Learning to Rank: Online Learning, Statistical Theory and Applications

by

Sougata Chaudhuri

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Statistics)
in The University of Michigan
2016

Doctoral Committee:

Assistant Professor Ambuj Tewari, Chair
Assistant Professor Jacob Abernethy
Associate Professor Long Nguyen
Associate Professor Clayton D. Scott


THE UNIVERSITY OF MICHIGAN
ARTES · SCIENTIA · VERITAS
1817

ACKNOWLEDGEMENTS

I would first like to express my deepest gratitude to my thesis advisor Ambuj Tewari, who has been a pillar of strength and constant source of encouragement throughout my graduate school journey. He was always present to brainstorm and discuss new ideas and cheerfully responded to the trivial questions. He stood by me as I navigated through the difficult path of getting papers accepted at competitive peer reviewed machine learning conferences, which is truly a test of patience and perseverance. Lastly, I truly appreciate his continuous three years of financial support through research grants, which helped me focus on research and avoid other distractions.

I would like to thank XuanLong Nguyen, Clay Scott and Jake Abernethy, for agreeing to be on my defense committee. Specifically, XuanLong's Stats 601 course and Clay's EECS 598 course motivated me to pursue a dissertation in the field of supervised machine learning. I have greatly enjoyed reading some of Jake's papers on topics in online learning, which got me interested in the direction of online learning to rank. I would like to thank Georgios Theocharous and Mohammad Ghavamzadeh for their excellent mentorship during my internship at Adobe labs. I would like to thank all the faculty members who taught me in graduate school and helped me develop a broad perspective on statistics and machine learning. I would also like to thank all my friends in the department who I have come to know over the past five years and who have taken the graduate school journey with me.

Lastly, a big thank you to my parents, Siddhartha and Manjula Chaudhuri and my wife, Twisha Chatterjee. Without you guys, I would not have been where I am now. Thank you for being with me throughout the journey.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Learning to Rank: Online Learning, Statistical Theory and Applications

by

Sougata Chaudhuri

Chair: Ambuj Tewari

Learning to rank is a supervised machine learning problem, where the output space is the special structured space of *permutations*. Learning to rank has diverse application areas, spanning information retrieval, recommendation systems, computational biology and others.

In this dissertation, we make contributions to some of the exciting directions of research in learning to rank. In the first part, we extend the classic, online perceptron algorithm for classification to learning to rank, giving a loss bound which is reminiscent of Novikoff's famous convergence theorem for classification. In the second part, we give strategies for learning ranking functions in an online setting, with a novel, feedback model, where feedback is restricted to labels of top ranked items. The second part of our work is divided into two sub-parts; one without side information and one with side information. In the third part, we provide novel generalization error bounds for algorithms applied to various Lipschitz and/or smooth ranking surrogates. In the last part, we apply ranking losses to learn policies for personalized advertisement recommendations, partially overcoming the problem of click sparsity. We conduct experiments on various simulated and commercial datasets, comparing our strategies with baseline strategies for online learning to rank and

personalized advertisement recommendation.

CHAPTER I

Introduction

A central problem in machine learning is: how to use empirical data to learn a functional dependence between "inputs" and "outputs"? Classical statistics and machine learning problems such as classification and regression correspond to cases where the set of possible outputs is simple: $\{-1, +1\}$ in the classification setting, and an interval on the real line in the regression setting. Modern machine learning has advanced well beyond such simple output spaces and moved to complex, *structured output spaces*. For instance, the outputs could be trees, graphs, or strings on a finite alphabet. The problems dealing with structured out spaces encompass important areas such as natural language processing and computer vision.

An interesting structured output space is the space of *rankings* or *permutations* of a set of objects. Naturally, researchers have worked on adapting general algorithms for structured output spaces to space of rankings (*Chapelle et al.*, 2007; *Yue et al.*, 2007; *Chakrabarti et al.*, 2008). However, the set of full or partial rankings of a set of objects has a very nice structure that calls for approaches that exploit that particular structure. In fact, the growing importance of problems where the output space is specifically the space of rankings has led to the development of an entire field of research titled *learning to rank*. Problems in learning to rank arise in diverse fields like search and information retrieval (*Liu*, 2011), computational biology (*Henneges et al.*, 2011) and recommendation

systems (*Karatzoglou et al.*, 2013). In fact, an excellent discourse on the application of learning to rank in recommendation systems can be found in the following "Netflix Blog" (http://technocalifornia.blogspot.com/2013/07/recommendations-as-personalized.html). The importance of learning to rank can also be gauged from the fact that modern machine learning texts tend to devote entire chapters to it (*Mohri et al.*, 2012; *Schapire and Freund*, 2012).

1.1 Learning to Rank for Information Retrieval

In a learning to rank problem that frequently arises in information retrieval, the objective is to rank *documents* associated with a *query*, in order of *relevance* of the documents to the given query. Web search engine research teams create training datasets in which a number of queries, each with their associated documents and relevance levels, are given. A ranking function is learnt by using the training data in a batch setting, with the hope that it will accurately order documents for a test query, according to their true relevance levels for the query. We provide a concrete example: assume that the search query is "Learning to Rank". The four candidate documents retrieved for the query are: a wikipedia entry on learning to rank, a journal article explaining ongoing research in learning to rank, a webpage which includes some learning to rank algorithms' codes and a webpage which just has the word "ranking" somewhere in the text. The relevances of the documents to the query, on a three point scale, are 2, 2, 1 and 0 respectively, according to expert judgement. The value 2 indicates that a document is highly relevant, 1 indicates that a document is somewhat relevant and 0 indicates that the document is not at all relevant. Ideally, a ranker should rank the documents in a way such that the first two documents are placed in top two positions (in any order), the third document is placed in third position and fourth document is placed in last position.

In order to measure the accuracy of a ranked list, in comparison to the ground truth in the form of relevance levels, various performance measures have been introduced. Some of the popular ones are: (Normalized) Discounted Cumulative Gain (NDCG) (*Järvelin*

and Kekäläinen, 2002), Average Precision (AP) (*Baeza-Yates and Ribeiro-Neto*, 1999), Expected Reciprocal Rank (ERR) (*Chapelle et al.*, 2009) and others. The measures are described in details at various points in the later chapters. Informally, the measures take in a ranking and relevance vector of m items and produce a real value. The higher (lower) the gain (loss) value, the more accurate is the ranked list.

Traditionally, ranking functions are represented as scoring functions with rankings obtained by sorting objects according to their scores. All the major performance measures are non-convex and discontinuous (and hence non-smooth), in the scores. Therefore, optimizing them during the training phase is a computationally intractable problem. For this reason, several existing ranking methods are based on minimizing *surrogate* losses. The surrogates losses are designed by balancing the conflicting requirements of remaining faithful to the original ranking measures while, at the same time, being easy to optimize (e.g. by being convex).

Ranking methods can be broadly categorized into three categories. In the *pointwise* approach, the problem is formulated as regression or classification problem, with the objective of predicting the true relevance level of individual documents (*Cossock and Zhang*, 2006a). In the *pairwise* approach, document pairs are taken as instances, and the problem is reduced to binary classification (which document in a pair is more relevant?) (*Herbrich et al.*, 1999; *Freund et al.*, 2003; *Burges et al.*, 2005). In the *listwise* approach, the entire list of documents associated with a query is taken as an instance, and listwise loss functions are minimized during training (*Cao et al.*, 2007a; *Xu and Li*, 2007).

1.2 Open Directions of Research in Learning to Rank

A non-exhaustive list of the plethora of existing ranking methods can be found in *Liu* (2011). In fact, the space of traditional ranking methods, where ranking functions are learnt from training data in a batch setting, has seen enormous development (*Chapelle and Chang*, 2011b). On the other hand, there are now a number of open, exciting directions of research

in learning to rank; some of which we will partially address in this thesis.

Online learning to rank : One direction of research involves developing algorithms for online learning of ranking functions. Instead of learning from labeled training data in a batch setting, online learning strategies continuously learn from streaming data. There are multiple advantages of learning from streaming data. A ranking function learnt from batch data might not be able to satisfy continuously changing user needs and preferences (*Radlinski et al.*, 2009). Learning strategies from streaming data can also balance between relevance and freshness of information (*Dong et al.*, 2010). Moreover, collecting reliable, supervised training data might be expensive and in some cases implausible. Of course, online learning also avoids difficulties associated with storing and processing large amounts of training data.

Theory of learning to rank: Learning to rank methods are compared by their empirical performance on benchmark datasets. However, benchmark datasets are not always reliable and often too small to truly capture the nuances of the methods. Learning theoretic guarantees are needed to predict performance on unseen test data. These guarantees are captured via *generalization error bounds* (does minimizing the surrogate on finite training data imply small expected surrogate loss on infinite unseen data?) and *calibration* (does small expected surrogate loss on infinite unseen data imply small target loss on infinite unseen data?) (*Ravikumar et al.*, 2011).

Application of ranking techniques to online advertisements: Traditional online advertisement selection algorithms fall under the umbrella of *contextual bandits* (*Langford and Zhang*, 2008). One practical issue that the algorithms face is extreme low rate of clicks on ads, which creates learning difficulty. Ranking methods can be applied to partially address this issue; either by modifying the *policy space* of contextual bandit algorithms with rankers or directly ranking the advertisements according to click probability.

1.3 Contributions of the Thesis

Our contributions in this thesis are divided into the following four chapters.

In Chapter II, we provide a novel extension of the online *perceptron* algorithm for classification to the learning to rank problem. We propose a novel family of listwise, large margin ranking surrogates, which are adaptable to NDCG and AP measures and derive a perceptron-like algorithm using these surrogates. We show that, if there exists a perfect oracle ranker which can correctly rank, with some margin, each instance in an online sequence, the cumulative NDCG (or AP) induced loss of perceptron algorithm on that sequence is bounded by a constant, irrespective of the length of the sequence. This result is a learning to rank analogue of Novikoff's convergence theorem for the classification perceptron. Moreover, we prove a lower bound on the cumulative loss achievable by any deterministic algorithm. Experiments on simulated datasets corroborate our theoretical results and demonstrate competitive performance on large industrial benchmark datasets.

In Chapter III, we consider two settings of online learning to rank with restricted feedback, cast as an online game between learner and adversary. In both settings, the learners objective is to present a ranked list of items to users (adversary). The learner's performance is judged on the entire ranked list and relevances of the items to the users. 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 and objective is to satisfy users with diverse preferences for the same set of items. The second setting is *contextual*, where items are defined by context and vary from user to user , in form of traditional query-document lists. 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 the first setting and ranking surrogates in the second setting. We also provide impossibility results for certain ranking measures and a certain class of surrogates, when feedback is restricted to top ranked item, i.e. $k = 1$. We empirically demonstrate the performance of our algorithms on simulated and real world datasets.

In Chapter IV, we investigate generalization error bounds in learning to rank. We show that for many Lipschitz surrogates, there is no degradation in generalization error bounds as document lists associated with queries become longer, which is an improvement over existing bounds. We also provide, for the first time in the learning to rank setting, generalization error bounds under ℓ_1 regularization and faster convergence rates if the loss function is smooth.

In Chapter V, we study the problem of personalized advertisement recommendation (PAR), which consist of a user visiting a system (website) and the system displaying one of K ads to the user. PAR problem is usually tackled by scalable *contextual bandit* algorithms, where the policies are generally based on classifiers. A practical problem in PAR is extreme click sparsity, due to very few users actually clicking on ads. We systematically study the drawback of using contextual bandit algorithms based on classifier-based policies, in face of extreme click sparsity. We then suggest an alternate policy, based on rankers, learnt by optimizing the Area Under the Curve (AUC) ranking loss, which can significantly alleviate the problem of click sparsity. We conduct extensive experiments on public datasets, as well as three industry proprietary datasets, to illustrate the improvement in click-through-rate (CTR) obtained by using the ranker-based policy over classifier-based policies.

1.4 Papers and Preprints

Every chapter in the thesis is a paper/preprint which has been published or is under review (at the time of writing). We list the publication information:

- Perceptron-like Algorithms for Online Learning to Rank- Under review for Journal of Artificial Intelligence Research (JAIR).
- Online Learning to Rank with Feedback on Top Ranked Items- Under review for Journal of Machine Learning Research (JMLR). The preprint is a combination of the following two published papers, with additional work.

  - Online Ranking with Top-1 Feedback- Appeared at 18th International Conference in Artificial Intelligence and Statistics (AISTATS), 2015 (Honorable Mention, Best Student Paper Award).
  - Online Learning to Rank with Feedback at the Top- Appeared at 19th International Conference in Artificial Intelligence and Statistics (AISTATS), 2016.

- Generalization Error Bounds for Learning to Rank: Does the Length of Documents Lists Matter?- Appeared at 32nd International Conference on Machine Learning (ICML), 2015.

- Personalized Advertisement Recommendation: A Ranking Approach to Address the Ubiquitous Click Sparsity Problem- Under review for International Conference on Information and Knowledge Management (CIKM), 2016.

