf{R})}{Z(R)}\right])$. Ravikumar et al. (2011) give concrete examples of NDCG calibrated surrogates, including how some of the popular surrogates can be converted into NDCG calibrated ones: e.g., the NDCG calibrated version of squared loss is $\|s - \frac{G(\mathbf{R})}{Z(R)}\|_2^2$.

We now state the *impossibility* result for the class of NDCG calibrated surrogates when feedback is restricted to top ranked item.

Theorem 21. *Fix the online learning to rank game with top 1 feedback and any NDCG calibrated surrogate. Then, for every learner's algorithm, there exists an adversary strategy such that the learner's expected regret is $\Omega(T)$.*

We note that the proof of Theorem. 3 of Piccolboni and Schindelhauer (2001) cannot be directly extended to prove the impossibility result because it relies on constructing a connected graph on vertices defined by neighboring actions of learner. In our case, due to the continuous nature of learner's actions, the graph will be an empty graph and proof will break down.

4. Experiments

We conducted experiments on simulated and commercial data sets to demonstrate the performance of our algorithms.

4.1 Non Contextual Setting

Objectives: We had the following objectives while conducting experiments in the non-contextual, online ranking with partial feedback setting:

- Investigate how performance of Algorithm 1 is affected by size of blocks during blocking.
- Investigate how performance of the algorithm is affected by amount of feedback received (i.e., generalizing k in top k feedback).
- Demonstrate the difference between regret rate of our algorithm, which operates in partial feedback setting, with regret rate of a full information algorithm which receives full relevance vector feedback at end of each round.

We applied Algorithm 1 in conjunction with Follow-The-Perturbed-Leader (FTPL) full information algorithm, as described in Sec. 2.6. Note that the online learning to rank problem is not as well studied as the batch problem. This is especially true for the partial feedback setting we have. Therefore benchmark data sets with published baseline performance measures, are not available for comparison, to the best of our knowledge. The generic partial monitoring algorithms that do exist cannot be applied due to computational inefficiency (Sec. 2.5).

Experimental Setting: All our experiments were conducted with respect to the DCG measure, which is quite popular in practice, and binary graded relevance vectors. Our experiments were conducted on the following simulated data set. We fixed number of items to 20 ($m = 20$). We then fixed a "true" relevance vector which had 5 items with relevance level 1 and 15 items with relevance level 0. We then created a total of T=10000 relevance vectors by corrupting the true relevance vector. The corrupted copies were created by independently flipping each relevance level (0 to 1 and vice-versa) with a small probability. The reason for creating adversarial relevance vectors in such a way was to reflect diversity of preferences in practice. In reality, it is likely that most users will have certain similarity in preferences, with small deviations on certain items and certain users. That is, some items are likely to be relevant in general, with most items not relevant to majority of users, with slight deviation from user to user. Moreover, the comparator term in our regret bound (i.e., cumulative loss/gain of the true best ranking in hindsight) makes practical sense if there is a ranking which satisfies most users.

Results: We plotted average regret over time under DCG. Average regret over time means cumulative regret up to time t, divided by t, for $1 \leq t \leq T$. Figure 1 demonstrates the effect of block size on the regret rate under DCG. We fixed feedback to relevance of top ranked item ($k = 1$). As can be seen in Corollary 10, the optimal regret rate is achieved by optimizing over number of blocks (hence block size), which requires prior knowledge of

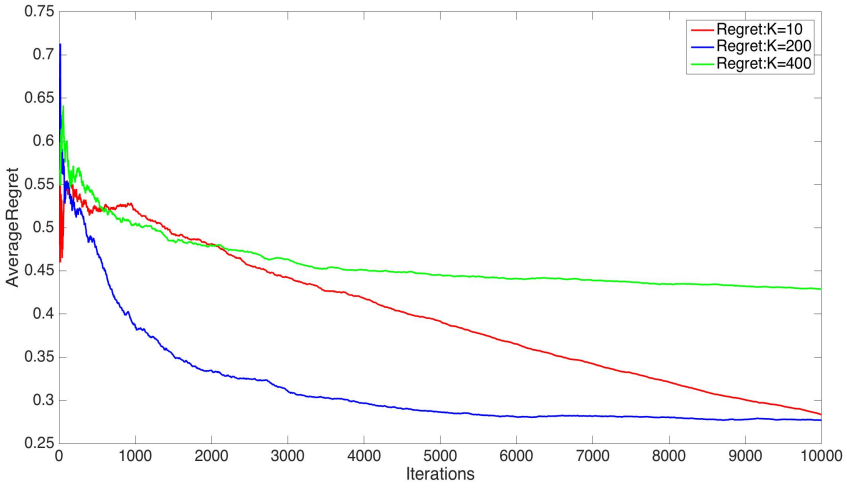

Figure 1: Average regret under DCG, with feedback on top ranked object, for varying block size, where block size is $\lceil T/K \rceil$. *Best viewed in color.*

time horizon T. We wanted to demonstrate how the regret rate (and hence the performance of the algorithm) differs with different block sizes. The optimal number of blocks in our setting is $K \sim 200$, with corresponding block size being $\lceil T/K \rceil = 50$. As can be clearly seen, with optimal block size, the regret drops fastest and becomes steady after a point. $K = 10$ means that block size is 1000. This means over the time horizon, number of exploitation rounds greatly dominates number of exploration rounds, leading to regret dropping at a slower rate initially than than the case with optimal block size. However, the regret drops of pretty sharply later on. This is because the relevance vectors are slightly corrupted copies of a "true" relevance vector and the algorithm gets a good estimate of the true relevance vector quickly and then more exploitation helps. When $K = 400$ (i.e., block size is 25), most of the time, the algorithm is exploring, leading to a substantially worse regret and poor performance.

Figure 2 demonstrates the effect of amount of feedback on the regret rate under DCG. We fixed $K = 200$, and varied feedback as relevance of top k ranked items per round, where $k = 1, 5, 10$. Validating our regret bound, we see that as k increases, the regret decreases.

Figure 3 compares regret of our algorithm, working with top 1 feedback, and FTPL full information algorithm, working with full relevance vector feedback at end of each round. We fixed $K = 200$ and the comparison was done from 1000 iterations onwards, i.e., roughly after the initial learning phase. FTPL full information algorithm has regret rate of $O(T^{1/2})$ (ignoring other parameters). So, as expected, FTPL with full information feedback outperforms our algorithm with highly restricted feedback; yet, we have demonstrated, both theoretically and empirically, that it is possible to have a good ranking strategy with highly restricted feedback.

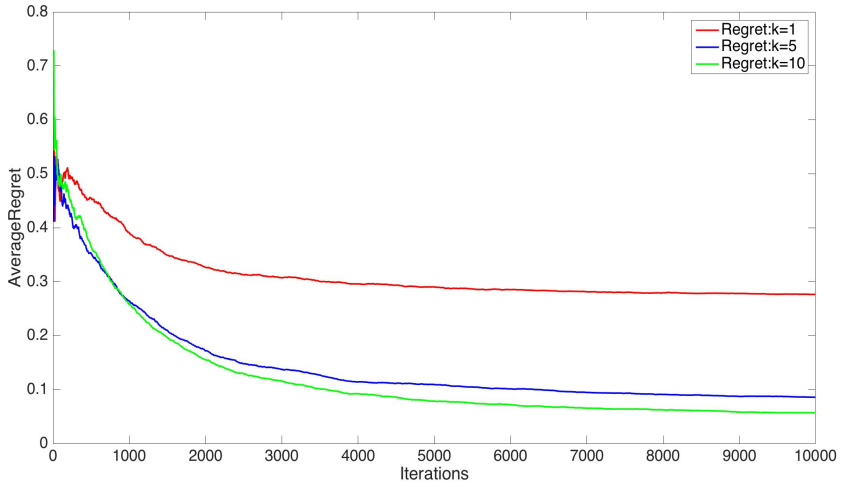

Figure 2: Average regret under DCG, where $K = 200$, for varying amount of feedback. *Best viewed in color.*

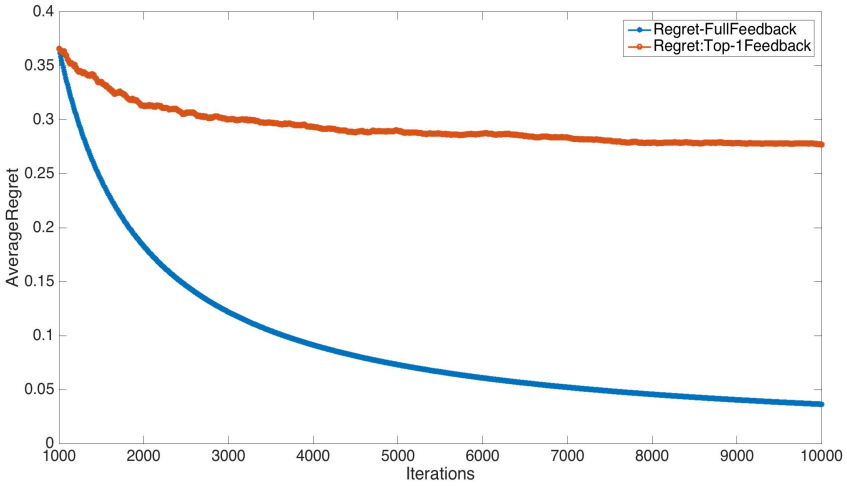

Figure 3: Comparison of average regret over time, for DCG, between top-1 feedback and full relevance vector feedback. *Best viewed in color.*

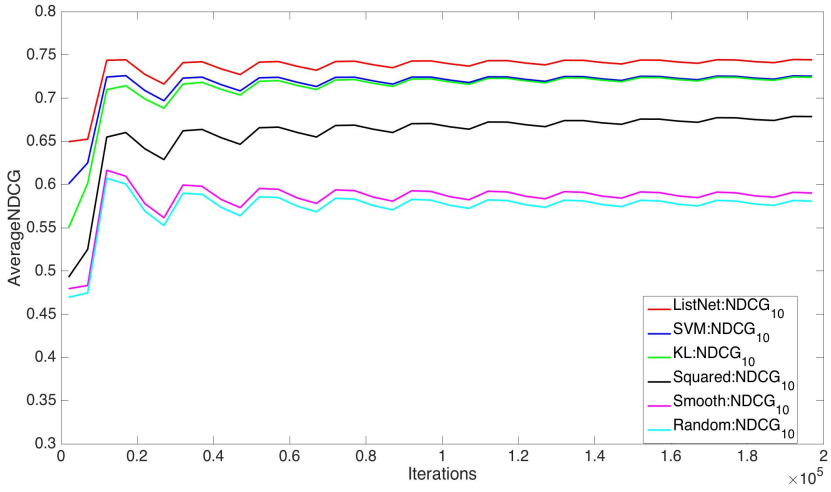

Figure 4: Average NDCG_{10} values of different algorithms, for Yahoo data set. ListNet:NDCG_{10} (in cyan) operates on full feedback and Random:NDCG_{10} (in red) does not receive any feedback. *Best viewed in color* .