CHAPTER II

Perceptron-like Algorithms for Online Learning to Rank

2.1 Introduction

The historical importance of the perceptron algorithm in the classification literature is immense (*Rosenblatt*, 1958; *Freund and Schapire*, 1999). *Vapnik* (1999) says, "In 1962 Novikoff proved the first theorem about the perceptron (Novikoff, 1962). *This theorem actually started learning theory*". Classically the perceptron algorithm was not linked to surrogate minimization but the modern perspective on perceptron is to interpret it as online gradient descent (OGD), during mistake rounds, on the hinge loss function (*Shalev-Shwartz*, 2011). The hinge loss has special properties that allow one to establish bounds on the cumulative zero-one loss (viz., the total number of mistakes) in classification, without making any statistical assumptions on the data generating mechanism. Novikoff's celebrated result about the perceptron says that, if there is a perfect linear classification function which can correctly classify, with some margin, every instance in an online sequence, then the total number of mistakes made by perceptron, on that sequence, is bounded.

Our work provides a novel extension of the perceptron algorithm to the learning to rank setting with a focus on two listwise ranking measures, NDCG and AP. Listwise measures are so named because the quality of ranking function is judged on an entire list of document, associated with a query, usually with an emphasis to avoid errors near top of the ranked list. Akin to the cumulative 0-1 loss (or, equivalently, the total number of mistakes)

bound of perceptron in classification, we develop bounds on cumulative losses induced by NDCG and AP. We prove that if there exists a *perfect* linear ranking function, which can correctly rank, with some margin, every instance in an online sequence, the cumulative loss of our perceptron is bounded, independent of the number of instances. The bound achieved, however, is sub-optimal, since it is dependent on number of documents per query. We prove a lower bound, on the cumulative loss achievable by any deterministic algorithm, which is *independent* of number of documents per query. We then propose a second perceptron type algorithm, which achieves the theoretical lower bound under a separability assumption, but has worse performance on real data where separability rarely holds.

We make the following contributions in this work.

- We develop a family of *listwise* large margin ranking surrogates. The family consists of Lipschitz functions and is parameterized by a set of weight vectors that makes the surrogates adaptable to losses induced by performance measures NDCG and AP. The family of surrogates is an extension of the hinge surrogate in classification that upper bounds the 0-1 loss. The family of surrogates has a special self-bounding property: the norm of the gradient of a surrogate can be bounded by the surrogate loss itself.

- We exploit the self bounding property of the surrogates to develop an online perceptron-like algorithm for learning to rank (Algorithm 2). We provide bounds on the cumulative NDCG and AP induced losses (Theorem 6). We prove that, if there is a *perfect* linear ranking function which can rank correctly, with some margin, every instance in an online sequence, our perceptron-like algorithm perfectly ranks all but a finite number of instances (Corollary 7). This implies that the cumulative loss induced by NDCG or AP is bounded by a constant, and our result can be seen as an extension of the classification perceptron mistake bound (Theorem 1). Our perceptron bound, however, depends linearly on the number of documents per query. In practice, during evaluation, NDCG is often cut off at a point which is much smaller than number of documents per query. In that scenario, we prove that the cumulative NDCG loss of

our perceptron is upper bounded by a constant which is dependent only on the cut-off point. (Theorem 8).

- We prove a lower bound, on the cumulative loss induced by NDCG or AP, that can be achieved by any deterministic online algorithm (Theorem 9) under a separability assumption. The lower bound is *independent* of the number of documents per query. We also prove that our lower bound is tight by proposing another perceptron-like algorithm which achieves the bound (Algorithm 3 and Theorem 10). However, the surrogate on which the perceptron type algorithm operates is not listwise in nature and does not adapt to different performance measures. Thus, its empirical performance on real data is significantly worse than the first perceptron algorithm (Algorithm 2).

- We provide empirical results on simulated as well as large scale benchmark datasets and compare the performance of our perceptron algorithm with the online version of the widely used ListNet learning to rank algorithm (*Cao et al.*, 2007a).

The rest of the chapter is organized as follows. Section 2.2 provides formal definitions and notations related to the problem setting. Section 2.3 provides a review of perceptron for classification, including algorithm and theoretical analysis. Section 2.4 introduces the family of listwise large margin ranking surrogates, and contrasts our surrogates with a number of existing large margin ranking surrogates in literature. Section 2.5 introduces the perceptron algorithm for learning to rank, and discusses various aspects of the algorithm and the associated theoretical guarantee. Section 2.6 establishes a minimax bound on NDCG/AP induced cumulative loss under the assumption that there is a perfect ranker with a margin. Section 2.7 compares our work with existing perceptron algorithms for ranking. Section 3.4 provides empirical results on simulated and large scale benchmark datasets. All missing proofs and discussions are provided in Appendix A.

2.2 Problem Definition

In learning to rank, we formally denote the input space as $\mathcal{X} \subseteq \mathbb{R}^{m \times d}$. Each input consists of m rows of document-query features represented as d dimensional vectors. Each input corresponds to a single query and, therefore, the m rows have features extracted from the same query but m different documents. In practice m changes from one input instance to another but we treat m as a constant for ease of presentation. For $X \in \mathcal{X}$, $X = (x_1, \ldots, x_m)^\top$, where $x_i \in \mathbb{R}^d$ is the feature extracted from a query and the ith document associated with that query. The supervision space is $\mathcal{Y} \subseteq \{0, 1, \ldots, n\}^m$, representing relevance score vectors. If $n = 1$, the relevance vector is binary graded. For $n > 1$, relevance vector is multi-graded. Thus, for $R \in \mathcal{Y}$, $R = (R_1, \ldots, R_m)^\top$, where R_i denotes relevance of ith document to a given query. Hence, R represents a vector and R_i, a scalar, denotes ith component of vector. Also, relevance vector generated at time t is denoted R_t with ith component denoted $R_{t,i}$.

The objective is to learn a ranking function which ranks the documents associated with a query in such a way that more relevant documents are placed ahead of less relevant ones. The prevalent technique is to learn a scoring function and obtain a ranking by sorting the score vector in descending order. For $X \in \mathcal{X}$, a linear scoring function is $f_w(X) = X \cdot w = s^w \in \mathbb{R}^m$, where $w \in \mathbb{R}^d$. The quality of the learnt ranking function is evaluated on a test query using various performance measures. We use two of the most popular performance measures in our chapter, viz. NDCG and AP.

NDCG, cut off at $k \leq m$ for a query with m documents, with relevance vector R and score vector s induced by a ranking function, is defined as follows:

$$\text{NDCG}_k(s, R) = \frac{1}{Z_k(R)} \sum_{i=1}^{k} G(R_{\pi_s(i)}) D(i). \tag{2.1}$$

Shorthand representation of $\text{NDCG}_k(s, R)$ is NDCG_k. Here, $G(r) = 2^r - 1$, $D(i) = \frac{1}{\log_2(i+1)}$, $Z_k(R) = \max_{\pi \in S_m} \sum_{i=1}^{k} G(R_{\pi(i)}) D(i)$. Further, S_m represents the set of permutations over m objects. $\pi_s = \text{argsort}(s)$ is the permutation induced by sorting score vector

s in descending order (we use π_s and $\mathrm{argsort}(s)$ interchangeably). A permutation π gives a mapping from ranks to documents and π^{-1} gives a mapping from documents to ranks. Thus, $\pi(i) = j$ means document j is placed at position i while $\pi^{-1}(i) = j$ means document i is placed at position j. For $k = m$, we denote $\text{NDCG}_m(s, R)$ as $\text{NDCG}(s, R)$. The popular performance measure, Average Precision (AP), is defined only for binary relevance vector, i.e., each component can only take values in $\{0, 1\}$:

$$\text{AP}(s, R) = \frac{1}{r} \sum_{j:R_{\pi_s(j)}=1} \frac{\sum_{i \le j} \mathbb{1}[R_{\pi_s(i)} = 1]}{j} \tag{2.2}$$

where $r = \|R\|_1$ is the total number of relevant documents.

All ranking performances measures are actually *gains*. When we say "NDCG induced loss", we mean a loss function that simply subtracts NDCG from its maximum possible value, which is 1 (same for AP).

2.3 Perceptron for Classification

We will first briefly review the perceptron algorithm for classification, highlighting the modern viewpoint that it executes online gradient descent (OGD) (*Zinkevich*, 2003) on hinge loss during mistake rounds and achieves a bound on total number of mistakes. This will allow us to directly compare and contrast our extension of perceptron to the learning to rank setting. For more details, we refer the reader to the survey written by *Shalev-Shwartz* (2011, Section 3.3).

In classification, an instance is of the form $x \in \mathbb{R}^d$ and corresponding supervision (label) is $y \in \{-1, 1\}$. A linear classifier is a scoring function $g_w(\cdot)$, parameterized by $w \in \mathbb{R}^d$, producing score $g_w(x) = x \cdot w = s \in \mathbb{R}$. Classification of x is obtained by using "sign" predictor on s, i.e., $\text{sign}(s) \in \{-1, 1\}$. The loss is of the form: $\ell(w, (x, y)) = \mathbb{1}[\text{sign}(x \cdot w) \neq y]$. The hinge loss is defined as: $\phi(w, (x, y)) = [1 - y(x \cdot w)]_+$, where $[a]_+ = \max\{0, a\}$.

The perceptron algorithm operates on the loss $f_t(w)$, defined on a sequence of data

$\{x_t, y_t\}_{t\geq 1}$, produced by an *adaptive adversary* as follows:

$$f_t(w) = \begin{cases} [1 - y_t(x_t \cdot w)]_+ & \text{if } \ell(w_t, (x_t, y_t)) = 1 \\ 0 & \text{if } \ell(w_t, (x_t, y_t)) = 0 \end{cases} \tag{2.3}$$

It is important to understand the concept of the loss $f_t(\cdot)$ and adaptive adversary here. An adaptive adversary is allowed to choose f_t at round t based on the moves of the perceptron algorithm (Algorithm 1) in previous rounds. Once the learner fixes its choice w_t at the end of step $t-1$, the adversary decides which function to play. It is either $[1 - y_t(x_t \cdot w)]_+$ or 0, depending on whether $\ell(w_t, (x_t, y_t))$ is 1 or 0 respectively. Notice that $f_t(w)$ is convex in both cases.

The perceptron updates a classifier $g_{w_t}(\cdot)$ (effectively updates w_t), in an online fashion. The update occurs by application of OGD on the sequence of functions $f_t(w)$ in the following way: perceptron initializes $w_1 = \vec{0}$ and uses update rule $w_{t+1} = w_t - \eta z_t$, where $z_t \in \partial f_t(w_t)$ (z_t is a subgradient) and η is the learning rate. If $\ell(w_t, (x_t, y_t)) = 0$, then $f_t(w_t) = 0$; hence $z_t = \vec{0}$. Otherwise, $z_t = -y_t x_t \in \partial f_t(w_t)$. Thus,

$$w_{t+1} = \begin{cases} w_t & \text{if } \ell(w_t, (x_t, y_t)) = 0 \\ w_t + \eta y_t x_t & \text{if } \ell(w_t, (x_t, y_t)) = 1. \end{cases} \tag{2.4}$$

The perceptron algorithm for classification is described below:

Theorem 1. *Suppose that the perceptron for classification algorithm runs on an online sequence of data* $\{(x_1, y_1), \ldots, (x_T, y_T)\}$ *and let* $R_x = \max_t \|x_t\|_2$. *Let* $f_t(\cdot)$ *be defined as in Eq. 2.3. For all* $u \in \mathbb{R}^d$, *the perceptron mistake bound is:*

$$\sum_{t=1}^{T} \ell(w_t, (x_t, y_t)) \leq \sum_{t=1}^{T} f_t(u) + R_x \|u\|_2 \sqrt{\sum_{t=1}^{T} f_t(u)} + R_x^2 \|u\|_2^2. \tag{2.5}$$

Algorithm 1 Perceptron Algorithm for Classification

Learning rate $\eta > 0$, $w_1 = \mathbf{0} \in \mathbb{R}^d$.
For $t = 1$ to T
  Receive x_t.
  Predict $p_t = \text{sign}(x_t \cdot w_t)$.
  Receive y_t
  If $\ell(w_t, (x_t, y_t)) \neq 0$
    $w_{t+1} = w_t + \eta y_t x_t$
  else
    $w_{t+1} = w_t$
End For

In the special case where there exists u s.t. $f_t(u) = 0$, $\forall\, t$, we have

$$\forall\, T,\ \sum_{t=1}^{T} \ell(w_t, (x_t, y_t)) \leq R_x^2 \|u\|_2^2. \tag{2.6}$$

As can be clearly seen from Eq. 2.5, the cumulative loss bound (i.e., total number of mistakes over T rounds) is upper bounded in terms of the cumulative sum of the functions $f_t(\cdot)$. In the special case where there exists a perfect linear classifier with margin, Eq. 2.6 shows that the total number of mistakes is bounded, regardless of the number of instances.

One drawback of the bound in Eq. 2.6 is that the concept of margin is not explicit, i.e., it is hidden in the norm of the parameter of the perfect classifier ($\|u\|_2$). Let us assume that there is a linear classifier parameterized by a unit norm vector $u_\star$, such that all instances x_t are not only correctly classified, but correctly classified with a *margin* γ, defined as:

$$y_t(x_t \cdot u_\star) \geq \gamma,\ \forall\, T \tag{2.7}$$

It is easy to see that the scaled vector $u = u_\star/\gamma$, whose norm is $1/\gamma^2$, will satisfy $f_t(u) = 0$ for all t. Therefore, we have following corollary.