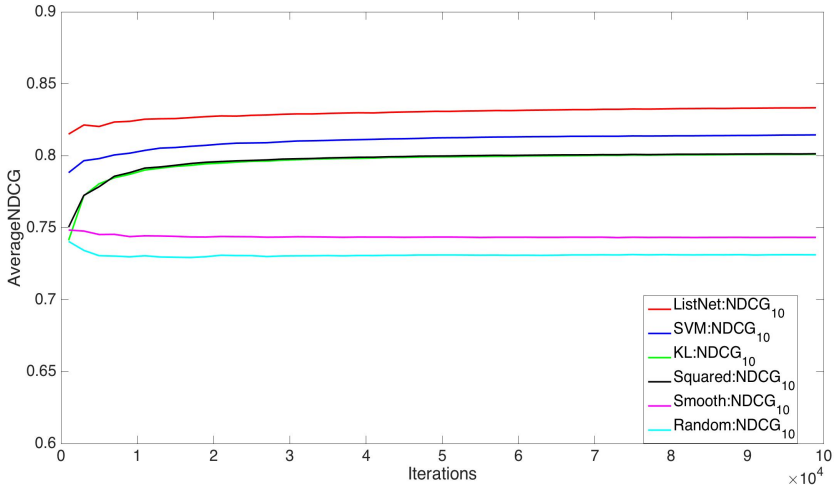

Figure 5: Average NDCG_{10} values of different algorithms, for Yandex data set. ListNet:NDCG_{10} (in cyan) operates on full feedback and Random:NDCG_{10} (in red) does not receive any feedback. *Best viewed in color.*

4.2 Contextual Setting

Objective: Since our contextual, online learning to rank with restricted feedback setting involves query-document matrices, we could conduct experiments on commercial, publicly available ranking data sets. Our objective was to demonstrate that it is possible to learn a good ranking function, even with highly restricted feedback, when standard online learning to rank algorithms would require full feedback at end of each round. As stated before, though our algorithm is designed to minimize surrogate based regret, the surrogate loss is only a proxy for ranking measures like NDCG. The users care about the ranking presented to them, and indeed the algorithm interacts with users by presenting ranked lists and getting feedback on top ranked item(s). We tested the quality of the ranked lists, and hence the performance of the evolving ranking functions, against the full relevance vectors, via ranking measure NDCG, cutoff at the 10th item. NDCG, cutoff at a point n, is defined as follows:

$$NDCG_n(\sigma, R) = \frac{1}{Z_n(R)} \sum_{i=1}^{n} \frac{2^{R(\sigma(i))} - 1}{\log_2(1+i)}$$

where $Z_n(R) = \max_{\sigma \in S_m} \sum_{i=1}^{n} \frac{2^{R(\sigma(i))}-1}{\log_2(1+i)}$. We want to emphasize that the algorithm was in no way affected by the fact that we were measuring its performance with respect to NDCG cutoff at 10. In fact, the cutoff point can be varied, but usually, researchers report performance under NDCG cutoff at 5 or 10.

Baselines: We applied Algorithm 2, with top 1 feedback, on Squared, KL (un-normalized ListNet) and SmoothDCG surrogates, and with top 2 feedback, on the RankSVM surrogate. Based on the objective of our work, we selected two different ranking algorithms as baselines. The first one is the online version of ListNet ranking algorithm (which is essentially OGD on cross-entropy function), with full relevance vector revealed at end of every round. ListNet is not only one of the most cited ranking algorithms (over 700 citations according to Google Scholar), but also one of the most validated algorithms (Tax et al., 2015). We emphasize that some of the ranking algorithms in literature, which have shown better empirical performance than ListNet, are based on non-convex surrogates with complex, non-linear ranking functions. These algorithms cannot usually be converted into online algorithms which learn from streaming data. Our second algorithm is a simple, fully random algorithm , which outputs completely random ranking of documents at each round. This algorithm, in effect, receives no feedback at end of each round. *Thus, we are comparing Algorithm 2, which learns from highly restricted feedback, with an algorithm which learns from full feedback and an algorithm which receives no feedback and learns nothing.*

Data sets: We compared the various ranking functions on two large scale commercial data sets. They were Yahoo's Learning to Rank Challenge data set (Chapelle and Chang, 2011) and a data set published by Russian search engine Yandex (IM-2009). The Yahoo data set had 19944 unique queries with 5 distinct relevance levels, while Yandex had 9126 unique queries with 5 distinct relevance levels.

Experimental Setting: We selected time horizon $T = 200,000$ (Yahoo) and $T = 100,000$

(Yandex) iterations for our experiments (thus, each algorithm went over each data set multiple times). The reason for choosing different time horizons is that there were roughly double the number of queries in Yahoo data set as compared to Yandex data set. All the online algorithms, other than the fully random one, involve learning rate η and exploration parameter γ (full information ListNet does not involve γ and SmoothDCG has an additional smoothing parameter ϵ). While obtaining our regret guarantees, we had established that $\eta = O(T^{-2/3})$ and $\gamma = O(T^{-1/3})$. In our experiments, for each instance of Algorithm 2, we selected a time varying $\eta = \frac{0.01}{t^{2/3}}$ and $\gamma = \frac{0.1}{t^{1/3}}$, for round t . We fixed $\epsilon = 0.01$. For ListNet, we selected $\eta = \frac{0.01}{t^{1/2}}$, since regret guarantee in OGD is established with $\eta = O(T^{-1/2})$. It should be noted that, while we varied the numerators in the parameters between $\{0.001, 0.01, 0.1, 1, 10\}$, and selected the parameters which produced the best (avg.) NDCG value over the entire time horizon, more emphasis could have been given on the issue of parameter tuning. However, better tuning of parameters could have only improved the performance of our algorithm. We plotted average NDCG_{10} against time, where average NDCG_{10} at time t is the cumulative NDCG_{10} up to time t, divided by t. We made an important observation while comparing the performance plots of the algorithms. As we have shown, construction of the unbiased estimators involve division by a probability value (Eq 11). The particular probability value can be $\frac{\gamma}{m}$, which is very small since γ goes to 0, when the top ranked item of the randomly drawn permutation does not match the top ranked item of the permutation given by the deterministic score vector (Sec 3.3). The mismatch happens with very low probability (since the random permutation is actually the deterministic permutation with high probability). While theoretically useful, in practice, dividing by such small value negatively affected the gradient estimation and hence the performance of our algorithm. So, when the mismatch happened, we made the denominator γ, instead of $\frac{\gamma}{m}$, and saw signficantly better performance.

Results: Figure 4 and Figure 5 show that ListNet, with full information feedback at end of each round, has highest average NDCG value throughout, as expected. However, Algorithm 2, with the convex surrogates, produce competitive performance. In fact, in the Yahoo data set, our algorithm, with RankSVM and KL, are very close to the performance of ListNet. RanSVM based algorithm does better than the others, since the estimator of RankSVM gradient is constructed from top 2 feedback, leading to lower variance of the estimator. KL based algorithm does much better than Squared loss based algorithm on Yahoo and equally as well on Yandex data set. *Crucially, our algorithm, based on all three convex surrogates, perform significantly better than the purely random algorithm, and are much closer to ListNet in performance, despite being much closer to the purely random algorithm in terms of feedback.* Our algorithm, with SmoothDCG, on the other hand, produce poor performance. We believe the reason is the non-convexity of the surrogate, which leads to the optimization procedure possibly getting stuck at a local minima. In batch setting, such problem is avoided by an annealing technique that successively reduces ϵ. We are not aware of an analogue in an online setting. Possible algorithms optimizing non-convex surrogates in an online manner, which require gradient of the surrogate, may be adapted to this partial feedback setting. The main purpose for including SmoothDCG in our work was to show that unbiased estimation of gradient, from restricted feedback, is possible even for non-convex surrogates.

5. Conclusion and Future Directions

We studied the problem of online learning to rank with a novel, restricted feedback model. The work is divided into two parts: in the first part, the set of items to be ranked is fixed, with varying user preferences, and in the second part, the items vary, as traditional query-documents matrices. The parts are tied by the feedback model; where the user gives feedback only on top k ranked items at end of each round, though the performance of the learner's ranking strategy is judged against full, *implicit* relevance vectors. In the first part, we gave comprehensive results on learnability with respect to a number of practically important ranking measures. We also gave a generic algorithm, along with an efficient instantiation, which achieves sub-linear regret rate for certain ranking measures. In the second part, we gave an efficient algorithm, which works on a number of popular ranking surrogates, to achieve sub-linear regret rate. We also gave an impossibility result for an entire class of ranking surrogates. Finally, we conducted experiments on simulated and commercial ranking data sets to demonstrate the performance of our algorithms.

We highlight some of the open questions of interest:

- What are the minimax regret rates with top k feedback model, for $k > 1$, for the ranking measures DCG, PairwiseLoss, Precision@n and their normalized versions NDCG, AUC and AP? Specifically, NDCG and AP are very popular in the learning to rank community. We showed that with top 1 feedback model, no algorithm can achieve sub-linear regret for NDCG and AP. Is it possible to get sub-linear regret with $1 < k < m$?