Corollary 2. *If the margin condition* (2.7) *holds, then total number of mistakes is upper bounded by $\dfrac{R_x^2}{\gamma^2}$, a bound independent of the number of instances in the online sequence.*

2.4 A Novel Family of Listwise Surrogates

We define the novel SLAM family of loss functions: these are Surrogate, Large margin, Listwise and Lipschitz losses, Adaptable to multiple performance measures, and can handle Multiple graded relevance. For score vector $s \in \mathbb{R}^m$, and relevance vector $R \in \mathcal{Y}$, the family of convex loss functions is defined as:

$$\begin{aligned} \phi^v_{SLAM}(s, R) &= \min_{\delta \in \mathbb{R}^m} \sum_{i=1}^{m} v_i \delta_i \\ \text{s.t.} \quad \delta_i \geq 0, \forall\, i, \quad & s_i + \delta_i \geq \Delta + s_j, \text{ if } R_i > R_j, \ \ \forall\, i, j. \end{aligned} \tag{2.8}$$

The constant Δ denotes margin and $v = (v_1, \ldots, v_m)$ is an element-wise non-negative weight vector. Different vectors v, to be defined later, yield different members of the SLAM family. Though Δ can be varied for empirical purposes, we fix $\Delta = 1$ for our analysis. The intuition behind the loss setting is that scores associated with more relevant documents should be higher, with a margin, than scores associated with less relevant documents. The weights decide how much weight to put on the errors.

The following reformulation of $\phi^v_{SLAM}(s, R)$ will be useful in later derivations.

$$\sum_{i=1}^{m} v_i \ \max(0, \max_{j=1,\ldots,m} \{\mathbb{1}(R_i > R_j)(1 + s_j - s_i)\}) \ . \tag{2.9}$$

Lemma 3. *For any relevance vector R, the function $\phi^v_{SLAM}(\cdot, R)$ is* convex.

Proof. Claim is obvious from the representation given in Eq. 2.9. □

2.4.1 Weight Vectors Parameterizing the SLAM Family

As we stated after Eq. 2.8, different weight vectors lead to different members of the SLAM family. The weight vectors play a crucial role in the subsequent theoretical analysis. We will provide two weight vectors, v^{AP} and v^{NDCG}, that result in upper bounds for AP and

NDCG induced losses respectively. Later, we will discuss the necessity of choosing such weight vectors.

Since the losses in SLAM family is calculated with the knowledge of the relevance vector R, for ease of subsequent derivations, we can assume, without loss of generality, that documents are sorted according to their relevance levels. Thus, we assume that $R_1 \geq R_2 \geq \ldots \geq R_m$, where R_i is the relevance of document i. Note that both v^{AP} and v^{NDCG} depend on the relevance vector R but we hide that dependence in the notation to reduce clutter.

Weight vector for AP loss: Let $R \in \mathbb{R}^m$ be a binary relevance vector. Let r be the number of relevant documents (thus, $R_1 = R_2 = \ldots = R_r = 1$ and $R_{r+1} = \ldots = R_m = 0$). We define vector $v^{\text{AP}} \in \mathbb{R}^m$ as

$$v_i^{\text{AP}} = \begin{cases} \frac{1}{r} & \text{if } i = 1, 2, \ldots, r \\ 0 & \text{if } i = r+1, \ldots, m. \end{cases} \tag{2.10}$$

Weight vector for NDCG loss: For a given relevance vector $R \in \mathbb{R}^m$, we define vector $v^{\text{NDCG}} \in \mathbb{R}^m$ as

$$v_i^{\text{NDCG}} = \frac{G(R_i)D(i)}{Z(R)},\ i = 1, \ldots, m. \tag{2.11}$$

Note: Both weights ensure that $v_1 \geq v_2 \geq \ldots \geq v_m$ (since $R_1 \geq R_2 \geq \ldots \geq R_m$). Using the weight vectors, we have the following upper bounds.

Theorem 4. *Let $v^{AP} \in \mathbb{R}^m$ and $v^{NDCG} \in \mathbb{R}^m$ be the weight vectors as defined in Eq.* (2.10) *and Eq.* (2.11) *respectively. Let AP(s, R) and NDCG(s, R) be the AP value and NDCG value determined by relevance vector $R \in \mathbb{R}^m$ and score vector $s \in \mathbb{R}^m$. Then, the*

following inequalities hold, $\forall\ s, \forall\ R$

$$\begin{aligned} \phi_{SLAM}^{v^{AP}}(s, R) &\geq 1 - AP(s, R) \\ \phi_{SLAM}^{v^{NDCG}}(s, R) &\geq 1 - NDCG(s, R)\ . \end{aligned} \tag{2.12}$$

2.4.2 Properties of SLAM Family and Upper Bounds

Listwise Nature of SLAM Family: The critical property for a surrogate to be considered *listwise* is that the loss must be calculated over the entire list of documents as a whole, with errors at the top penalized more than errors at the bottom. Since perfect ranking places the most relevant documents at top, errors corresponding to most relevant documents should be penalized more in SLAM in order to be considered a listwise family. Both v^{NDCG} and v^{AP} has the property that the more relevant documents get more weight.

Upper Bounds on NDCG and AP: By Theorem 4, the weight vectors make losses in SLAM family upper bounds on NDCG and AP induced losses. The SLAM loss family is analogous to the hinge loss in classification. Similar to hinge loss, the surrogate losses of SLAM family are 0 when the predicted scores respect the relevance labels (with some margin). The upper bound property will be crucial in deriving guarantees for a perceptron-like algorithm in learning to rank. Like hinge loss, the upper bounds can possibly be loose in some cases, but, as we show next, the upper bounding weights make SLAM family Lipschitz continuous with a small Lipschitz constant. This naturally restricts SLAM losses from growing too quickly. Empirically, we will show that the perceptron developed based on the SLAM family produce competitive performance on large scale industrial datasets. Along with the theory, the empirical performance supports the fact that upper bounds are quite meaningful.

Lipschitz Continuity of SLAM: Lipschitz continuity of an arbitrary loss, $\ell(s, R)$ w.r.t. s in ℓ_2 norm, means that there is a constant L_2 such that $|\ell(s_1, R) - \ell(s_2, R)| \leq L_2 \|s_1 - s_2\|_2$, for all $s_1, s_2 \in \mathbb{R}^m$. By duality, it follows that $L_2 \geq \sup_s \|\nabla_s \ell(s, R)\|_2$. We calculate L_2 as

follows:

Let $b_{ij} = \{\mathbb{1}(R_i > R_j)(1 + s_j - s_i)\}$. The gradient of ϕ^v_{SLAM}, w.r.t. to s, from Eq. (2.9), is: $\nabla_s \phi^v_{SLAM}(s, R) = \sum_{i=1}^m v_i \, a^i$, where

$$a^i = \begin{cases} \mathbf{0} \in \mathbb{R}^m & \text{if } \max\limits_{j=1,\ldots,m} b_{ij} \leq 0 \\ \mathbf{e}_k - \mathbf{e}_i \in \mathbb{R}^m & \text{otherwise} \\ & k = \operatorname{argmax}\limits_{j=1,\ldots,m} b_{ij} \end{cases} \tag{2.13}$$

and $\mathbf{e}_i$ is a standard basis vector along coordinate i.

Since $\|a^i\|_1 \leq 2$, it is easy to see that $\|\nabla_s \phi^v_{SLAM}(s, R)\|_1 \leq 2 \sum_{i=1}^m v_i$. Since ℓ_1 norm dominates ℓ_2 norm, $\phi^v_{SLAM}(s, R)$ is Lipschitz continuous in ℓ_2 norm whenever we can bound $\sum_{i=1}^m v_i$. It is easy to check that $\sum_{i=1}^m v_i^{\text{AP}} = 1$ and $\sum_{i=1}^m v_i^{\text{NDCG}} = 1$. Hence, v^{NDCG} and v^{AP} induce Lipschitz continuous surrogates, with Lipschitz constant at most 2.

Comparison with Surrogates Derived from Structured Prediction Framework: We briefly highlight the difference between SLAM and listwise surrogates obtained from the structured prediction framework (*Chapelle et al.*, 2007; *Yue et al.*, 2007; *Chakrabarti et al.*, 2008). Structured prediction for ranking models assume that the supervision space is the space of *full rankings* of a document list. Usually a large number of full rankings are compatible with a relevance vector, in which case the relevance vector is arbitrarily mapped to a full ranking. In fact, here is a quote from one of the relevant papers (*Chapelle et al.*, 2007), "It is often the case that this y_q is not unique and we simply take of one of them at random" (y_q refers to a correct full ranking pertaining to query q). Thus, all but one correct full ranking will yield a loss. In contrast, in SLAM, documents with same relevance level are essentially exchangeable (see Eq. (2.9)). Thus, our assumption that documents are sorted according to relevance during design of weight vectors is without arbitrariness, and there will be no change in the amount of loss when documents within same relevance class are compared.

2.5 Perceptron-like Algorithms

We present a perceptron-like algorithm for learning a ranking function in an online setting, using the SLAM family. Since our proposed perceptron like algorithm works for both NDCG and AP induced losses, for derivation purposes, we denote a performance measure induced loss as *RankingMeasureLoss* (RML). Thus, RML can be NDCG induced loss or AP induced loss.

Informal Definition: The algorithm works as follows. At time t, the learner maintains a linear ranking function, parameterized by w_t. The learner receives X_t, which is the document list retrieved for query q_t and ranks it. Then the ground truth relevance vector R_t is received and ranking function updated according to the perceptron rule.

Let $b_{ij} = \{\mathbb{1}(R_i > R_j)(1 + s_j - s_i)\}$. For subsequent ease of derivations, we write SLAM loss from Eq. (2.9) as: $\phi^v_{SLAM}(s^w, R) = \sum_{i=1}^m v_i \, c_i,$, where

$$c_i = \begin{cases} 0 & \text{if } \max\limits_{j=1,\ldots,m} b_{ij} \leq 0 \\ 1 + s^w_k - s^w_i \in \mathbb{R} & \text{otherwise} \\ & k = \operatorname*{argmax}\limits_{j=1,\ldots,m} b_{ij}. \end{cases} \tag{2.14}$$

and $s^w = Xw \in \mathbb{R}^m$.

Like classification perceptron, our perceptron-like algorithm operates on the loss $f_t(w)$, defined on a sequence of data $\{X_t, R_t\}_{t\geq 1}$, produced by an *adaptive adversary* (i.e., an adversary who can see the learner's move before making its move) as follows:

$$f_t(w) = \begin{cases} \phi^{v_t}_{SLAM}(s^w_t, R_t) & \text{if RML}(s^{w_t}_t, R_t) \neq 0 \\ 0 & \text{if RML}(s^{w_t}_t, R_t) = 0 \end{cases} \tag{2.15}$$

Here, $s^w_t = X_t w$ and $v_t = v_t^{\text{NDCG}}$ or v_t^{AP} depending on whether RML is NDCG or AP induced loss. Since weight vector v depends on relevance vector R (Eq. (2.10), (2.11)), the subscript t in v_t denotes the dependence on R_t. Moreover, w_t is the parameter produced

by our perceptron (Algorithm 2) at the end of step $t-1$, with the adaptive adversary being influenced by the move of perceptron (recall Eq. 2.3 and discussion thereafter).

It is clear from Theorem. 4 and Eq. (2.15) that $f_t(w_t) \geq \text{RML}(s_t^{w_t}, R_t)$. It should also be noted that that $f_t(\cdot)$ is convex in either of the two cases. Thus, we can run the online gradient descent (OGD) algorithm (*Zinkevich*, 2003) to learn the sequence of parameters w_t, starting with $w_1 = \mathbf{0}$. The OGD update rule, $w_{t+1} = w_t - \eta z_t$, for some $z_t \in \partial f_t(w_t)$ and step size η, requires a subgradient z_t that, in our case, is computed as follows. When $\text{RML}(s_t^{w_t}, R_t) = 0$, we have $z_t = \mathbf{0} \in \mathbb{R}^d$. When $\text{RML}(s_t^{w_t}, R_t) \neq 0$, we have

$$
\begin{aligned}
z_t &= X_t^\top (\sum_{i=1}^{m} v_{t,i}\, a_{t,i}) \in \mathbb{R}^d, \\
a_{t,i} &= \begin{cases} \mathbf{0} \in \mathbb{R}^m & \text{if } c_i^t = 0 \\ \mathbf{e}_k - \mathbf{e}_i \in \mathbb{R}^m & \text{if } c_i^t \neq 0 \end{cases}
\end{aligned}
\tag{2.16}
$$

where $\mathbf{e}_k$ is the standard basis vector along coordinate k and $c_i^t \in \mathbb{R}$ is as defined in Eq. (2.14) (with $s^w = s_t^{w_t} = X_t w_t$).

We now obtain a perceptron-like algorithm for the learning to rank problem.

Algorithm 2 Perceptron Algorithm for Learning to Rank

Learning rate $\eta > 0$, $w_1 = \mathbf{0} \in \mathbb{R}^d$.
For $t = 1$ to T
  Receive X_t (document list for query q_t).
  Set $s_t^{w_t} = X_t w_t$, predicted ranking output= $\text{argsort}(s_t^{w_t})$.
  Receive R_t
  If $\text{RML}(s_t^{w_t}, R_t) \neq 0$ // Note: $\text{RML}(s_t^{w_t}, R_t) = \text{RML}(\text{argsort}(s_t^{w_t}), R_t)$
    $w_{t+1} = w_t - \eta z_t$ // z_t is defined in Eq. (2.16)
  else
    $w_{t+1} = w_t$
End For

2.5.1 Bound on Cumulative Loss

We provide a theoretical bound on the cumulative loss (as measured by RML) of perceptron for the learning to rank problem. The technique uses regret analysis of online convex optimization algorithms. We state the standard OGD bound used to get our main theorem (*Zinkevich*, 2003). An important thing to remember is that OGD guarantee holds for convex functions played by an adaptive adversary, which is important for OGD based perceptron Algorithm

Proposition (OGD regret). *Let f_t be a sequence of convex functions. The update rule of function parameter is $w_{t+1} = w_t - \eta z_t$, where $z_t \in \partial f_t(w_t)$. Then for any $w \in \mathbb{R}^d$, the following regret bound holds after T rounds,*

$$\sum_{t=1}^{T} f_t(w_t) \; - \sum_{t=1}^{T} f_t(w) \; \leq \; \frac{\|w\|_2^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|z_t\|_2^2. \tag{2.17}$$

We first control the norm of the subgradient z_t, defined in Eq. (2.16). To do this, we will need to use the $p \to q$ norm of matrix.

Definition ($\mathbf{p \to q}$ norm). *Let $A \in \mathbb{R}^{m \times n}$ be a matrix. The $p \to q$ norm of A is:*

$$\|A\|_{p \to q} = \max_{v \neq 0} \frac{\|Av\|_q}{\|v\|_p}$$

Lemma 5. *Let R_X be the bound on the maximum ℓ_2 norm of the feature vectors representing the documents. Let $v_{t,max} = \max\limits_{i,j} \{\frac{v_{t,i}}{v_{t,j}}\}$, $\forall\, i, j$ with $v_{t,i} > 0,\ v_{t,j} > 0$, and m be bound on number of documents per query. Then we have the following ℓ_2 norm bound,*

$$\forall\, t,\ \|z_t\|_2^2 \leq 4\, m\, R_X^2\, v_{t,max}\, f_t(w_t)\ . \tag{2.18}$$

Proof. For a mistake round t, we have $z_t = X_t^\top (\sum_{i=1}^{m} v_{t,i} a_{t,i})$ from Eq. (2.16) .

1st bound for z_t:

$$\|X_t^\top(\sum_{i=1}^{m} v_{t,i}\, a_{t,i})\|_2 \leq \|X_t^\top\|_{1\to 2}\| \sum v_{t,i}\, a_{t,i}\|_1 \leq 2R_X \sum v_{t,i} \leq 2R_X.$$

The first inequality uses the $1 \to 2$ norm and last inequality holds because $\sum_{i=1}^{m} v_i^{NDCG} = 1$ and $\sum_{i=1}^{m} v_i^{AP} = 1$.

2nd bound for z_t (exploiting self-bounding property of SLAM):

We note that in a mistake round, RML$(s_t^{w_t}, R_t) \neq 0$. Thus, $\exists\ i', k'$ s.t. $R_{t,i'} > R_{t,k'}$ but $s_{t,i'}^{w_t} < s_{t,k'}^{w_t}$ (i.e., there is at least 1 pair of documents whose ranks are inconsistent with their relevance levels). Now, $\phi_{SLAM}^{v_t}(s_t^{w_t}, R_t) = \sum v_{t,i}\ c_i^t$ (Eq. (2.14)). For (i', k'), we have $c_{i'}^t \geq 1 + s_{t,k'}^{w_t} - s_{t,i'}^{w_t} > 1$.

Since $R_{t,i'} > R_{t,k'}$, document i' has strictly greater than minimum possible relevance, i.e., $R_{t,i'} > 0$. By our calculations of weight vector v for both NDCG and AP, we have $v_{t,i'} > 0$. Thus, by definition, $v_{t,max} \geq 1$ (since $v_{t,i'} > 0$ and $\frac{v_{t,i'}}{v_{t,i'}} = 1$ and $v_{t,max} = \max_{i,j}\{\frac{v_{t,i}}{v_{t,j}}\},\ \forall\ i, j$ with $v_{t,i} > 0,\ v_{t,j} > 0$). Then, $\forall\ i$, $v_{t,i} \leq\ v_{t,max} \cdot v_{t,i'} \leq\ v_{t,max} \cdot v_{t,i'} \cdot c_{i'}^t$. Thus, we have:

$$\sum_{i=1}^{m} v_{t,i} \leq m\ v_{t,max}\ v_{t,i'}\ c_{i'}^t \leq\ m\ v_{t,max}(\sum_{i=1}^{m} v_{t,i}\ c_i^t) = m\ v_{t,max}\ \phi_{SLAM}^{v_t}(s_t^{w_t}, R_t).$$

Thus, $\|z_t\|_2 \leq 2R_X \sum_i v_{t,i} \leq 2R_X\ m\ v_{t,max}\ \phi_{SLAM}^{v_t}(s_t^{w_t}, R_t)$.

Combining 1st and 2nd bound for z_t, we get $\|z_t\|_2^2 \leq 4R_X^2 m\ v_{t,max}\ \phi_{SLAM}^{v_t}(s_t^{w_t}, R_t)$, for mistake rounds.

Since, for non-mistake rounds, we have $z_t = 0$ and $f_t(w_t) = 0$, we get the final inequality.

□

Taking $\max_{t=1}^{T} v_{t,max} \leq v_{max}$, we have the following theorem, which uses the norm bound on z_t:

Theorem 6. *Suppose Algorithm 2 receives a sequence of instances $(X_1, R_1), \ldots, (X_T, R_T)$ and let R_X be the bound on the maximum ℓ_2 norm of the feature vectors representing the documents. Then the following inequality holds, after optimizing over learning rate η, $\forall\, w \in \mathbb{R}^d$:*

$$\sum_{t=1}^{T} \textit{RML}(s_t^{w_t}, R_t) \leq \sum_{t=1}^{T} f_t(w) + \sqrt{4\|w\|_2^2 m R_X^2} \sqrt{\sum_{t=1}^{T} f_t(w)} + 4\|w\|_2^2 m R_X^2 v_{max}\,. \tag{2.19}$$

In the special case where there exists w s.t. $f_t(w) = 0$, $\forall\, t$, we have

$$\sum_{t=1}^{T} \textit{RML}(s_t^{w_t}, R_t) \;\leq\; 4\|w\|_2^2 m R_X^2 v_{max}. \tag{2.20}$$

Proof. The proof follows by plugging in expression for $\|z_t\|_2^2$ (Lemma 5) in OGD equation (Prop. OGD Regret), optimizing over η, using the algebraic trick: $x - b\sqrt{x} - c \leq 0 \implies x \leq b^2 + c + b\sqrt{c}$ and then using the inequality $f_t(w_t) \geq \text{RML}(s_t^{w_t}, R_t)$. □

Note: The perceptron bound, in Eq. 2.19, is a loss bound, i.e., the left hand side is cumulative NDCG/AP induced loss while right side is function of cumulative surrogate loss. We discuss in details the significance of this bound later.

Like perceptron for binary classification, the constant in Eq. 2.19 needs to be expressed in terms of a "margin". A natural definition of margin in case of ranking data is as follows: let us assume that there is a linear scoring function parameterized by a unit norm vector $w_\star$, such that all documents for all queries are ranked not only correctly, but correctly with a *margin* γ:

$$\min_{t=1}^{T} \min_{i,j:R_{t,i}>R_{t,j}} w_\star^\top X_{t,i} - w_\star^\top X_{t,j} \geq \gamma. \tag{2.21}$$

Corollary 7. *If the margin condition* (2.21) *holds, then total loss, for both NDCG and AP induced loss, is upper bounded by $\frac{4mR_X^2 v_{max}}{\gamma^2}$, a bound independent of the number of instances in the online sequence.*

Proof. Fix a t and the example (X_t, R_t). Set $w = w_\star/\gamma$. For this w, we have

$$\min_{i,j:R_{t,i}>R_{t,j}} w^\top X_{t,i} - w^\top X_{t,j} > 1,$$

which means that

$$\min_{i,j:R_{t,i}>R_{t,j}} s^w_{t,i} - s^w_{t,j} > 1$$

This immediately implies that $\mathbb{1}(R_{t,i} > R_{t,j})(1 + s^w_{t,j} - s^w_{t,i}) \leq 0, \forall\ i, j$. Therefore, $\phi^{v_t}_{SLAM}(s^w_t, R_t) = 0$ and hence $f_t(w) = 0$. Since this holds for all t, we have $\sum_{t=1}^T f_t(w) = 0$.

□

2.5.1.1 Perceptron Bound-General Discussion

We remind once again that RML$(s^{w_t}_t, R_t)$ is either $1-$NDCG$(s^{w_t}_t, R_t)$ or $1-$AP$(s^{w_t}_t, R_t)$, depending on measure of interest.

Dependence of perceptron bound on number of documents per query: The perceptron bound in Eq. 2.19 is meaningful only if v_{max} is a finite quantity.

For AP, it can be seen from the definition of v^{AP} in Eq. 2.10 that $v_{max} = 1$. Thus, for *AP induced loss, the constant in the perceptron bound is*: $\frac{4mR^2_X}{\gamma^2}$.

For NDCG, v_{max} depends on maximum relevance level. Assuming maximum relevance level is finite (in practice, maximum relevance level is usually below 5), $v_{max} = O(\log(m))$. Thus, for *NDCG induced loss, the constant in the perceptron bound is*: $\frac{4m\log(m)R^2_X}{\gamma^2}$.

Significance of perceptron bound: The main perceptron bound is given in Eq. 2.19, with the special case being captured in Corollary 7. At first glance, the bound might seem non-informative because the left side is the cumulative NDCG/AP induced loss bound, while the right side is a function of the cumulative surrogate loss.

The first thing to note is that the perceptron bound is derived from the regret bound in Eq. 2.17, which is the well-known regret bound of the OGD algorithm applied to an arbitrary convex, Lipschitz surrogate. So, even ignoring the bound in Eq. 2.19, the perceptron algorithm is a valid online algorithm, applied to the sequence of convex functions $f_t(\cdot)$, to learn ranking function w_t, with a meaningful regret bound. Second, as we had mentioned in the introduction, our perceptron bound is the extension of perceptron bound in classification, to the cumulative NDCG/AP induced losses in the learning to rank setting. This can be observed by noticing the similarity between Eq. 2.19 and Eq. 2.5. In both cases, the the cumulative target loss on the left is bounded by a function of the cumulative surrogate loss on the right, where the surrogate is the hinge (and hinge like SLAM) loss.

The interesting aspects of perceptron loss bound becomes apparent on close investigation of the cumulative surrogate loss term $\sum_{t=1}^{T} f_t(w)$ and comparing with the regret bound. It is well known that when OGD is run on any convex, Lipschitz surrogate, the guarantee on the regret scales at the rate $O(\sqrt{T})$. So, if we only ran OGD on an arbitrary convex, Lipschitz surrogate, then, even with the assumption of existence of a perfect ranker, the upper bound on the cumulative loss would have scaled as $O(\sqrt{T})$. However, in the perceptron loss bound, if $\sum_{t=1}^{T} f_t(w) = o(T^{\alpha})$, then the upper bound on the cumulative loss would scale as $O(T^{\alpha})$, which can be much better than $O(T^{1/2})$ for $\alpha < 1/2$. In the best case of $\sum_{t=1}^{T} f_t(w) = 0$, the total cumulative loss would be bounded, irrespective of the number of instances.

Comparison and contrast with perceptron for classification: The perceptron for learning to rank is an extension of the perceptron for classification, both in terms of the algorithm and the loss bound. To obtain the perceptron loss bounds in the learning to rank setting, we had to address multiple non-trivial issues, which do not arise in the classification setting. Among them, perhaps, the most challenging was that, unlike in classification, the NDCG/AP losses are not $\{0, 1\}$-valued. The analysis is trivial in classification perceptron since on a mistake round, the absolute value of gradient of hinge loss is 1, which is same as

the loss itself. In our setting, Lemma 5 is crucial, where we exploit the structure of SLAM surrogate to bound the square of gradient by the surrogate loss.

2.5.1.2 Perceptron Bound Dependent On NDCG Cut-Off Point

The bound on the cumulative loss in Eq. (2.19) is dependent on m, the maximum number of documents per query. It is often the case in learning to rank that though a list has m documents, the focus is on the top k documents ($k \ll m$) in the order sorted by score. The measure used for top-k documents is NDCG_k (Eq. 2.1) (there does not exist an equivalent definition for AP).

We consider a modified set of weights v^{NDCG_k} s.t. $\phi_{SLAM}^{v^{\text{NDCG}_k}}(s, R) \geq 1 - \text{NDCG}_k(s, R)$ holds $\forall\ s$, for every R. *We provide the definition of v^{NDCG_k} later in the proof of Theorem8 .*

Overloading notation with $v_t = v_t^{\text{NDCG}_k}$, let $v_{t,max} = \max_{i,j}\{\frac{v_{t,i}}{v_{t,j}}\}$ with $v_{t,i} > 0$, $v_{t,j} > 0$ and $v_{max} \geq \max_{t=1}^{T} v_{t,max}$.