- We used FTPL as the sub-routine in Algorithm 1 to get an efficient algorithm. It might be possible to use other full information algorithms as sub-routine, retaining the efficiency, but getting tighter rates in terms of parameters (other than T) and better empirical performance.

- We applied Algorithm 2 on three convex surrogates and one non-convex surrogates. It would be interesting to investigate what other surrogates the algorithm can be applied on, guided by Lemma 14, and test its empirical performance. Since the algorithm learns a ranking function in the traditional query-documents setting, the question is more of practical interest.

- We saw that Algorithm 2, when applied to SmoothDCG, does not produce competitive empirical performance. It has been shown that a ranking function, learnt by optimizing SmoothDCG in the batch setting, has extremely competitive empirical performance (Qin and Liu, 2006). In the batch setting, simulated annealing is used to prevent the optimization procedure getting stuck in local minima. Any algorithm that optimizes non-convex surrogates in an online manner, by accessing its gradient, can replace the online gradient descent part in our algorithm and tested on SmoothDCG for empirical performance.

- We proved an impossibility result for NDCG calibrated surrogates with top 1 feedback. What is the minimax regret for NDCG calibrated surrogates, with top k feedback, for $k > 1$?

Acknowledgments

The authors acknowledge the support of NSF under grants IIS-1319810 and CAREER IIS-1452099.

Appendix A. Proofs for Non Contextual Setting

We provide technical details of results of Online Ranking with Restricted Feedback- Non Contextual Setting.

Proof of Theorem 8:

Proof. **Full information feedback**: Instead of top k feedback, assume that at end of each round, after learner reveals its action, the full relevance vector R is revealed to the learner. Since the knowledge of full relevance vector allows the learner to calculate the loss for every action (SumLoss(σ, R), $\forall\ \sigma$), the game is in full information setting, and the learner, using the full information algorithm, will have an $O(C\sqrt{T})$ expected regret for *SumLoss* (ignoring computational complexity). Here, C denotes parameter specific to the full information algorithm used.

Blocking with full information feedback: We consider a blocked variant of the full information algorithm. We still assume that full relevance vector is revealed at end of each round. Let the time horizon T be divided into K blocks, i.e., $\{B_1, \ldots, B_K\}$, of equal size. Here, $B_i = \{(i-1)(T/K)+1, (i-1)(T/K)+2, (i-1)T/K+3, \ldots, i(T/K)\}$. While operating in a block, the relevance vectors revealed at end of each round are accumulated, but not used to generate learner's actions like in the "without blocking" variant. Assume at the start of block B_i, there was some vector $s_{i-1} \in \mathbb{R}^m$. Then, at each round in the block, the randomized full information algorithm exploits s_{i-1} and outputs a permutation (basically maintains a distribution over actions, using s_{i-1}, and samples from the distribution). At the end of a block, the average of the accumulated relevance vectors (R_i^{avg}) for the block is used to update, as $s_{i-1} + R_i^{avg}$, to get s_i for the next block. The process is repeated for each block.

Formally, the full information algorithm creates distribution ρ_i over the actions, at beginning of block B_i, exploiting information s_{i-1}. Thus, $\rho_i \in \Delta$, where Δ is the probability simplex over $m!$ actions. Note that ρ_i is a deterministic function of $\{R_1^{avg}, \ldots, R_{i-1}^{avg}\}$.

Since action σ_t, for $t \in B_i$, is generated according to distribution ρ_i (we will denote this as $\sigma_t \sim \rho_i$), and in block i, distribution ρ_i is fixed, we have

$$\mathbb{E}_{\sigma_t \sim \rho_i}[\sum_{t \in [B_i]} SumLoss(\sigma_t, R_t)] = \sum_{t \in B_i} \rho_i \cdot [SumLoss(\sigma_1, R_t), \ldots, SumLoss(\sigma_{m!}, R_t)].$$

Thus, the total expected loss of this variant of the full information problem is:

$$\begin{aligned}
\mathbb{E}\sum_{t=1}^{T}[SumLoss(\sigma_t, R_t)] &= \sum_{i=1}^{K} \mathbb{E}_{\sigma_t \sim \rho_i}[\sum_{t \in B_i} SumLoss(\sigma_t, R_t)] \\
&= \sum_{i=1}^{K} \sum_{t \in B_i} \rho_i \cdot [SumLoss(\sigma_1, R_t), \ldots, SumLoss(\sigma_{m!}, R_t)] \\
&= \sum_{i=1}^{K} \sum_{t \in B_i} \rho_i \cdot [\sigma_1^{-1} \cdot R_t, \ldots, \sigma_{m!}^{-1} \cdot R_t)] \ \text{ By defn. of SumLoss} \\
&= \frac{T}{K} \sum_{i=1}^{K} \rho_i \cdot [\sigma_1^{-1} \cdot R_i^{avg}, \ldots, \sigma_{m!}^{-1} \cdot R_i^{avg}]
\end{aligned}$$

$$= \frac{T}{K}\sum_{i=1}^{K}\mathbb{E}_{\sigma_i\sim\rho_i}[SumLoss(\sigma_i, R_i^{avg})]$$
$$= \frac{T}{K}\mathbb{E}_{\sigma_1\sim\rho_1,\ldots,\sigma_K\sim\rho_K}\sum_{i=1}^{K}SumLoss(\sigma_i, R_i^{avg}) \quad (17)$$

where $R_i^{avg} = \sum_{t\in B_i}\frac{R_t}{T/K}$. Note that, at end of every block $i \in [K]$, ρ_i is updated to ρ_{i+1}. By the regret bound of the full information algorithm, for K rounds of full information problem, we have:

$$\mathbb{E}_{\sigma_1\sim\rho_1,\ldots,\sigma_K\sim\rho_K}\sum_{i=1}^{K}SumLoss(\sigma_i, R_i^{avg}) \le \min_{\sigma}\sum_{i=1}^{K}SumLoss(\sigma, R_i^{avg}) + C\sqrt{K}$$
$$= \min_{\sigma}\sum_{i=1}^{K}\sigma^{-1}\cdot R_i^{avg} + C\sqrt{K} \quad (18)$$
$$= \min_{\sigma}\sum_{t=1}^{T}\sigma^{-1}\cdot\frac{R_t}{T/K} + C\sqrt{K}$$

Now, since

$$\min_{\sigma}\sum_{t=1}^{T}\sigma^{-1}\cdot\frac{R_t}{T/K} = \min_{\sigma}\frac{1}{T/K}\sum_{t=1}^{T}SumLoss(\sigma, R_t),$$

combining Eq. 17 and Eq. 18, we get:

$$\sum_{t=1}^{T}\mathbb{E}_{\sigma_t\in\rho_i}[SumLoss(\sigma_t, R_t)] \le \min_{\sigma}\sum_{t=1}^{T}SumLoss(\sigma, R_t) + C\frac{T}{\sqrt{K}}. \quad (19)$$

Blocking with top k feedback: However, in our top k feedback model, the learner does not get to see the full relevance vector at each end of round. Thus, we form the unbiased estimator $\hat{R}_i$ of R_i^{avg}, using Lemma 7. That is, at start of each block, we choose $\lceil m/k\rceil$ time points uniformly at random, and at those time points, we output a random permutation which places k distinct objects on top (refer to Algorithm 1). At the end of the block, we form the vector $\hat{R}_i$ which is the unbiased estimator of R_i^{avg}. Note that using random vector $\hat{R}_i$ instead of true R_i^{avg} introduces randomness in the distribution ρ_i itself . But significantly, ρ_i is dependent only on information received up to the beginning of block i and is independent of the information collected in the block. We show the exclusive dependence as $\rho_i(\hat{R}_1, \hat{R}_2, .., \hat{R}_{i-1})$. Thus, for block i, we have:

$$\mathbb{E}_{\sigma_t\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}\sum_{t\in[B_i]}SumLoss(\sigma_t, R_t)$$
$$= \frac{T}{K}\mathbb{E}_{\sigma_i\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}SumLoss(\sigma_i, R_i^{avg})$$
(From Eq. 17)

$$= \frac{T}{K}\mathbb{E}_{\sigma_i\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}\mathbb{E}_{\hat{R}_i}SumLoss(\sigma_i,\hat{R}_i)$$
$$(\because \text{SumLoss is linear in both arguments and } \hat{R}_i \text{ is unbiased})$$
$$= \frac{T}{K}\mathbb{E}_{\hat{R}_i}\mathbb{E}_{\sigma_i\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}SumLoss(\sigma_i,\hat{R}_i).$$

In the last step above, we crucially used the fact that, since random distribution ρ_i is independent of $\hat{R}_i$, the order of expectations is interchangeable. Taking expectation w.r.t. $\hat{R}_1, \hat{R}_2, .., \hat{R}_{i-1}$, we get,

$$\mathbb{E}_{\hat{R}_1,...,\hat{R}_{i-1}}\mathbb{E}_{\sigma_t\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}\sum_{t\in[B_i]}SumLoss(\sigma_t,R_t) = \frac{T}{K}\mathbb{E}_{\hat{R}_1,...,\hat{R}_{i-1},\hat{R}_i}E_{\sigma_i\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}SumLoss(\sigma_i,\hat{R}_i). \tag{20}$$

Thus,

$$\begin{aligned}
\mathbb{E}\sum_{t=1}^{T}SumLoss(\sigma_t,R_t) &= \mathbb{E}\sum_{i=1}^{K}\sum_{t\in[B_i]}SumLoss(\sigma_t,R_t)\\
&= \sum_{i=1}^{K}E_{\hat{R}_1,...,\hat{R}_{i-1}}E_{\sigma_t\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}\sum_{t\in[B_i]}SumLoss(\sigma_t,R_t)\\
&= \frac{T}{K}\sum_{i=1}^{K}\mathbb{E}_{\hat{R}_1,...,\hat{R}_{i-1},\hat{R}_i}\mathbb{E}_{\sigma_i\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}SumLoss(\sigma_i,\hat{R}_i)\\
&\quad \text{(From Eq. 20)}\\
&= \frac{T}{K}\mathbb{E}_{\hat{R}_1,...,\hat{R}_K}\mathbb{E}_{\sigma_i\sim\rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})}\sum_{i=1}^{K}SumLoss(\sigma_i,\hat{R}_i)
\end{aligned}$$