Theorem 8. *Suppose the perceptron algorithm receives a sequence of instances $(X_1, R_1), \ldots, (X_T, R_T)$. Let k be the cut-off point of NDCG. Also, for any $w \in \mathbb{R}^d$, let $f_t(w)$ be as defined in Eq.* (2.15)*, but with $\phi_{SLAM}^{v_t}(s_t^w, R_t) = \phi_{SLAM}^{v_t^{NDCG_k}}(s_t^w, R_t)$. Then, the following inequality holds, after optimizing over learning rate η,*

$$\sum_{t=1}^{T}(1 - \textit{NDCG}_k(s_t^{w_t}, R_t)) \leq \sum_{t=1}^{T} f_t(w) + \sqrt{4\|w\|_2^2 k R_X^2 v_{max}}\sqrt{\sum_{t=1}^{T} f_t(w)} + 4\|w\|_2^2 k R_X^2 v_{max}. \tag{2.22}$$

In the special case where there exists w s.t. $f_t(w) = 0$, $\forall\ t$, we have

$$\sum_{t=1}^{T}(1 - \textit{NDCG}_k(s_t^{w_t}, R_t)) \leq 4\|w\|_2^2 k R_X^2 v_{max}. \tag{2.23}$$

Discussion: Assuming maximum relevance level is finite, we have $v_{max} = O(\log(k))$ (using definition of v^{NDCG_k}). Thus, the *constant term in the perceptron bound for NDCG_k induced loss is*: $4\|u\|^2 k \log(k) R_X^2$. This is a significant improvement from original error

term, even though the perceptron algorithm is running on queries with m documents, which can be very large. Margin dependent bound can be defined in same way as before.

Proof. We remind again that ranking performance measures only depend on the permutation of documents and individual relevance level. They do not depend on the identity of the documents. Documents with same relevance level can be considered to be interchangeable, i.e., relevance levels create equivalence classes. Thus, w.l.o.g., we assume that $R_1 \geq R_2 \geq \ldots \geq R_m$ and documents with same relevance level are sorted according to score. Also, $\pi^{-1}(i)$ means position of document i in permutation π.

We define v^{NDCG_k} as

$$v_i^{\text{NDCG}_k} = \begin{cases} \frac{G(R_i)D(i)}{Z_k(R)} & \text{if } i = 1, 2, \ldots, k \\ 0 & \text{if } i = k+1, \ldots, m. \end{cases} \tag{2.24}$$

We now prove the upper bound property that $\phi_{SLAM}^{v^{\text{NDCG}_k}}(s, R) \geq 1 - \text{NDCG}_k(s, R)$ holds $\forall\ s$, for every R. We have the following equations:

$$\frac{\sum_{i=1}^m G(R_i)D(i)\mathbb{1}(i \leq k)}{Z_k(R)} = 1 \text{ and } \text{NDCG}_k(s, R) = \frac{\sum_{i=1}^m G(R_i)D(\pi_s^{-1}(i))\mathbb{1}(\pi_s^{-1}(i) \leq k)}{Z_k(R)}.$$
$$\implies 1 - \text{NDCG}_k(s, R) = \frac{\sum_{i=1}^m G(R_i)\big(D(i)\mathbb{1}(i \leq k) - D(\pi_s^{-1}(i))\mathbb{1}(\pi_s^{-1}(i) \leq k)\big)}{Z_k(R)}.$$

For $i > k$: $D(i)\mathbb{1}(i \leq k) = 0$ and since $D(\pi_s^{-1}(i))$ is non-negative, every term in $1 - \text{NDCG}_k(s, R)$ is non-positive for $i > k$.

For $i \leq k$, there are four possible cases:

1. $i \geq \pi_s^{-1}(i)$ and $\pi_s^{-1}(i) > k$. This is infeasible since $i \leq k$.

2. $i \geq \pi_s^{-1}(i)$ and $\pi_s^{-1}(i) \leq k$. In this case, the numerator in $1-\text{NDCG}_k$ is $G(R_i)(D(i)-D(\pi_s^{-1}(i)))$. Now, since $D(\cdot)$ is a decreasing function, the contribution of the document i to NDCG induced loss is non-positive and can be ignored (since SLAM by

definition is sum of positive weighted indicator functions).

3. $i < \pi_s^{-1}(i)$ and $\pi_s^{-1}(i) > k$. In this case, the numerator in $1 - \text{NDCG}_k$ is $G(R_i)D(i)$. Since $i < \pi_s^{-1}(i)$, that means document i was outscored by a document j, where $i < j$ (otherwise, document i would have been put in a position same or above what it is at currently, by π_s, i.e, $i \geq \pi_s^{-1}(i)$.) Moreover, $R_i > R_j$ (because of the assumption that within same relevance class, scores are sorted). Hence the indicator of SLAM at i would have come on and $v_i^{\text{NDCG}_k} = \frac{G(R_i)(D(i)}{Z_k(R)}$.

4. $i < \pi_s^{-1}(i)$ and $\pi_s^{-1}(i) \leq k$. In this case, the numerator in $1-\text{NDCG}_k$ is $G(R_i)(D(i)-D(\pi_s^{-1}(i)))$. By same reason as c.), the indicator of SLAM at i would have come on and $v_i^{\text{NDCG}_k} > \dfrac{G(R_i)(D(i) - D(\pi_s^{-1}(i)))}{Z_k(R)}$ by definition of v^{NDCG_k} and the fact that $D(i) > D(i) - D(\pi_s^{-1}(i))$.

Hence, the upper bound property holds.

The proof of Theorem 8 now follows directly following the argument in the proof of Lemma 5, by noting a few things:

a) $\sum_{i=1}^{k} v_i^{\text{NDCG}_k} = 1$. b) $\phi_{SLAM}^{v^{\text{NDCG}_k}}(s, R)$ has same structure as $\phi_{SLAM}^{v^{\text{NDCG}}}(s, R)$ but with different weights. Hence structure of z_t remains same but with weights of v^{NDCG_k}.

Hence, 1st bound on gradient of z_t in proof of Lemma 5 remains same. For the 2nd bound on gradient of z_t, the crucial thing that changes is that $\sum_{i=1}^{m} v_i^t \leq k v_{t,max} v_{t,i'} c_{i'}^t$, with the new definitions of $v_{t,max}$ according to v^{NDCG_k}. This implies $2R_X \sum v_{t,i} \leq 2R_X\, k\, v_{t,max}\, \phi^{v_t}(s_t^{w_t}, R_t)$.

□

2.6 Minimax Bound on Cumulative NDCG/AP Induced Loss

2.6.1 Lower Bound

The following theorem gives a lower bound on the cumulative NDCG/AP induced loss, achievable by any deterministic online algorithm

Theorem 9. *Suppose the number of documents per query is restricted to two and relevance vectors are restricted to being binary graded. Let* $\mathcal{X} = \{X \in \mathbb{R}^{2\times d} | \ \|X_{j:}\|_2 \leq R_X\}$ *and* $\frac{R_X^2}{\gamma^2} \leq d$*. Then, for any deterministic online algorithm, there exists a ranking dataset which is separable by margin* γ *(Eq. 2.21), on which the algorithm suffers* $\Omega(\lfloor \frac{R_X^2}{\gamma^2} \rfloor)$ *cumulative NDCG/AP induced loss.*

Proof. Let $T = \lfloor \frac{R_X^2}{\gamma^2} \rfloor$. Since $\frac{R_X^2}{\gamma^2} \leq d$, hence $T \leq d$ and $T\gamma^2 \leq R_X^2$. Let a ranking dataset consist of the following T document matrices: document feature matrix $X_i = [R_X \cdot e_i; -R_X \cdot e_i]^\top$, where e_i is the unit vector of length d with 1 in ith coordinate and 0 in others.

Given an algorithm $\mathcal{A}$, let the relevance vectors for the dataset be as following: for matrix X_i, if $\mathcal{A}$ ranks 1st document above 2nd, then $R_{i,1} = 1, R_{i,2} = 0$, else, the relevances are reversed.

For the above dataset, $\mathcal{A}$ will make a ranking mistake in each round and NDCG/AP induced loss per round will be $\Omega(1)$. Therefore, the cumulative NDCG/AP induced loss will be $\Omega(T) = \Omega(\lfloor \frac{R_X^2}{\gamma^2} \rfloor)$.

Now, it remains to be shown that the dataset is actually separable with margin γ by a ranking function with unit norm parameter.

Let a ranking function parameter $w \in \mathbb{R}^d$ be defined as follows: $\forall\ i \in [T]$, the i th component of w is

$$w_i = \begin{cases} \frac{\gamma}{2 \cdot R_X} & \text{if } R_{i,1} > R_{i,2} \\ \frac{-\gamma}{2 \cdot R_X} & \text{otherwise} \end{cases}$$

For $i \notin [T]$ but $i \in [d]$, $w_i = 0$.

The unit norm condition holds because $\|w\|_2^2 = \frac{T\gamma^2}{4 \cdot R_X^2} \leq 1$.

Let $X_{i,j:}$ indicate jth row of matrix X_i. The margin condition holds as follows: $\forall\ i \in [T]$, if $R_{i,1} > R_{i,2}$, then $X_{i,1:} \cdot w - X_{i,2:} \cdot w = \frac{\gamma}{2} - \frac{-\gamma}{2} = \gamma$, else if $R_{i,1} < R_{i,2}$, then $X_{i,2:} \cdot w - X_{i,1:} \cdot w = \frac{\gamma}{2} - \frac{-\gamma}{2} = \gamma$.

□

2.6.2 Algorithm Achieving Lower Bound

We will show that the lower bound established in the previous section is actually the minimax bound, achievable by another perceptron type algorithm. Thus, Algorithm 2 is sub-optimal in terms of the bound achieved, since it has a dependence on number of documents per query.

Our algorithm is inspired by the work of *Crammer and Singer* (2002). Following their work, we define a new surrogate via a constrained optimization problem for ranking as follows:

$$\begin{aligned} \phi_C(s, R) = \min \delta \\ \text{s.t.} \quad \delta \geq 0, \quad s_i + \delta \geq \Delta + s_j, \text{ if } R_i > R_j, \ \ \forall\, i, j. \end{aligned} \tag{2.25}$$

The above constrained optimization problem can be recast as a hinge-like convex surrogate:

$$\phi_C(s, R) = \max_{i \in [m]} \max_{j \in [m]} \mathbf{1}\left[R(i) > R(j)\right] \left(1 + s_j - s_i\right)_+ . \tag{2.26}$$

The key difference between the above surrogate and the previously proposed SLAM family of surrogates is that the above surrogate does not adapt to different ranking measures. It also does not exhibit the listwise property since it treats an incorrectly ranked pair in a uniform way independent of where they are placed by the ranking induced by s.

Similar to Algorithm 2, we define a sequence of losses $f_t(w)$, defined on a sequence of data $\{X_t, R_t\}_{t \geq 1}$,as follows:

$$f_t(w) = \begin{cases} \phi_C(s_t^w, R_t) & \text{if RML}(s_t^{w_t}, R_t) \neq 0 \\ 0 & \text{if RML}(s_t^{w_t}, R_t) = 0 \end{cases} \tag{2.27}$$

Here, $s_t^w = X_t w$ and w_t is the parameter produced by Algorithm 3 at time t, with the adaptive adversary being influenced by the move of perceptron. Note that $f_t(w_t) \geq \text{RML}(s_t^{w_t}, R_t)$,

since, $f_t(w)$ is always non-negative and if $\text{RML}(s_t^{w_t}, R_t) > 0$, there is at least one pair of documents whose scores do not agree with their relevances. At that point, the surrogate value becomes greater than 1.

During a mistake round, the gradient z is calculated as follows: let i^*, j^* be any pair of indices that achieve the max in Eq. 2.26. Then,

$$z = \nabla_w \phi_C(s^w, R) = X^\top \{(-e_{i^*} + e_{j^*}) \mathbf{1}\left[R(i^*) > R(j^*)\right] \mathbf{1}\left[1 + s_{j^*} - s_{i^*} \geq 0\right]\}. \quad (2.28)$$

Note that if there are multiple index pairs achieving the max, then an arbitrary subgradient can be written as a convex combination of subgradients computed using each of the pairs.

Algorithm 3 New Perceptron Algorithm Achieving Lower Bound

Learning rate $\eta > 0$, $w_1 = \mathbf{0} \in \mathbb{R}^d$.
For $t = 1$ to T
  Receive X_t (document list for query q_t).
  Set $s_t^{w_t} = X_t w_t$, predicted ranking output= $\text{argsort}(s_t^{w_t})$.
  Receive R_t
  If $\text{RML}(s_t^{w_t}, R_t) \neq 0$ // Note: $\text{RML}(s_t^{w_t}, R_t) = \text{RML}(\text{argsort}(s_t^{w_t}), R_t)$
    $w_{t+1} = w_t - \eta z_t$ // z_t is defined in Eq. (2.28)
  else
    $w_{t+1} = w_t$
End For

We have the following loss bound for Algorithm 3.