Now using Eq. 18, we can upper bound the last term above as

$$\begin{aligned}
&\leq \frac{T}{K}\{\mathbb{E}_{\hat{R}_1,...,\hat{R}_K}[\min_{\sigma}\sum_{i=1}^{K}\sigma^{-1}\cdot\hat{R}_i] + C\sqrt{K}\}\\
&\leq \frac{T}{K}\{\min_{\sigma}\sum_{i=1}^{K}\sigma^{-1}\cdot R_i^{avg} + C\sqrt{K}\}\\
&\quad \text{(Jensen's Inequality)}\\
&\leq \min_{\sigma}\sum_{t=1}^{T}\sigma^{-1}\cdot R_t + C\frac{T}{\sqrt{K}}\\
&= \min_{\sigma}\sum_{t=1}^{T}SumLoss(\sigma,R_t) + C\frac{T}{\sqrt{K}}.
\end{aligned}$$

Effect of exploration: Since in each block B_i, $\lceil m/k \rceil$ rounds are reserved for exploration, where we do not draw σ_t from distribution ρ_i, we need to account for it in our

regret bound. Exploration leads to an extra regret of $C^I\lceil m/k\rceil K$, where C^I is a constant depending on the loss under consideration and specific full information algorithm used. The extra regret is because loss in each of the exploration rounds is at most C^I and there are a total of $\lceil m/k\rceil K$ exploration rounds over all K blocks. Thus, overall regret :

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] - \min_{\sigma}\sum_{t=1}^{T} SumLoss(\sigma, R_t) \leq C^I\lceil m/k\rceil K + C\frac{T}{\sqrt{K}}. \tag{21}$$

Now we optimize over K, to get:

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] \leq \min_{\sigma}\sum_{t=1}^{T} SumLoss(\sigma, R_t) + 2(C^I)^{1/3}C^{2/3}\lceil m/k\rceil^{1/3}T^{2/3} \tag{22}$$

□

Proof of Corollary 9:

Proof. We only need to instantiate the constants C and C^I from Theorem 8, with respect to SumLoss and FTPL. FTPL has the following parameters in its regret bound, for any online full information linear optimization problem: D is the ℓ_1 diameter of learner's action set, R is upper bound on difference between losses of 2 actions on same information vector and A is the ℓ_1 diameter of the set of information vectors (adversary's action set).

For SumLoss, it can be easily calculated that $R = \sum_{i=1}^{m}\sigma^{-1}(i)R(i) = O(m^2)$, $D = \sum_{i=1}^{m}\sigma^{-1}(i) = O(m^2)$, and $A = \sum_{i=1}^{m}R(i) = O(m)$.

FTPL gets $O(C\sqrt{T})$ regret over T rounds when $\epsilon = \sqrt{\frac{D}{RAT}}$. Here, $C = 2\sqrt{DRA} = O(m^{5/2})$ and $C^I = R$. Substituting the values of D, R, A, we conclude. □

In the proof above, we used the standard FTPL bound from the work of Kalai and Vempala (2005). In particular, we used uniform noise distribution drawn from a hypercube. We could have used other distributions, e.g. Gaussian. For Gaussian perturbations, Theorem 1.7 of Abernethy et al. (2016) gives an FTPL regret bound of $B_1B_2m^{1/4}\sqrt{2T}$ where B_1, B_2 are upper bounds on the maximum ℓ_1 norm of vectors in the learner's and adversary's sets respectively. For us, B_1 is $O(m^{3/2})$ and B_2 is $O(m^{1/2})$ giving a bound of $O(m^{2.25}\sqrt{T})$ which is slightly better than the $O(m^{2.5}\sqrt{T})$ bound used in the proof above.

Extension of results from SumLoss to DCG and Precision@n:

DCG: Due to structural differences, there are minor differences in definitions and proofs of theorems for SumLoss and DCG. We prove that local observability condition fails to hold for DCG, when restricted to top 1 feedback. We can skip the explicit proof of global observability, since the application of Algorithm 1 already establishes that $O(T^{2/3})$ regret can be achieved.

With slight abuse of notations, the loss matrix L implicitly means gain matrix, where entry in cell $\{i, j\}$ of L is $f(\sigma_i)\cdot g(R_j)$. The columns of feedback matrix H are expanded to account for greater number of moves available to adversary (due to multi-graded relevance vectors). In Definition 1, learner action i is optimal if $\ell_i \cdot p \geq \ell_j \cdot p,\ \forall j \neq i$.

In Definition 2, the maximum number of distinct elements that can be in a row of H is $n+1$. The signal matrix now becomes $S_i \in \{0,1\}^{(n+1)\times 2^m}$, where $(S_i)_{j,\ell} = \mathbb{1}(H_{i,\ell} = j-1)$.

Local Observability Fails: Since we are trying to establish a lower bound, it is sufficient to show it for binary relevance vectors, since the adversary can only be more powerful otherwise.

In Lemma 2, proved for SumLoss, $\ell_i \cdot p$ equates to $f(\sigma) \cdot \mathbb{E}[R]$. From definition of DCG, and from the structure and properties of $f(\cdot)$, it is clear that $\ell_i \cdot p$ is *maximized* under the same condition, i.e, $\mathbb{E}[R(\sigma_i(1)] \geq \mathbb{E}[R(\sigma_i(2)] \geq \ldots \geq \mathbb{E}[R(\sigma_i(m)]$. Thus, all actions are Pareto-optimal.

Careful observation of Lemma 3 shows that it is directly applicable to DCG, in light of extension of Lemma 2 to DCG.

Finally, just like in SumLoss, simple calculations with $m = 3$ and $n = 1$, in light of Lemma 2 and 3, show that local observability condition fails to hold.

We show the calculations:

$$S_{\sigma_1} = S_{\sigma_2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{aligned} \ell_{\sigma_1} =& [0, 1/2, 1/\log_2 3, 1/2 + 1/\log_2 3, 1, 3/2, \\ & 1 + 1/\log_2 3, 3/2 + 1/\log_2 3] \\ \ell_{\sigma_2} =& [0, 1/\log_2 3, 1/2, 1/2 + 1/\log_2 3, 1, 1 + 1/\log_2 3, \\ & 3/2, 3/2 + 1/\log_2 3] \end{aligned}$$

It is clear that $\ell_1 - \ell_2 \notin Col(S_{\sigma_1}^\top)$. Hence, Definition 5 fails to hold.

Proof of Corrolary 10

For DCG, the parameters of FTPL are: $R = \sum_{i=1}^m f^s(\sigma^{-1}(i)) g^s(R(i)) = O(m(2^n - 1)), D = \sum_{i=1}^m f^s(\sigma^{-1}(i)) = O(m), A = \sum_{i=1}^m g^s(R(i)) = O(m(2^n - 1))$. Again, $C = 2\sqrt{DRA}$ and $C^I = R$.

Precision@n:

Proof of Corrolary 11

For Precision@n, the parameters of FTPL are: $D = \sum_{i=1}^m f^s(\sigma^{-1}(i)) = O(n), R = \sum_{i=1}^m f^s(\sigma^{-1}(i)) g^s(R(i)) = O(n), A = \sum_{i=1}^m g^s(R(i)) = O(m)$. Again, $C = 2\sqrt{DRA}$ and $C^I = R$.

Non-existence of Sublinear Regret Bounds for NDCG, AP and AUC

We show via simple calculations that for the case $m = 3$, global observability condition fails to hold for NDCG, when feedback is restricted to top ranked item, and relevance vectors are restricted to take binary values. It should be noted that allowing for multi-graded relevance vectors only makes the adversary more powerful; hence proving for binary relevance vectors is enough.

The intuition behind failure to satisfy global observability condition is that the $NDCG(\sigma, R) = f(\sigma) \cdot g(R)$, where where $g(r) = R/Z(R)$ (see Sec.2.2). Thus, $g(\cdot)$ cannot be represented by univariate, scalar valued functions. This makes it impossible to write the difference between two rows of the loss matrix as linear combination of columns of (transposed) signal matrices.

Similar intuitions hold for AP and AUC.

Proof of Lemma 12

Proof. We will first consider NDCG and then, AP and AUC.

NDCG:

The first and last row of Table 1, when calculated for NDCG, are:

$$\ell_{\sigma_1} = [1, 1/2, 1/\log_2 3, (1+\log_2 3/2))/(1+\log_2 3), 1, 3/(2(1+1/\log_2 3)), 1, 1]$$
$$\ell_{\sigma_6} = [1, 1, \log_2 2/\log_2 3, 1, 1/2, 3/(2(1+1/\log_2 3)), (1+(\log_2 3)/2))/(1+\log_2 3), 1]$$

We remind once again that NDCG is a gain function, as opposed to SumLoss.

The difference between the two vectors is:

$$\ell_{\sigma_1} - \ell_{\sigma_6} = [0, -1/2, 0, -\log_2 3/(2(1+\log_2 3)), 1/2, 0, \log_2 3/(2(1+\log_2 3)), 0].$$

The signal matrices are same as SumLoss:

$$S_{\sigma_1} = S_{\sigma_2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$S_{\sigma_3} = S_{\sigma_5} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$S_{\sigma_4} = S_{\sigma_6} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

It can now be easily checked that $\ell_{\sigma_1} - \ell_{\sigma_6}$ does not lie in the (combined) column span of the (transposed) signal matrices.

We show similar calculations for AP and AUC.

AP:

We once again take $m = 3$. The first and last row of Table 1, when calculated for AP, is:

$$\ell_{\sigma_1} = [1, 1/3, 1/2, 7/12, 1, 5/6, 1, 1]$$
$$\ell_{\sigma_6} = [1, 1, 1/2, 1, 1/3, 5/6, 7/12, 1]$$

Like NDCG, AP is also a gain function.