Theorem 10. *Suppose Algorithm 3 receives a sequence of instances $(X_1, R_1), \ldots, (X_T, R_T)$. Let R_X be the bound on the maximum ℓ_2 norm of the feature vectors representing the documents and $f_t(w)$ be as defined in Eq. 2.27. Then the following inequality holds, after optimizing over learning rate η, $\forall\ w \in \mathbb{R}^d$:*

$$\sum_{t=1}^{T} \textit{RML}(s_t^{w_t}, R_t) \leq \sum_{t=1}^{T} f_t(w) + 2\|w\|_2 R_X \sqrt{\sum_{t=1}^{T} f_t(w)} + 4\|w\|_2^2 R_X^2 \,. \quad (2.29)$$

In the special case where there exists w s.t. $f_t(w) = 0$, $\forall\, t$, we have

$$\sum_{t=1}^{T} \textit{RML}(s_t^{w_t}, R_t) \;\leq\; 4\|w\|_2^2 R_X^2. \tag{2.30}$$

Proof. We first bound the ℓ_2 norm of the gradient. From Eq. 2.28, we have:

1st bound for z_t:

$$\|z_t\|_2 \leq \|X_t^\top\|_{1\to 2} \|\{(-e_{i^*} + e_{j^*})\mathbf{1}\left[R(i^*) > R(j^*)\right]\mathbf{1}\left[1 + s_{j^*} - s_{i^*} \geq 0\right]\}\|_1 \leq 2R_X.$$

2nd bound for z_t:

On a mistake round, since there exists at least 1 pair of documents, whose scores and relevance levels are discordant. Hence, $\phi_C(s^w, R) > 1$. Hence, $\|z_t\|_2 \leq 2R_X \leq 2R_X \phi_C(s_t^{w_t}, R_t)$.

Thus, $\|z_t\|_2^2 \leq 4R_X^2 \phi_C(s_t^{w_t}, R_t)$. Since $\|z_t\|_2 = 0$ on non-mistake round, we finally have:

$\|z_t\|_2^2 \leq 4R_X^2 f_t(w_t),\ \forall\, t.$

The proof then follows as previous: by plugging in expression for $\|z_t\|^2$ in OGD equation (Prop. OGD Regret), optimizing over η, using the algebraic trick: $x - b\sqrt{x} - c \leq 0 \implies x \leq b^2 + c + b\sqrt{c}$ and then using the inequality $f_t(w_t) \geq \text{RML}(s_t^{w_t}, R_t)$. □

As before, we can immediately derive a margin based bound.

Corollary 11. *If the margin condition* (2.21) *holds, then total loss, for both NDCG and AP induced loss, is upper bounded by $\frac{4R_X^2}{\gamma^2}$, a bound independent of the number of instances in the online sequence.*

Proof. Proof is similar to that of Corollary 7. □

Comparison of Algorithm 2 and Algorithm 3: Both of our proposed perceptron-like algorithms can be thought of analogues of the classic perceptron in the learning to rank

setting. Algorithm 3 achieves the minimax optimal bound on separable datasets, unlike Algorithm 2, whose bound scales with number of documents per query. However, Algorithm 3 operates on a surrogate (Eq 2.26) which is not listwise in nature, even though it forms an upper bound on the listwise ranking measures. To emphasize, the surrogate does not differentially weigh between errors at different points of the ranked list, which is an important property of popular surrogates in learning to rank. As our empirical results show (Sec. 3.4), on commercial datasets which are not separable, Algorithm 3 has significantly worse performance than Algorithm 2.

2.7 Related Work in Perceptron for Ranking

There exist a number of papers in the literature dealing with perceptron in the context of ranking. We will compare and contrast our work with existing work, paying special attention to the papers whose setting come closest to ours.

First, we would like to point out that, to the best of our knowledge, there is no work that establishes a number of documents independent bound for NDCG, cut-off at the top k position (Theorem 8). Moreover, we believe our work, for the first time, formally establishes minimax bound, achievable by any deterministic online algorithm, in the learning to rank setting, under the assumption of separability.

Crammer and Singer (2001) were one of the first to introduce perceptron in ranking. The setting as well as results of their perceptron are quite different from ours. Their paper assumes there is a fixed set of ranks $\{1, 2, \ldots, k\}$. An instance is a vector of the form $x \in \mathbb{R}^d$ and the supervision is one of the k ranks. The perceptron has to learn the correct ranking of x, with the loss being 1 if correct rank is not predicted. The paper does not deal with query-documents list and does not consider learning to rank measures like NDCG/AP.

The results of *Wang et al.* (2015) have some similarity to ours. Their paper introduces algorithms for online learning to rank, but does not claim to have any "perceptron type" results. However, their main theorem (Theorem 2) has a perceptron bound flavor to it, where

the cumulative NDCG/AP losses are upper bounded by cumulative surrogate loss and a constant. The major differences with our results are these: *Wang et al.* (2015) consider a different instance/supervision setting and consequently have a different surrogate loss. It is assumed that for each query q, only a pair of documents (x_i, x_j) are received at each online round, with the supervision being $\{+1, -1\}$, depending on whether x_i is more/less relevant than x_j. The surrogate loss is defined at pair of documents level, and not at a query-document matrix level. Moreover, there is no equivalent result to our Theorem 8, neither is any kind of minimax bound established.

The recent work of *Jain et al.* (2015) also contains results similar to ours. One the one hand their predtron algorithm is more general. But on the other hand, the bound achieved by predtron, applied to the ranking case, has a scaling factor $O(m^5)$, significantly worse than our linear scaling. Moreover, it does not have the NDCG_k bounds scaling as a function of k proved anywhere.

There are other, less related papers; all of which deal with perceptron in ranking, in some form or the other. *Ni and Huang* (2008) introduce the concept of margin in a particular setting, with corresponding perceptron bounds. However, their paper does not deal with query-document matrices, nor NDCG/AP induced losses. The works of *Elsas et al.* (2008) and *Harrington* (2003) introduce online perceptron based ranking algorithms, but do not establish theoretical results. *Shen and Joshi* (2005) give a perceptron type algorithm with a theoretical guarantee, but in their paper, the supervision is in form of full rankings (instead of relevance vectors).

2.8 Experiments

We conducted experiments on a simulated dataset and three large scale industrial benchmark datasets. Our results demonstrate the following:

- We simulated a margin γ separable dataset. On that dataset, the two algorithms

(Algorithm 2 and Algorithm 3) ranks all but a finite number of instances correctly, which agrees with our theoretical prediction.

- On three commercial datasets, which are not separable, Algorithm 2 shows competitive performance with a strong baseline algorithm, indicating its practical usefulness. Algorithm 3 performs quite poorly on two of the datasets, indicating that despite minimax optimality under margin separability, it has limited practical usefulness.

Baseline Algorithm: We compared our algorithms with the online version of the popular ListNet ranking algorithm (*Cao et al.*, 2007a). ListNet is not only one of the most cited ranking algorithms (over 800 citations according to Google Scholar), but also one of the most validated algorithms (*Tax et al.*, 2015). We conducted online gradient descent on the cross-entropy convex surrogate of ListNet to learn a ranking algorithm in an online manner. While there exists ranking algorithms which have demonstrated better empirical performance than ListNet, they are generally based on non-convex surrogates with non-linear ranking functions. These algorithms cannot be converted in a straight forward way (or not at all) into online algorithms which learn from streaming data. We also did not compare our algorithms with other perceptron algorithms since they do not usually have similar setting to ours and would require modifications. *We emphasize that our objective is not simply to add one more ranking algorithms to the huge variety that already exists. Our experiments on real data are to show that Algorithm 2 has competitive performance and has a major advantage over Algorithm 3, due to the difference in the nature of surrogates being used for the two algorithms*.

Experimental Setting: For all datasets, we report average NDCG_{10} over a time horizon. Average NDCG_{10} at iteration t is the cumulative NDCG_{10} up to iteration t, divided by t. We remind that at each iteration t, a document matrix is ranked by the algorithm, with the performance (according to NDCG_{10}) measured against the true relevance vector corresponding to the document matrix. For all the algorithms, the corresponding best learning rate η was fixed after conducting experiments with multiple different rates and observing

the best time averaged NDCG_{10} over a fixed time interval. We did not conduct any experiments with AP since the real datasets have multi-graded relevance and NDCG is a suitable measure for such datasets.

Simulated Dataset: We simulated a margin separable dataset (Eq. (2.21)). Each query had $m = 20$ documents, each document represented by 20 dimensional feature vector, and five different relevance level $\{4, 3, 2, 1, 0\}$, with relevances distributed uniformly over the documents. The feature vectors of equivalent documents (i.e., documents with same relevance level) were generated from a Gaussian distribution, with documents of different relevance levels generated from different Gaussian distribution. A 20 dimensional unit norm ranker was generated from a Gaussian distribution, which induced separability with margin. Fig. 5.1 compares performance of Algorithm 2, Algorithm 3 and online ListNet. The NDCG_{10} values of the perceptron type algorithms rapidly converge to 1, validating their finite cumulative loss property. To re-iterate, since for separable datasets, cumulative NDCG induced loss is bounded by constant, hence, the time averaged NDCG should rapidly converge to 1. The OGD algorithm for ListNet has only a regret guarantee of $O(\sqrt{t})$; hence the time averaged regret converges at rate $O(\frac{1}{\sqrt{t}})$, i.e., its convergence is significantly slower than the perceptron-like algorithms.

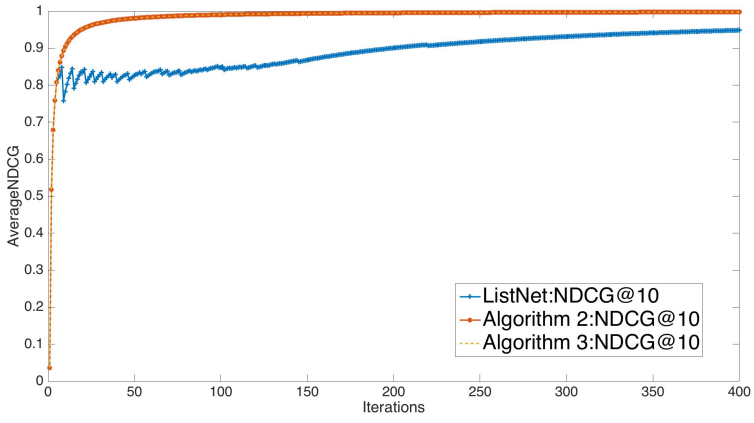


Figure 2.1: Time averaged NDCG_{10} for Algorithm 2, Algorithm 3 and ListNet, for separable dataset. The two perceptron-like algorithms have imperceptible difference.

Commercial Datasets: We chose three large scale ranking datasets released by the industry to analyze the performance of our algorithms. MSLR-WEB10K (*Liu et al.*, 2007a) is the dataset published by Microsoft's Bing team, consisting of $10,000$ unique queries, with feature dimension of size 245 and 5 distinct relevance levels. Yahoo Learning to Rank Challenge dataset (*Chapelle and Chang*, 2011a) consists of 19,944 unique queries, with feature dimension of size 700 and 5 distinct relevance levels. Yandex, Russia's biggest search engine, published a dataset (link to the dataset given in the work of *Chapelle and Chang* (2011a)) consisting of 9124 queries, with feature dimension of size 245 and 5 distinct relevance levels. Algorithm 2 performs slightly better than ListNet on MSLR-WEB dataset (average NDCG@10 over last ten iterations= 0.25 vs 0.22), performs slightly worse on Yahoo dataset (average NDCG@10 over last ten iterations= 0.75 vs 0.74) and has overlapping performance on Yandex dataset. The experiments validate that our proposed perceptron type algorithm (Algorithm 2) has competitive performance compared to online ListNet on real ranking datasets, even though it does not achieve the theoretical lower bound. Algorithm 3 performs quite poorly on both Yandex and Yahoo datasets. One possible reason for the poor performance is that the underlying surrogate (Eq 2.26) is not listwise in nature. It does not put more emphasis on errors at the top and hence, is not very suitable for a listwise ranking measure like NDCG, even though it achieves the theoretical lower bound on separable datasets.

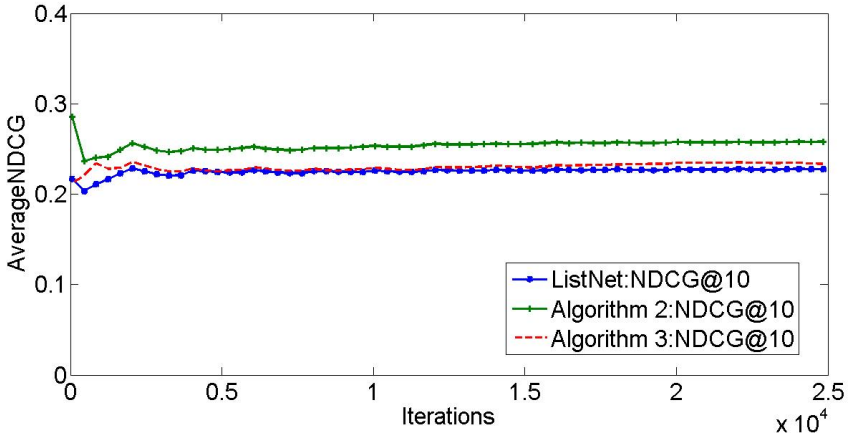
2.9 Conclusion

We proposed two perceptron-like algorithms for learning to rank, as analogues of the perceptron for classification algorithm. We showed how, under assumption of separability (i.e., existence of a perfect ranker), the cumulative NDCG/AP induced loss is bounded by a constant. The first algorithm operates on a listwise, large margin family of surrogates, which are adaptable to NDCG and AP. The second algorithm is based on another large margin surrogate, which does not have the listwise property. We also proved a lower bound

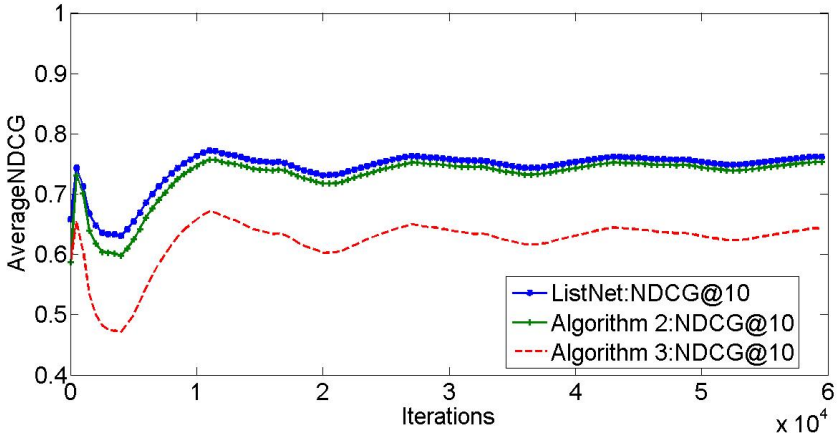
on cumulative NDCG/AP loss under a separability condition and showed that it is the minimax bound, since our second algorithm achieves the bound. We conducted experiments on simulated and commercial datasets to corroborate our theoretical results.