The difference between the two vectors is:

$$\ell_{\sigma_1} - \ell_{\sigma_6} = [0, -2/3, 0, -5/12, 2/3, 0, 5/12, 0].$$

The signal matrices are same as in the SumLoss case:

$$S_{\sigma_1} = S_{\sigma_2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$S_{\sigma_3} = S_{\sigma_5} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$S_{\sigma_4} = S_{\sigma_6} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

It can now be easily checked that $\ell_{\sigma_1} - \ell_{\sigma_6}$ does not lie in the (combined) column span of the (transposed) signal matrices. □

AUC:

For AUC, we will show the calculations for $m = 4$. This is because global observability does hold with $m = 3$, as the normalizing factors for all relevance vectors with non-trivial mixture of 0 and 1 are same (i.e, when relevance vector has 1 irrelevant and 2 relevant objects, and 1 relevant and 2 irrelevant objects, the normalizing factors are same). The normalizing factor changes from $m = 4$ onwards; hence global observability fails.

Table 1 will be extended since $m = 4$. Instead of illustrating the full table, we point out the important facts about the loss matrix table with $m = 4$ for AUC.

The 2^4 relevance vectors heading the columns are:

$R_1 = 0000,\ R_2 = 0001,\ R_3 = 0010,\ R_4 = 0100,\ R_5 = 1000,\ R_6 = 0011,\ R_7 = 0101,\ R_8 = 1001,\ R_9 = 0110,\ R_{10} = 1010,\ R_{11} = 1100,\ R_{12} = 0111,\ R_{13} = 1011,\ R_{14} = 1101,\ R_{15} = 1110,\ R_{16} = 1111.$

We will calculate the losses of 1st and last (24th) action, where $\sigma_1 = 1234$ and $\sigma_{24} = 4321$.

$$\ell_{\sigma_1} = [0, 1, 2/3, 1/3, 0, 1, 3/4, 1/2, 1/2, 1/4, 0, 1, 2/3, 1/3, 0, 0]$$
$$\ell_{\sigma_{24}} = [0, 0, 1/3, 2/3, 1, 0, 1/4, 1/2, 1/2, 3/4, 1, 0, 1/3, 2/3, 1, 0]$$

AUC, like SumLoss, is a loss function.

The difference between the two vectors is:

$$\ell_{\sigma_1} - \ell_{\sigma_{24}} = [0, 1, 1/3, -1/3, -1, 1, 1/2, 0, 0, -1/2, -1, 1, 1/3, -1/3, -1, 0].$$

The signal matrices for AUC with $m = 4$ will be slightly different. This is because there are 24 signal matrices, corresponding to 24 actions. However, groups of 6 actions will share the same signal matrix. For example, all 6 permutations that place object 1 first will have same signal matrix, all 6 permutations that place object 2 first will have same signal matrix, and so on. For simplicity, we denote the signal matrices as S_1, S_2, S_3, S_4, where S_i corresponds to signal matrix where object i is placed at top. We have:

$$S_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$S_3 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$S_4 = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

It can now be easily checked that $\ell_{\sigma_1} - \ell_{\sigma_{24}}$ does not lie in the (combined) column span of transposes of S_1, S_2, S_3, S_4.

Appendix B. Proofs for Contextual Setting

We provide technical details of results of Online Ranking with Restricted Feedback- Contextual Setting.

Proof of Lemma 14: We restate the lemma before giving the proof, for ease of reading:

Lemma 14: Let $F : \mathbb{R}^m \mapsto \mathbb{R}^a$ be a vector valued function, where $m \geq 1$, $a \geq 1$. For a fixed $x \in \mathbb{R}^m$, let k entries of x be observed at random. That is, for a fixed probability distribution $\mathbb{P}$ and some random $\sigma \sim \mathbb{P}(S_m)$, observed tuple is $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$. The necessary condition for existence of an unbiased estimator of $F(x)$, that can be constructed from $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$, is that it should be possible to decompose $F(x)$ over k (or less) coordinates of x at a time. That is, $F(x)$ should have the following structure:

$$F(x) = \sum_{(i_1, i_2, \ldots, i_\ell) \in {}^mP_\ell} h_{i_1, i_2, \ldots, i_\ell}(x_{i_1}, x_{i_2}, \ldots, x_{i_\ell})$$

where $\ell \leq k$, ${}^mP_\ell$ is ℓ permutations of m and $h : \mathbb{R}^\ell \mapsto \mathbb{R}^a$. Moreover, when $F(x)$ can be written in form of Eq 10 , with $\ell = k$, an unbiased estimator of $F(x)$, based on $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$, is,

$$g(\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}) = \frac{\sum_{(j_1, j_2, \ldots, j_k) \in S_k} h_{\sigma(j_1), \ldots, \sigma(j_k)}(x_{\sigma(j_1)}, \ldots, x_{\sigma(j_k)})}{\sum_{(j_1, \ldots, j_k) \in S_k} p(\sigma(j_1), \ldots, \sigma(j_k))}$$

where S_k is the set of $k!$ permutations of [k] and $p(\sigma(1), \ldots, \sigma(k))$ is as in Eq 9 .

Proof. For a fixed $x \in \mathbb{R}^m$ and probability distribution $\mathbb{P}$, let the random permutation be $\sigma \sim \mathbb{P}(S_m)$ and the observed tuple be $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$. Let $\hat{G} = G(\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)})$ be an unbiased estimator of $F(x)$ based on the random observed tuple. Taking expectation, we get:

$$\begin{aligned} F(x) =& \mathbb{E}_{\sigma \sim \mathbb{P}}\left[\hat{G}\right] = \sum_{\pi \in S_m} \mathbb{P}(\pi) G(\pi, x_{\pi(1)}, \ldots, x_{\pi(k)}) \\ =& \sum_{(i_1, i_2, \ldots, i_k) \in {}^mP_k} \sum_{\pi \in S_m} \mathbb{P}(\pi) \mathbb{1}(\pi(1) = i_1, \pi(2) = i_2, \ldots, \pi(k) = i_k) G(\pi, x_{i_1}, x_{i_2}, \ldots, x_{i_k}) \end{aligned}$$

We note that $\mathbb{P}(\pi) \in [0, 1]$ is independent of x for all $\pi \in S_m$. Then we can use the following construction of function $h(\cdot)$:

$$h_{i_1, i_2, \ldots, i_k}(x_{i_1}, \ldots, x_{i_k}) = \sum_{\pi \in S_m} \mathbb{P}(\pi) \mathbb{1}(\pi(1) = i_1, \pi(2) = i_2, \ldots, \pi(k) = i_k) G(\pi, x_{i_1}, x_{i_2}, \ldots, x_{i_k})$$

and thus,

$$F(x) = \sum_{(i_1, i_2, \ldots, i_k) \in {}^mP_k} h_{i_1, i_2, \ldots, i_k}(x_{i_1}, x_{i_2}, \ldots, x_i)$$

Hence, we conclude that for existence of an unbiased estimator based on the random observed tuple, it should be possible to decompose $F(x)$ over k (or less) coordinates of x at a time. The "less than k" coordinates arguement follows simply by noting that if

$F(x)$ can be decomposed over ℓ coordinates at a time ($\ell < k$) and observation tuple is $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)})\}$, then any $k-\ell$ observations can be thrown away and the rest used for construction of the unbiased estimator.

The construction of the unbiased estimator proceeds as follows:

Let $F(x) = \sum_{i=1}^{m} h_i(x_i)$ and feedback is for top-1 item ($k = 1$). The unbiased estimator according to Lemma. 14 is:

$$g(\sigma, x_{\sigma(1)}) = \frac{h_{\sigma(1)}(x_{\sigma(1)})}{p(\sigma(1))} = \frac{h_{\sigma(1)}(x_{\sigma(1)})}{\sum_{\pi} \mathbb{P}(\pi)\mathbb{1}(\pi(1) = \sigma(1))}$$

Taking expectation w.r.t. σ, we get:

$$\mathbb{E}_{\sigma}[g(\sigma, x_{\sigma(1)})] = \sum_{i=1}^{m} \frac{h_i(x_i)(\sum_{\pi} \mathbb{P}(\pi)\mathbb{1}(\pi(1) = i))}{\sum_{\pi} \mathbb{P}(\pi)\mathbb{1}(\pi(1) = i)} = \sum_{i=1}^{m} h_i(x_i) = F(x)$$

Now, let $F(x) = \sum\limits_{i \neq j = 1}^{m} h_{i,j}(x_i, x_j)$ and the feedback is for top-2 item ($k = 2$). The unbiased estimator according to Lemma. 14 is:

$$g(\sigma, x_{\sigma(1)}, x_{\sigma(2)}) = \frac{h_{\sigma(1),\sigma(2)}(x_{\sigma(1)}, x_{\sigma(2)}) + h_{\sigma(2),\sigma(1)}(x_{\sigma(2)}, x_{\sigma(1)})}{p(\sigma(1), \sigma(2)) + p(\sigma(2), \sigma(1))}$$

We will use the fact that for any 2 permutations σ_1, σ_2, which places the same 2 objects in top-2 positions but in opposite order, estimators based on σ_1 (i.e, $g(\sigma_1, x_{\sigma_1(1)}, x_{\sigma_1(2)})$) and σ_2 (i.e, $g(\sigma_2, x_{\sigma_2(1)}, x_{\sigma_2(2)})$) have same numerator and denominator. For eg., let $\sigma_1(1) = i, \sigma_1(2) = j$. Numerator and denominator for $g(\sigma_1, x_{\sigma_1(1)}, x_{\sigma_1(2)})$ are $h_{i,j}(x_i, x_j) + h_{j,i}(x_j, x_i)$ and $p(i,j) + p(j,i)$ respectively. Now let $\sigma_2(1) = j, \sigma_2(2) = i$. Then numerator and denominator for $g(\sigma_2, x_{\sigma_2(1)}, x_{\sigma_2(2)})$ are $h_{j,i}(x_j, x_i) + h_{i,j}(x_i, x_j)$ and $p(j,i) + p(i,j)$ respectively.

Then, taking expectation w.r.t. σ, we get:

$$\begin{aligned}
\mathbb{E}_{\sigma} g(\sigma, x_{\sigma(1)}, x_{\sigma(2)}) &= \sum_{i \neq j = 1}^{m} \frac{(h_{i,j}(x_i, x_j) + h_{j,i}(x_j, x_i))p(i,j)}{p(i,j) + p(j,i)} \\
&= \sum_{i > j = 1}^{m} \frac{(h_{i,j}(x_i, x_j) + h_{j,i}(x_j, x_i))(p(i,j) + p(j,i))}{p(i,j) + p(j,i)} \\
&= \sum_{i > j = 1}^{m} (h_{i,j}(x_i, x_j) + h_{j,i}(x_j, x_i)) = \sum_{i \neq j = 1}^{m} h_{i,j}(x_i, x_j) = F(x)
\end{aligned}$$

This chain of logic can be extended for any $k \geq 3$. Explicitly, for general $k \leq m$, let $\mathbb{S}(i_1, i_2, \ldots, i_k)$ denote all permutations of the set $\{i_1, \ldots, i_k\}$. Then, taking expectation of the unbiased estimator will give:

$$
\begin{aligned}
&\mathbb{E}_\sigma g(\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}) \\
&= \sum_{(i_1,i_2,\ldots,i_k)\in\ {}^mP_k} \frac{\left(\sum_{(j_1,\ldots,j_k)\in\mathbb{S}(i_1,\ldots,i_k)} h_{j_1,\ldots,j_k}(x_{j_1},\ldots,x_{j_k})\right) p(i_1,\ldots,i_k)}{\sum_{(j_1,\ldots,j_k)\in\mathbb{S}(i_1,\ldots,i_k)} p(j_1,\ldots,j_k)} \\
&= \sum_{i_1>i_2>\ldots>i_k=1}^{m} \frac{\left(\sum_{(j_1,\ldots,j_k)\in\mathbb{S}(i_1,\ldots,i_k)} h_{j_1,\ldots,j_k}(x_{j_1},\ldots,x_{j_k})\right)\left(\sum_{(j_1,\ldots,j_k)\in\mathbb{S}(i_1,\ldots,i_k)} p(j_1,\ldots,j_k)\right)}{\sum_{(j_1,\ldots,j_k)\in\mathbb{S}(i_1,\ldots,i_k)} p(j_1,\ldots,j_k)} \\
&= \sum_{i_1>i_2>\ldots>i_k=1}^{m} \left(\sum_{(j_1,\ldots,j_k)\in\mathbb{S}(i_1,\ldots,i_k)} h_{j_1,\ldots,j_k}(x_{j_1},\ldots,x_{j_k})\right) \\
&= \sum_{(i_1,i_2,\ldots,i_k)\in\ {}^mP_k} h_{i_1,i_2,\ldots,i_k}(x_{i_1},x_{i_2},\ldots,x_{i_k}) = F(x)
\end{aligned}
$$

Note: For $k = m$, i.e., when the full feedback is received, the unbiased estimator is:

$$
\begin{aligned}
g(\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(m)}) &= \frac{\sum_{(j_1,j_2,\ldots,j_m)\in S_m} h_{\sigma(j_1),\ldots,\sigma(j_m)}(x_{\sigma(j_1)},\ldots,x_{\sigma(j_m)})}{\sum_{(j_1,\ldots,j_m)\in S_m} p(\sigma(j_1),\ldots,\sigma(j_m))} \\
&= \frac{\sum_{(i_1,i_2,\ldots,i_m)\in\ {}^mP_m} h_{i_1,\ldots,i_m}(x_{i_1},\ldots,x_{i_m})}{1} = F(x)
\end{aligned}
$$

Hence, with full information, the unbiased estimator of $F(x)$ is actually $F(x)$ itself, which is consistent with the theory of unbiased estimator.

□

Proof of Lemma 18 :

Proof. All our unbiased estimators are of the form $X^\top f(s, R, \sigma)$. We will actually get a bound on $f(s, R, \sigma)$ by using Lemma 17 and $p \to q$ norm relation, to equate out X:

$$\|\tilde{z}\|_2 = \|X^\top f(s,R,\sigma)\|_2 \leq \|X^\top\|_{1\to 2}\|f(s,R,\sigma)\|_1 \leq R_D\|f(s,R,\sigma)\|_1$$

since $R_D \geq \max_{j=1}^m \|X_{j:}\|_2$.

Squared Loss: The unbiased estimator of gradient of squared loss, as given in the main text, is:

$$\tilde{z} = X^\top(2(s - \frac{R(\sigma(1))e_{\sigma(1)}}{p(\sigma(1))}))$$

where $p(\sigma(1)) = \sum_{\pi\in S_m} \mathbb{P}(\pi)\mathbb{1}(\pi(1) = \sigma(1))$ ($\mathbb{P} = \mathbb{P}_t$ is the distribution at round t as in Algorithm 2)

Now we have:

$$\|s - \frac{R(\sigma(1))e_{\sigma(1)}}{p(\sigma(1))}\|_1 \leq mR_DU + \frac{R_{max}}{p(\sigma(1))} \leq \frac{mR_DUR_{max}}{p(\sigma(1)}$$

Thus, taking expectation w.r.t σ, we get:

$$\mathbb{E}_\sigma\|\tilde{z}\|_2^2 \leq m^2R_D^4U^2R_{max}^2\mathbb{E}_\sigma\frac{1}{p(\sigma(1))^2} = m^2R_D^4U^2R_{max}^2\sum_{i=1}^{m}\frac{p(i)}{p^2(i)}$$

Now, since $p(i) \geq \frac{\gamma}{m}$, $\forall\ i$, we get: $\mathbb{E}_\sigma\|\tilde{z}\|_2^2 \leq \frac{C^{sq}}{\gamma}$, where $C^{sq} = m^4R_D^4U^2R_{max}^2$.

RankSVM Surrogate: The unbiased estimator of gradient of the RankSVM surrogate, as given in the main text, is:

$$\tilde{z} = X^\top\left(\frac{h_{s,\sigma(1),\sigma(2)}(R(\sigma(1)),R(\sigma(2))) + h_{s,\sigma(2),\sigma(1)}(R(\sigma(2)),R(\sigma(1)))}{p(\sigma(1),\sigma(2)) + p(\sigma(2),\sigma(1))}\right)$$

where $h_{s,i,j}(R(i),R(j)) = \mathbb{1}(R(i) > R(j))\mathbb{1}(1 + s(j) > s(i))(e_j - e_i)$ and $p(\sigma(1),\sigma(2)) = \sum_{\pi\in S_m}\mathbb{P}(\pi)\mathbb{1}(\pi(1) = \sigma(1),\pi(2) = \sigma(2))$ ($\mathbb{P} = \mathbb{P}_t$ as in Algorithm 2)).

Now we have:

$$\|\frac{h_{s,\sigma(1),\sigma(2)}(R_{\sigma(1)},R_{\sigma(2)}) + h_{s,\sigma(2),\sigma(1)}(R_{\sigma(2)},R_{\sigma(1)})}{p(\sigma(1),\sigma(2)) + p(\sigma(2),\sigma(1))}\|_1 \leq \frac{2}{p(\sigma(1),\sigma(2)) + p(\sigma(2),\sigma(1))}$$

Thus, taking expectation w.r.t σ, we get:

$$\mathbb{E}_\sigma\|\tilde{z}\|_2^2 \leq 4R_D^2\mathbb{E}_\sigma\frac{1}{(p(\sigma(1),\sigma(2)) + p(\sigma(2),\sigma(1)))^2} \leq 4R_D^2\sum_{i>j}^{m}\frac{p(i,j) + p(j,i)}{(p(i,j) + p(j,i))^2}$$

Now, since $p(i,j) \geq \frac{\gamma}{m^2}$, $\forall\ i,j$, we get: $\mathbb{E}_\sigma\|\tilde{z}\|_2^2 \leq \frac{C^{svm}}{\gamma}$, where $C^{svm} = O(m^4R_D^2)$.

KL based Surrogate: The unbiased estimator of gradient of the KL based surrogate, as given in the main text, is:

$$\tilde{z} = X^\top\left(\frac{(\exp(s(\sigma(1))) - \exp(R(\sigma(1))))e_{\sigma(1)}}{p(\sigma(1))}\right)$$

where $p(\sigma(1)) = \sum_{\pi\in S_m}\mathbb{P}(\pi)\mathbb{1}(\pi(1) = \sigma(1))$ ($\mathbb{P} = \mathbb{P}_t$ as in Alg. 2)).

Now we have:

$$\|\frac{(\exp(s(\sigma(1))) - \exp(R(\sigma(1))))e_{\sigma(1)}}{p(\sigma(1))}\|_1 \leq \frac{\exp(R_DU)}{p(\sigma(1))}$$

Thus, taking expectation w.r.t σ, we get:

$$\mathbb{E}_\sigma\|\tilde{z}\|_2^2 \leq R_D^2\exp(2R_DU)\mathbb{E}_\sigma\frac{1}{p(\sigma(1))^2}$$

Following the same arguement as in squared loss, we get: $\mathbb{E}_\sigma\|\tilde{z}\|_2^2 \leq \frac{C^{KL}}{\gamma}$, where $C^{KL} = m^2R_D^2\exp(2R_DU)$.