An important aspect of perceptron type algorithms is that the ranking function is updated only on a mistake round. Since non-linear ranking functions are generally have better performance than linear ranking functions, an online algorithm learning a flexible non-linear kernel ranking function would be very useful in practice. We highlight how perceptron's "update only on mistake round" aspect can prove to be powerful when learning a non-linear kernel ranking function. Since the score of each document is obtained via inner product of ranking parameter w and feature representation of document x, this can be easily kernelized to learn a range of non-linear ranking functions. However, the inherent difficulty of applying OGD to a convex ranking surrogate with kernel function is that at each update step, the document list (X matrix) will need to be stored in memory. For moderately large dataset, this soon becomes a practical impossibility. One way of bypassing the problem is to approximately represent the kernel function via an explicit feature projection (*Rahimi and Recht*, 2007; *Le et al.*, 2013). However, even for moderate length features (like 136 for MSWEB10K), the projection dimension becomes too high for efficient computation. Another technique is to have a finite budget for storing document matrices and discard carefully chosen members from the budget when budget capacity is exceeded. This budget extension has been studied for perceptron in classification (*Dekel et al.*, 2008; *Cavallanti et al.*, 2007). The fact that perceptron updates are only on mistake rounds leads to strong theoretical bounds on target loss. For OGD on general convex surrogates, the fact that function update happens on every round leads to inherent difficulties when using their kernelized versions (*Zhao et al.*, 2012) (the theoretical guarantees on the target loss are not as strong as in the kernelized perceptron on a budget case). The results presented in this chapter open up a fruitful direction for further research: namely, to extend the perceptron algorithm to non-linear ranking functions by using kernels and establishing

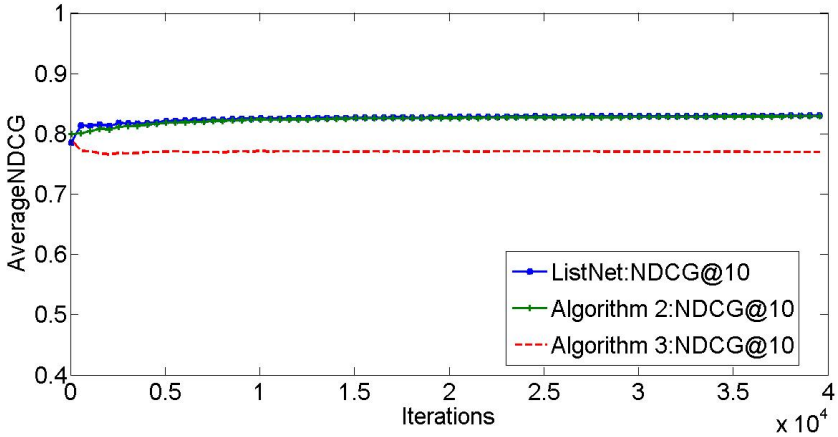
theoretical performance bounds in the presence of a memory budget.



(a) MSLR-WEB10K



(b) Yahoo



(c) Yandex

Figure 2.2: Time averaged $NDCG_{10}$ for Algorithm 2, Algorithm 3 and ListNet for 3 commercial datasets. While Algorithm 2 has competitive performance to ListNet, Algorithm 3 performs quite poorly on Yahoo and Yandex datasets.

CHAPTER III

Online Learning to Rank with Feedback on Top Ranked Items

3.1 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 learn 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 such that it doest not even allow for 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 on the relevance of the items to the user. The relevance of the items, reflecting varying user preferences, can be considered encoded as relevance vectors. The system's objective is to learn from the feedback it receives and update its ranker over time, to satisfy majority of the users. 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 $k << m$ (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 *economic and user-burden* constraints and *privacy* concerns.

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 can

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 suggestions. 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 suggestions.

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 a certain product. The app develops features representing the categories, and thus, the interaction with the users happen 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 activities, 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. Throughout our work, we do not make any *stochastic* assumption on the relevance vector generation process or the feature generation process (in the second problem setting). Thus, our work is in purely adversarial setting, with the adversary considered to be *oblivious* to the learner's strategies. We discuss the two problem settings in greater details, separately.

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 highly restricted feedback. 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*, 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 property of the loss and feedback functions called *observability* (*Bartok et al.*, 2014; *Foster and Rakhlin*, 2012). Observability is of two kinds: *local* and *global*. 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.*, 2007a), 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 thus impractical in our case 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*, 2004a), 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 under 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 vector. Surprisingly, we are able to translate this intuitive hurdle into a provable impossibility.

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

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 bandit problem, while non-contextual multi-armed bandit has received a lot of attention, the authors in (*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, in an online manner, with explicit but restricted feedback. 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 ad-

versary 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.*, 2007b) 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.

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.*, 2007b) 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*, 2001b) 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*, 2006), and vice versa (*Bartók and Szepesvári*, 2012; *Gentile and Orabona*, 2014).

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

We divide the rest of the paper into two parts. Sec.3.2 and its sub-sections detail the notations, definitions and technicalities associated with online ranking with restricted feedback in the non-contextual setting. Sec.3.3 and its subsections detail the notations, definitions and technicalities associated with online ranking with restricted feedback in the contextual setting. Sec.3.4 demonstrates the performance of our algorithms on simulated and commercial datasets. Sec.3.5 discusses open questions and future directions of research.

3.2 Online Ranking with Restricted Feedback- Non Contextual Setting

This section will detail the first part of our work and will be self contained. All proofs not in main text are in the appendix.

3.2.1 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. *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 2^m outcomes (when relevance levels are restricted to binary) or from n^m outcomes (when there are n relevance levels, $n > 2$). 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 ranked object, i.e., $R_t(\sigma_t(1))$, is 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:

$$\mathbb{E}_{\sigma_1,\ldots,\sigma_T}\left[\sum_{t=1}^{T} RL(\sigma_t, R_t)\right] - \min_{\sigma} \sum_{t=1}^{T} RL(\sigma, R_t). \tag{3.1}$$

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.

3.2.2 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:

$$\sum_{t=1}^{T} PL(\sigma_t, R_t) - \sum_{t=1}^{T} PL(\sigma, R_t) = \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) - \sum_{t=1}^{T} SumLoss(\sigma, R_t). \quad (3.2)$$

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.

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: AP 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.

3.2.3 Summary of Results

We summarize our main results here before delving into technical details. **Result 1**: The minimax regret under DCG and PairwiseLoss (and hence SumLoss), restricted to top 1 ($k = 1$) feedback, is $\Theta(T^{2/3})$. The minimax regret under Precision@n, for top k feedback, with $1 \leq k < n$, is unknown.

Result 2: We propose a generic algorithm which achieves regret of $O(T^{2/3})$, with top k feedback ($k \geq 1$) under DCG, PairwiseLoss and Precision@n. The generic algorithm uses a reduction to a full information algorithm. We instantiate a particular full information algorithm, to get an efficient algorithm, which has a running time $O(m \log m)$ and regret $O(poly(m)T^{2/3})$.

Result 3: The minimax regret for any of the normalized versions – NDCG, AP and AUC – , restricted to top 1 feedback, is $\Theta(T)$. Thus, there is no algorithm that guarantees *sub-linear* regret for the normalized measures.

Result 4: The regret rates, as a function of T, both for DCG and NDCG, does not change when we consider non-binary, multi-graded relevance vectors.

3.2.4 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 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

Table 3.1: 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

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 3.1 and 3.2 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

Table 3.2: 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

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.

3.2.5 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*.

3.2.5.1 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).

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 12. *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

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

Thus, we have

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

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 is 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, the 2nd column of $(S^{\top}_{\tilde{\sigma}_{j(a)}})$, i.e., $(S^{\top}_{\tilde{\sigma}_{j(a)}})_{:,2}$ is $[(S^{\top}_{\tilde{\sigma}_{j(a)}})_{R_1,2}, (S^{\top}_{\tilde{\sigma}_{j(a)}})_{R_2,2}, \ldots, (S^{\top}_{\tilde{\sigma}_{j(a)}})_{R_{2^m},2}]$ □

3.2.5.2 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 $\ell_i - \ell_j \in \oplus_{k \in N^+_{i,j}} Col(S^{\top}_k)$. 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 13. *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. By dot product rule between 2 vectors, $l_i \cdot p$ 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 14. *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 13, 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 13 and 14 lead to following result.

Theorem 15. *The local observability condition, as per Definition 5, fails w.r.t. loss matrix L and feedback matrix H defined for SumLoss, already at $m = 3$.*

3.2.6 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 15 with Theorem 4

of *Bartok et al.* (2014).

Corollary 16. *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 12), combined with Theorem 3.1 in *Cesa-Bianchi* (2006), gives an algorithm (inspired by the algorithm originally given in *Piccolboni and Schindelhauer* (2001a)) obtaining $O(T^{2/3})$ regret.

Corollary 17. *The algorithm in Figure 1 of Cesa-Bianchi (2006) achieves* $O(T^{2/3})$ *regret bound for SumLoss.*

However, the algorithm in *Cesa-Bianchi* (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 result 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 12, 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. It makes the game at least as easy as with top 1 feedback (for example, the learner can just throw away relevance feedback on items ranked 2nd onwards). So, with top k feedback, with 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 bandit game (disregarding computational complexity), which has regret rate of $O(T^{1/2})$ and hence locally observable.

3.2.7 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 $O(poly(m)\ T^{2/3})$.

3.2.7.1 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: for any online game in an adversarial setting played over T rounds, if the loss of each action is known at end of each round (full information), the algorithm should have an expected regret rate of $O(T^{1/2})$, where the regret is the difference between cumulative loss of the algorithm and

cumulative loss of best action in hindsight. For the generic full information algorithm, we ignore computational complexity and dependence on parameters, other than T, in the regret rate.

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 4 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 18. *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.,* $\{R_{i_j}(k * (j - 1) + 1), R_{i_j}(k * (j - 1) + 2), \ldots, R_{i_j}(k * j)\}$*, becomes 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.* $\hat{R}_t = [R_{i_j}(k*(j-1)+1), R_{i_j}(k*(j-1)+2), \ldots, R_{i_j}(k*j)]_{\{j=1,2,\ldots,\lceil m/k \rceil\}}$ *is an unbiased estimator of* $R^{avg}_{1:t}$*.*

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

We have the following regret bound, obtained from application of Algrotihm 4 on Sum-Loss with top k feedback.

Theorem 19. *The expected regret under SumLoss, obtained by applying Algorithm 4, 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*

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] - \min_{\sigma} \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) \leq C^I \lceil m/k \rceil K + C\frac{T}{\sqrt{K}} \quad (3.3)$$

Optimizing over block size K, we get $K = C^{II}\dfrac{T^{2/3}}{\lceil m/k \rceil^{2/3}}$ and final regret bound as:

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] - \min_{\sigma} \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) \leq C^{III} \lceil m/k \rceil^{1/3} T^{2/3} \quad (3.4)$$

where C, C^I, C^{II} and C^{III} are parameters (independent of T) depending on the specific full information algorithm used.

Algorithm 4 RankingwithTop-kFeedback(RTop-kF)- Non Contextual

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

[1] For e.g., 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.

3.2.7.2 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 modifications are needed in Algorithm 4 to implement FTPL as the full information algorithm:

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 $M(y) = \underset{\sigma}{\operatorname{argmin}}\ \sigma^{-1} \cdot y$.

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 have the following corollary:

Corollary 20. *The expected regret of SumLoss, obtained by applying Algorithm 4, 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:*

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] - \min_{\sigma} \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) \leq O(m^{7/3}\lceil m/k \rceil^{1/3} T^{2/3}). \quad (3.5)$$

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

3.2.8 Regret Bounds for PairwiseLoss, DCG and Precision@n

PairwiseLoss: As we saw in Eq. 3.2, the regret of SumLoss is same as regret of PairwiseLoss. Thus, SumLoss in Corollary 20 can be replaced by PairwiseLoss to get exactly same result.