□

Proof of Lemma 15 :

Proof. Let $m = 3$. Collection of all terms which are functions of 1st coordinate of R, i.e, $R(1)$, in the gradient of RankSVM is:

$$\mathbb{1}(R(1) > R(2))\mathbb{1}(1 + s(2) > s(1))(e_2 - e_1) + \mathbb{1}(R(2) > R(1))\mathbb{1}(1 + s(1) > s(2))(e_1 - e_2)$$
$$+ \mathbb{1}(R(1) > R(3))\mathbb{1}(1 + s(3) > s(1))(e_3 - e_1) + \mathbb{1}(R(3) > R(1))\mathbb{1}(1 + s(1) > s(3))(e_1 - e_3).$$

Now let $s(1) = 1, s(2) = 0, s(3) = 0$. Then the collection becomes:

$$\mathbb{1}(R(2) > R(1))(e_1 - e_2) + \mathbb{1}(R(3) > R(1))(e_1 - e_3) =$$
$$(\mathbb{1}(R(2) > R(1)) + \mathbb{1}(R(3) > R(1)))e_1 - \mathbb{1}(R(2) > R(1))e_2 - \mathbb{1}(R(3) > R(1))e_3$$

Now, if the gradient can be decomposed over each coordinate of R, then the collection of terms associated with $R(1)$ should *only and only be a function* of $R(1)$. Specifically, $(\mathbb{1}(R(2) > R(1)) + \mathbb{1}(R(3) > R(1)))$ (the non-zero coefficient of e_1) should be a function of only $R(1)$ (similarly for e_2 and e_3).

Now assume that the $(\mathbb{1}(R(2) > R(1)) + \mathbb{1}(R(3) > R(1)))$ can be expressed as a function of $R(1)$ only. Then the difference between the coefficient's values, for the following two cases: $R(1) = 0, R(2) = 0, R(3) = 0$ and $R(1) = 1, R(2) = 0, R(3) = 0$, would be same as the difference between the coefficient's values, for the following two cases: $R(1) = 0, R(2) = 1, R(3) = 1$ and $R(1) = 1, R(2) = 1, R(3) = 1$ (Since the difference would be affected only by change in $R(1)$ value). It can be clearly seen that the change in value between the first two cases is: $0 - 0 = 0$, while the change in value between the second two cases is: $2 - 0 = 2$. Thus, we reach a contradiction. □

Proof of Lemma 16 :

Proof. The term associated with the 1st coordinate of R, i.e, $R(1)$, in the gradient of ListNet is $= \sum_{i=1}^{m} \left(\frac{-\exp(R(i))}{\sum_{j=1}^{m} \exp(R(j))} + \frac{\exp(s(i))}{\sum_{j=1}^{m} \exp(s(j))} \right) e_i$ (in fact, the same term is associated with every coordinate of R).

Specifically, $f(R) = \left(\frac{-\exp(R(1))}{\sum_{j=1}^{m} \exp(R(j))} + \frac{\exp(s_1)}{\sum_{j=1}^{m} \exp(s(j))} \right)$ is the non-zero coefficient of e_1, associated with $R(1)$. Now, if $f(R)$ would have only been a function of $R(1)$, then $\frac{\partial^2 f(R)}{\partial R(1) \partial R(j)}, \forall\ j \neq 1$ would have been zero. It can be clearly seen this is not the case.

Now, the term associated jointly with $R(1)$ and $R(2)$, in the gradient of ListNet is same as before, i.e, $\sum_{i=1}^{m} \left(\frac{-\exp(R(i))}{\sum_{j=1}^{m} \exp(R(j))} + \frac{\exp(s(i))}{\sum_{j=1}^{m} \exp(s(j))} \right) e_i$ (since $R(1)$ and $R(2)$ are present in all the summation terms of the gradient).

Specifically, $f(R) = \left(\frac{-\exp(R(i))}{\sum_{j=1}^{m} \exp(R(j))} + \frac{\exp(s(i))}{\sum_{j=1}^{m} \exp(s(j))} \right)$ is the non-zero coefficient of e_1. Now, if $f(R)$ would have only been a function of $R(1)$ and $R(2)$, then $\frac{\partial^3 f(R)}{\partial R(1) \partial R(2) \partial R(j)}$, $\forall j \neq 1, j \neq 2$ would have been zero. It can be clearly seen this is not the case.

The same argument can be extended for any $k < m$.

□

Proof of Theorem. 21:

Proof. (*Sketch*) The proof builds on the proof of hopeless finite action partial monitoring games given by Piccolboni and Schindelhauer (2001). An examination of their proof of Theorem. 3 indicates that for hopeless games, there have to exist two probability distributions (over adversary's actions), which are indistinguishable in terms of feedback but the optimal learner's actions for the distributions are different. We first provide a mathematical explanation as to why such existence lead to hopeless games. Then, we provide a characterization of indistinguishable probability distributions in our problem setting, and then exploit the characterization of optimal actions for NDCG calibrated surrogates (Theorem 20) to explicitly construct two such probability distributions. This proves the result. Full proof is given below. □

Proof. We will first fix the setting of the online game. We consider $m = 3$ and fixed the document matrix $X \in \mathbb{R}^{3\times 3}$ to be the identity. At each round of the game, the adversary generates the fixed X and the learner chooses a score vector $s \in \mathbb{R}^3$. Making the matrix X identity makes the distinction between weight vectors w and scores s irrelevant since $s = Xw = w$. We note that allowing the adversary to vary X over the rounds only makes him more powerful, which can only increase the regret. We also restrict the adversary to choose binary relevance vectors. Once again, allowing adversary to choose multi-graded relevance vectors only makes it more powerful. Thus, in this setting, the adversary can now choose among $2^3 = 8$ possible relevance vectors. The learner's action set is infinite, i.e., the learner can choose any score vector $s = Xw = \mathbb{R}^m$. The loss function $\phi(s, R)$ is any NDCG calibrated surrogate and feedback is the relevance of top-ranked item at each round, where ranking is induced by sorted order (descending) of score vector. We will use p to denote randomized adversary one-short strategies, i.e. distributions over the 8 possible relevance score vectors. Let $s^*_p = \text{argmin}_s \, \mathbb{E}_{R\sim p}\phi(s, R)$. We note that in the definition of NDCG calibrated surrogates, Ravikumar et al. (2011) assume that the optimal score vector for each distribution over relevance vectors is unique and we subscribe to that assumption. The assumption was taken to avoid some boundary conditions.

It remains to specify the choice of U, a bound on the Euclidean norm of the weight vectors (same as score vectors for us right now) that is used to define the best loss in hindsight. It never makes sense for the learner to play anything outside the set $\cup_p s^*_p$ so that we can set $U = \max\{\|s\|_2 \, : \, s \in \cup_p s^*_p\}$.

The paragraph following Lemma 6 of Thm. 3 in Piccolboni and Schindelhauer (2001) gives the main intuition behind the argument the authors developed to prove hopelessness of finite action partial monitoring games. To make our proof self contained, we will explain the intuition in a rigorous way.

Key insight: Two adversary strategies $p, \tilde{p}$ are said to be indistinguishable from the learner's feedback perspective, if for every action of the learner, the probability distribution over the feedbacks received by learner is the same for p and $\tilde{p}$. Now assume that adversary always selects actions according to one of the two such indistinguishable strategies. Thus, the learner will always play one of s^*_p and $s^*_{\tilde{p}}$. By uniqueness, $s^*_p \neq s^*_{\tilde{p}}$. Then, the learner

incurs a constant (non-zero) regret on any round where adversary plays according to p and learner plays $s^*_{\tilde{p}}$, or if the adversary plays according to $\tilde{p}$ and learner plays s^*_p. We show that in such a setting, adversary can simply play according to $(p+\tilde{p})/2$ and the learner suffers an expected regret of $\Omega(T)$.

Assume that the adversary selects $\{R_1, \ldots, R_T\}$ from product distribution $\otimes p$. Let the number of times the learner plays s^*_p and $s^*_{\tilde{p}}$ be denoted by random variables N_1^p and N_2^p respectively, where N^p shows the exclusive dependence on p. It is always true that $N_1^p + N_2^p = T$. Moreover, let the expected per round regret be ϵ_p when learner plays $s^*_{\tilde{p}}$, where the expectation is taken over the randomization of adversary. Now, assume that adversary selects $\{R_1, \ldots, R_T\}$ from product distribution $\otimes\tilde{p}$. The corresponding notations become $N_1^{\tilde{p}}$ and $N_2^{\tilde{p}}$ and $\epsilon_{\tilde{p}}$. Then,

$$\mathbb{E}_{(R_1,\ldots,R_T)\sim\otimes p}\mathbb{E}_{(s_1,\ldots,s_T)}[\text{Regret}((s_1,\ldots,s_T),(R_1,\ldots,R_T))] = 0\cdot\mathbb{E}[N_1^p] + \epsilon_p\cdot\mathbb{E}[N_2^p]$$

and

$$\mathbb{E}_{(R_1,\ldots,R_T)\sim\otimes\tilde{p}}\mathbb{E}_{(s_1,\ldots,s_T)}[\text{Regret}((s_1,\ldots,s_T),(R_1,\ldots,R_T))] = \epsilon_{\tilde{p}}\cdot\mathbb{E}[N_1^{\tilde{p}}] + 0\cdot\mathbb{E}[N_2^{\tilde{p}}]$$

Since p and $\tilde{p}$ are indistinguishable from perspective of learner, $\mathbb{E}[N_1^p] = \mathbb{E}[N_1^{\tilde{p}}] = \mathbb{E}[N_1]$ and $\mathbb{E}[N_2^p] = \mathbb{E}[N_2^{\tilde{p}}] = \mathbb{E}[N_2]$. That is, the random variable denoting number of times s^*_p is played by learner does not depend on adversary distribution (same for $s^*_{\tilde{p}}$.). Using this fact and averaging the two expectations, we get:

$$\begin{aligned}\mathbb{E}_{(R_1,\ldots,R_T)\sim\frac{\otimes p+\otimes\tilde{p}}{2}}\mathbb{E}_{(s_1,\ldots,s_T)}[\text{Regret}((s_1,\ldots,s_T),(R_1,\ldots,R_T))] &= \frac{\epsilon_{\tilde{p}}}{2}\cdot\mathbb{E}[N_1] + \frac{\epsilon_p}{2}\cdot\mathbb{E}[N_2] \\ &\geq \min(\frac{\epsilon_p}{2},\frac{\epsilon_{\tilde{p}}}{2})\cdot\mathbb{E}[N_1+N_2] = \epsilon\cdot T\end{aligned}$$

Since

$$\sup_{R_1,\ldots,R_T}\mathbb{E}[\text{Regret}((s_1,\ldots,s_T),(R_1,\ldots,R_T))] \geq \mathbb{E}_{(R_1,\ldots,R_T)\sim\frac{\otimes p+\otimes\tilde{p}}{2}}\mathbb{E}_{(s_1,\ldots,s_T)}[\text{Regret}((s_1,\ldots,s_T),(R_1,\ldots,R_T))]$$

we conclude that for every learner algorithm, adversary has a strategy, s.t. learner suffers an expected regret of $\Omega(T)$.

Now, the thing left to be shown is the existence of two indistinguishable distributions p and $\tilde{p}$, s.t. $s^*_p \neq s^*_{\tilde{p}}$.

Characterization of indistinguishable strategies in our problem setting: Two adversary's strategies p and $\tilde{p}$ will be indistinguishable, in our problem setting, if for every score vector s, the relevances of the top-ranked item, according to s, are same for relevance vector drawn from p and $\tilde{p}$. Since relevance vectors are restricted to be binary, mathematically, it means that $\forall s$, $\mathbb{P}_{R\sim p}(R(\pi_s(1)) = 1) = \mathbb{P}_{R\sim\tilde{p}}(R(\pi_s(1)) = 1)$ (actually, we also need $\forall s$, $\mathbb{P}_{R\sim p}(R(\pi_s(1)) = 0) = \mathbb{P}_{R\sim\tilde{p}}(R(\pi_s(1)) = 0)$, but due to the binary nature, $\mathbb{P}_{R\sim p}(R(\pi_s(1)) = 1) = \mathbb{P}_{R\sim\tilde{p}}(R(\pi_s(1)) = 1) \implies \mathbb{P}_{R\sim p}(R(\pi_s(1)) = 0) = \mathbb{P}_{R\sim\tilde{p}}(R(\pi_s(1)) = 0)$). Since the equality has to hold $\forall s$, this implies $\forall j \in [m]$, $\mathbb{P}_{R\sim p}(R(j) = 1) = \mathbb{P}_{R\sim\tilde{p}}(R(j) = 1)$ (as every item will be ranked at top by some score vector). Hence, $\forall j \in [m]$, $\mathbb{E}_{R\sim p}[R(j)] = \mathbb{E}_{R\sim\tilde{p}}[R(j)] \implies \mathbb{E}_{R\sim p}[R] = \mathbb{E}_{R\sim\tilde{p}}[R]$. It can be

Table 3: Relevance and probability vectors.

p	0.0	0.1	0.15	0.05	0.2	0.3	0.2	0.0
$\tilde{p}$	0.0	0.3	0.0	0.0	0.15	0.15	0.4	0.0
Rel.	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
	0	1	1	0	1	0	0	1
	0	1	0	1	0	1	0	1
	0	0	1	1	0	0	1	1

seen clearly that the chain of implications can be reversed. Hence, $\forall s$, $\mathbb{P}_{R\sim p}(R(\pi_s(1)) = 1) = \mathbb{P}_{R\sim\tilde{p}}(R(\pi_s(1)) = 1) \iff \mathbb{E}_{R\sim p}[R] = \mathbb{E}_{R\sim\tilde{p}}[R]$.

Explicit adversary strategies: Following from the discussion so far and Theorem 20, if we can show existence of two strategies p and $\tilde{p}$ s.t. $\mathbb{E}_{R\sim p}[R] = \mathbb{E}_{R\sim\tilde{p}}[R]$, but argsort $\left(\mathbb{E}_{R\sim p}\left[\frac{G(\mathbf{R})}{Z(R)}\right]\right) \neq$ argsort $\left(\mathbb{E}_{R\sim\tilde{p}}\left[\frac{G(\mathbf{R})}{Z(R)}\right]\right)$, we are done.

The 8 possible relevance vectors (adversary's actions) are $(R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8) = (000, 110, 101, 011, 100, 010, 001, 111)$. Let the two probability vectors be:

$$p = (0.0, 0.1, 0.15, 0.05, 0.2, 0.3, 0.2, 0.0)$$
$$\tilde{p} = (0.0, 0.3, 0.0, 0.0, 0.15, 0.15, 0.4, 0.0).$$

The data is provided in table format in Table. 3.

Under the two distributions, it can be checked that $\mathbb{E}_{R\sim p}[R] = \mathbb{E}_{R\sim\tilde{p}}[R] = (0.45, 0.45, 0.4)^\top$.

However, $\mathbb{E}_{R\sim p}\left[\frac{G(\mathbf{R})}{Z(R)}\right] = (0.3533, 0.3920, 0.3226)^\top$, but $\mathbb{E}_{R\sim\tilde{p}}\left[\frac{G(\mathbf{R})}{Z(R)}\right] = (0.3339, 0.3339, 0.4000)^\top$.

Hence, argsort $\left(\mathbb{E}_{R\sim p}\left[\frac{G(\mathbf{R})}{Z(R)}\right]\right) = [2, 1, 3]^\top$ but argsort $\left(\mathbb{E}_{R\sim\tilde{p}}\left[\frac{G(\mathbf{R})}{Z(R)}\right]\right) \in \{[3, 1, 2]^\top, [3, 2, 1]^\top\}$. □

References

Jacob Abernethy, Chansoo Lee, and Ambuj Tewari. Perturbation techniques in online learning and optimization. In Tamir Hazan, George Papandreou, and Daniel Tarlow, editors, *Perturbations, Optimization, and Statistics*, Neural Information Processing Series, chapter 8. MIT Press, 2016.

Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

Nir Ailon. Improved bounds for online learning over the permutahedron and other ranking polytopes. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2014.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information Retrieval*, volume 463. ACM Press, 1999.

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Gábor Bartók and Csaba Szepesvári. Partial monitoring with side information. In *Algorithmic Learning Theory*, pages 305–319, 2012.

Gabor Bartok, Dean P. Foster, David Pal, Alexander Rakhlin, and Csaba Szepesvari. Partial monitoringclassification, regret bounds, and algorithms. *Mathematics of Operations Research*, 39(4):967–997, 2014.

Aditya Bhaskara and Aravindan Vijayaraghavan. Approximating matrix p-norms. In *Proceedings of the Twenty-Second Annual ACM-SIAM symposium on Discrete Algorithms*, pages 497–511. SIAM, 2011.

A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

Clément Calauzenes, Nicolas Usunier, and Patrick Gallinari. On the (non-) existence of convex, calibrated surrogate losses for ranking. In *Advances in Neural Information Processing Systems*, 2012.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International conference on Machine learning*, pages 129–136. ACM, 2007.

Nicolo Cesa-Bianchi, Gábor Lugosi, and Gilles Stoltz. Regret minimization under partial monitoring. *Mathematics of Operations Research*, 31(3):562–580, 2006.

Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research-Proceedings Track*, pages 1–24, 2011.

Olivier Chapelle and Mingrui Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information retrieval*, 13(3):216–235, 2010.

Olivier Chapelle, Quoc Le, and Alex Smola. Large margin optimization of ranking measures. In *NIPS Workshop: Machine Learning for Web Search*, 2007.

Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. *Advances in neural information processing systems*, 16(16):313–320, 2004.

David Cossock and Tong Zhang. Subset ranking using regression. In *Conference on Learning Theory*, pages 605–619, 2006.

David Cossock and Tong Zhang. Statistical analysis of bayes optimal subset ranking. *Information Theory, IEEE Transactions on*, 54(11):5140–5154, 2008.

John C. Duchi, Lester W. Mackey, and Michael I. Jordan. On the consistency of ranking algorithms. In *Proceedings of the 27th International Conference on Machine Learning*, pages 327–334, 2010.

Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. Online convex optimization in the bandit setting. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 385–394, 2005.

Dean P. Foster and Alexander Rakhlin. No internal regret via neighborhood watch. In *International Conference on Artificial Intelligence and Statistics*, pages 382–390, 2012.

Claudio Gentile and Francesco Orabona. On multilabel classification and ranking with bandit feedback. *The Journal of Machine Learning Research*, 15(1):2451–2487, 2014.

Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank. *Information Retrieval*, 16 (1):63–90, 2013.

IM-2009. `http://imat2009.yandex.ru/en/`, 2009.

Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48. ACM, 2000.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, pages 422–446, 2002.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD*, pages 133–142. ACM, 2002.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, pages 291–307, 2005.

Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Foundations of Computer Science, 2003*, pages 594–605, 2003.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems*, pages 817–824, 2008.

Tian Lin, Bruno Abrahao, Robert Kleinberg, and John Lui. Combinatorial partial monitoring game with linear feedback and its applications. In *Proceedings of the 31th International Conference on Machine Learning*, pages 901–909. ACM, 2014.

Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.

Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 3–10, 2007.

Antonio Piccolboni and Christian Schindelhauer. Discrete prediction games with arbitrary feedback and loss. In *Computational Learning Theory*, pages 208–223. Springer, 2001.

Tao Qin and Tie-Yan Liu. Microsoft letor website, 2006. `http://research.microsoft.com/en-us/um/beijing/projects/letor/`.

Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine learning*, pages 784–791. ACM, 2008.

Filip Radlinski, Paul N Bennett, Ben Carterette, and Thorsten Joachims. Redundancy, diversity and interdependent document relevance. In *ACM SIGIR*, pages 46–52. ACM, 2009.

Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. On NDCG consistency of listwise ranking methods. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 618–626, 2011.

Mark Sanderson. *Test collection based evaluation of information retrieval systems*, volume 13. Now Publishers Inc, 2010.

Niek Tax, Sander Bockting, and Djoerd Hiemstra. A cross-benchmark comparison of 87 learning to rank methods. *Information Processing and Management*, pages 757–772, 2015.

Zheng Wen, Branislav Kveton, and Azin Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In *Proceedings of the 32nd International Conference on Machine Learning*, 2014.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of ACM SIGIR*, pages 271–278, 2007.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.