DCG: All the results of SumLoss can be extended to DCG (see appendix). 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.3.2.2) 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 4 can be applied (with slight variation), with FTPL full information algorithm and top k feedback, to achieve regret of $O(T^{2/3})$. 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 *exploita-*

tion round, the selection of σ_t remains exactly same as in SumLoss, i.e., $\sigma_t = M(\hat{s}_{i-1} + p_t)$ where $M(y) = \underset{\sigma}{\operatorname{argmin}}\ \sigma^{-1} \cdot y$. This is because $\underset{\sigma}{\operatorname{argmax}}\ f(\sigma) \cdot y = \underset{\sigma}{\operatorname{argmin}}\ \sigma^{-1} \cdot y$, 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 21. *The expected regret of DCG, obtained by applying Algorithm 4, with FTPL full information algorithm and feedback on top k ranked items at end of each round ($k \geq 1$), and $K = O\left(\frac{m^{1/3}T^{2/3}}{\lceil m/k \rceil^{2/3}}\right)$, $\epsilon = O(\frac{1}{(2^n-1)^2\sqrt{mK}})$, is:*

$$\max_{\sigma} \sum_{t=1}^{T} DCG(\sigma_t, R_t) - \mathbb{E}\left[\sum_{t=1}^{T} DCG(\sigma_t, R_t)\right] \leq O((2^n - 1)m^{4/3}\lceil m/k \rceil^{1/3} T^{2/3}). \quad (3.6)$$

Assuming that $\lceil m/k \rceil \sim m/k$, the regret rate 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 4 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 $M(y) = \underset{\sigma}{\operatorname{argmin}}\ \sigma^{-1} \cdot y$.

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 14 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 22. *The expected regret of Precision@n, obtained by applying Algorithm 4, 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:*

$$\max_{\sigma} \sum_{t=1}^{T} Precision@n(\sigma_t, R_t) - \mathbb{E}\left[\sum_{t=1}^{T} Precision@n(\sigma_t, R_t)\right] \leq O(n\, m^{1/3} \lceil m/k \rceil^{1/3} T^{2/3}). \tag{3.7}$$

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

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

As stated in Sec. 3.2.2, 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 23. *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 24. *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.*

3.3 Online Ranking with Restricted Feedback- Contextual Setting

This section will detail the second part of our work and will be self contained. All proofs not in the main text are in the appendix.

3.3.1 Notation and Preliminaries

We introduce notations, which are in addition to the those we introduced in Sec.3.2.1. 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.

3.3.2 Problem Setting and Learning to Rank Algorithm

Formal problem setting: We formalize the problem as a game being played between a learner and an *oblivious* adversary over T rounds (i.e., an adversary who generates moves without knowledge of the learner's algorithm). 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.

Relation between feedback and structure of surrogates: Algorithm 5 is our general algorithm for learning a ranking function, online, from partial feedback. The key step in Algorithm 5 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

Algorithm 5 Ranking with Top-k Feedback (RTop-kF)- Contextual

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

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 $\mathbb{P}$ 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,

$$p(j_1, \ldots, j_k) = \sum_{\pi \in S_m} \mathbb{P}(\pi)\mathbb{1}(\pi(1) = j_1, \ldots, \pi(k) = j_k). \tag{3.8}$$

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

Lemma 25. *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:

$$F(x) = \sum_{(i_1,i_2,\ldots,i_\ell)\in\ {}^mP_\ell} h_{i_1,i_2,\ldots,i_\ell}(x_{i_1}, x_{i_2}, \ldots, x_{i_\ell}) \tag{3.9}$$

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 3.9 , with $\ell = k$, an unbiased estimator of $F(x)$, based on $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$, is,

$$g(\sigma,x_{\sigma(1)}, \ldots, x_{\sigma(k)}) = \frac{\sum_{(j_1,j_2,\ldots,j_k)\in S_k} h_{\sigma(j_1),\ldots,\sigma(j_k)}(x_{\sigma(j_1)}, \ldots, x_{\sigma(j_k)})}{\sum_{(j_1,\ldots,j_k)\in S_k} p(\sigma(j_1), \ldots, \sigma(j_k))} \tag{3.10}$$

where S_k is the set of $k!$ permutations of $[k]$ and $p(\sigma(1), \ldots, \sigma(k))$ is as in Eq 3.8 .

Illustrative Examples: We provide simple examples to concretely illustrate the abstract functions in Lemma 25. 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: $g(\sigma, x_{\sigma(1)}) = \frac{h_{\sigma(1)}(x_{\sigma(1)})}{p(\sigma(1))}$.

3.3.3 Unbiased Estimators of Gradients of Surrogates

Algorithm 5 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. 3.8 with $\mathbb{P} = \mathbb{P}_t$ where $\mathbb{P}_t$ is the distribution in round t in Algorithm 5.

3.3.3.1 Convex Surrogates

Pointwise Method: We will construct the unbiased estimator of the gradient of squared loss (*Cossock and Zhang*, 2006b): $\phi_{sq}(s, R) = \|s - R\|_2^2$. 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 25, **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 $\nabla_s \phi_{svm}(s, R) = \sum_{i \neq j=1}^m \mathbb{1}(R(i) > R(j)) \mathbb{1}(1 + s(j) > s(i))(e_j - e_i) \in \mathbb{R}^m$. Since s is a known quantity, from Lemma 25, we can construct $F(R)$ as follows: $F(R) = F_s(R) = \sum_{i \neq j=1}^m h_{s,i,j}(R(i), R(j))$, where $h_{s,i,j}(R(i), R(j)) = \mathbb{1}(R(i) > R(j)) \mathbb{1}(1 + s(j) > s(i))(e_j - e_i)$. Since $F_s(R)$ is decomposable over 2 coor-

dinates 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 25 shows that it is the minimum information required to construct the unbiased estimator.

Lemma 26. *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.*, 2007b) 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[2]. 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 27. *The gradient of ListNet surrogate $\phi_{LN}(s, R)$ cannot be decomposed over k, for $k = 1, 2$, coordinates of R at a time.*

[2]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.

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 25, this means that if we want to run Alg. 5 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 *unnormalized* 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 25, 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 5 with top-k feedback. For example, the class of popular listwise surrogates that are developed from structured prediction perspective (*Chapelle 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.

3.3.3.2 Non-convex Surrogate

We provide an example of a non-convex surrogate for which Alg. 5 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 25, 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:

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

3.3.4 Computational Complexity of Algorithm 5

Three of the four key steps governing the complexity of Algorithm 5, 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.

3.3.5 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:

$$\mathbb{E}\left[\sum_{t=1}^{T} \phi(X_t w_t, R_t)\right] \leq \min_{w:\|w\|_2 \leq U} \sum_{t=1}^{T} \phi(X_t w, R_t) + \frac{U^2}{2\eta} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T} \|\tilde{z}_t\|_2^2\right] \tag{3.11}$$

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 5, 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 [3]

$$\mathbb{E}\left[\sum_{t=1}^{T}\phi(\tilde{s}_t, R_t)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T}\phi(X_t w_t, R_t)\right] + O\left(\gamma T\right) \tag{3.12}$$

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 3.3.3.1 , 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 28. *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 29. *For parameter γ in Algorithm 5 , 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 $\leq$ 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*

[3]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.

than four. Then we have,

$$\mathbb{E}_t \left[\|\tilde{z}_t\|_2^2 \right] \leq \frac{C^\phi}{\gamma} \tag{3.13}$$

Plugging Eq. 3.13 and Eq. 3.12 in Eq. 3.11, 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:

Theorem 30. *For any sequence of instances and labels $(X_t, R_t)_{\{t\in[T]\}}$, applying Algorithm 5 with top-1 feedback for ϕ_{sq} and ϕ_{KL} and top-2 feedback for ϕ_{svm}, will produce the following bound:*

$$\mathbb{E}\left[\sum_{t=1}^{T} \phi(\tilde{s}_t, R_t)\right] - \min_{w:\|w\|_2\leq U} \sum_{t=1}^{T} \phi(X_t w, R_t) \leq C^\phi O\left(T^{2/3}\right) \tag{3.14}$$

where C^ϕ is a surrogate dependent function, as described in Lemma 29 , and expectation is taken over underlying randomness of the algorithm, over T rounds.

Discussion: It is known that online bandit games are special instances of partial monitoring games. For bandit online convex optimization problems with Lipschitz, convex surrogates, the best regret rate known so far, that can be achieved by an efficient algorithm, is $O(T^{3/4})$ (however, see the work of *Bubeck and Eldan* (2015) for a non-constructive $O(\log^4(T)\sqrt{T})$ bound). Surprisingly, Alg. 5, when applied in a partial monitoring setting to the Lipschitz, convex surrogates that we have listed, achieves a better regret rate than what is known in the bandit setting. Moreover, as we show subsequently, for an entire class of Lipschitz convex surrogates (subclass of NDCG calibrated surrogates), sub-linear (in T) regret is not even achievable. Thus, our work indicates that even within the class of Lipschitz, convex surrogates, regret rate achievable is dependent on the structure of surrogates; something that does not arise in bandit convex optimization.

3.3.6 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 expect 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 3.2.2.

Theorem 31. *(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*

$$s^*_\phi(\eta) = g\left(\mathbb{E}_{R\sim\eta}\left[\frac{G(\mathbf{R})}{Z(R)}\right]\right). \tag{3.15}$$

Informally, Eq. 3.15 states that $\operatorname{argsort}(s^*_\phi(\eta)) \subseteq \operatorname{argsort}(\mathbb{E}_{R\sim\eta}\left[\frac{G(\mathbf{R})}{Z(R)}\right])$ *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 32. *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* $\Omega(T)$.

Note that our result is for top 1 feedback. Minimax regret for the problem setting with top k feedback, with $k \geq 2$ remains an open question.

Proof. (*Sketch*) The proof builds on the proof of hopeless finite action partial monitoring games given by *Piccolboni and Schindelhauer* (2001b). 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 31) to explicitly construct two such probability distributions. This proves the result. Full proof is provided in the appendix. □

We note that the proof of Theorem. 3 of *Piccolboni and Schindelhauer* (2001b) 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.

3.4 Empirical Results

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

3.4.1 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 4 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 4 in conjunction with Follow-The-Perturbed-Leader (FTPL) full information algorithm, as described in Sec. 3.2.7. We note that since our work is first of its kind in the literature, we had no comparable baselines. The generic partial monitoring algorithms that do exist cannot be applied due to computational inefficiency (Sec. 3.2.6).

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 dataset. 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 relevance 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) only makes 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 5.1 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 21, the optimal regret rate is achieved by optimizing over number of blocks (hence block size), which requires prior knowledge of 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 3.2 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 5.2(a) 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.

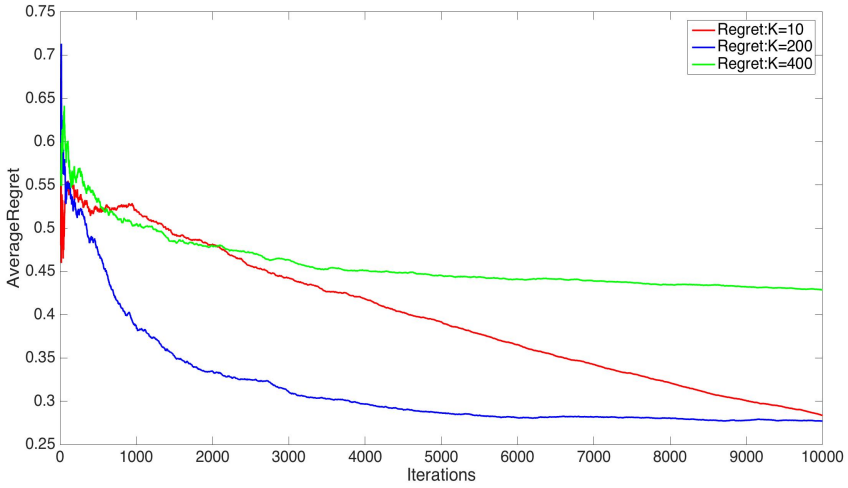


Figure 3.1: Average regret under DCG, with feedback on top ranked object, for varying block size, where block size is $\lceil T/K \rceil$.

3.4.2 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 datasets.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 of interest. The users only 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,

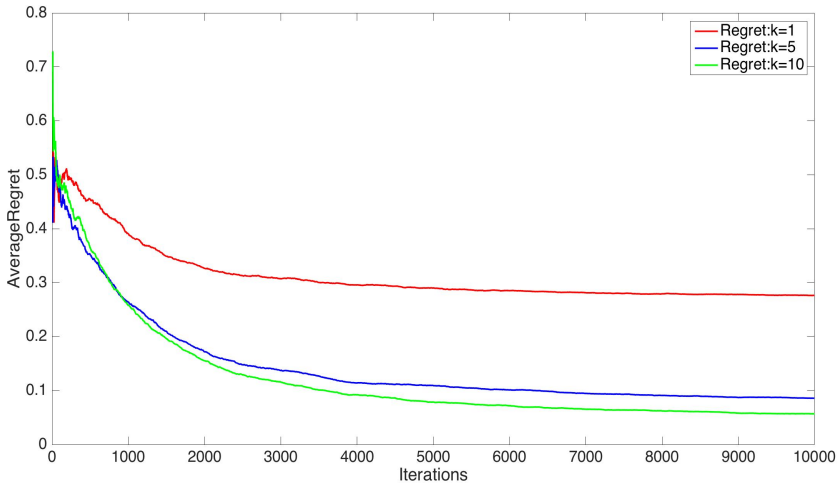


Figure 3.2: Average regret under DCG, where $K = 200$, for varying amount of feedback.

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 5, 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*

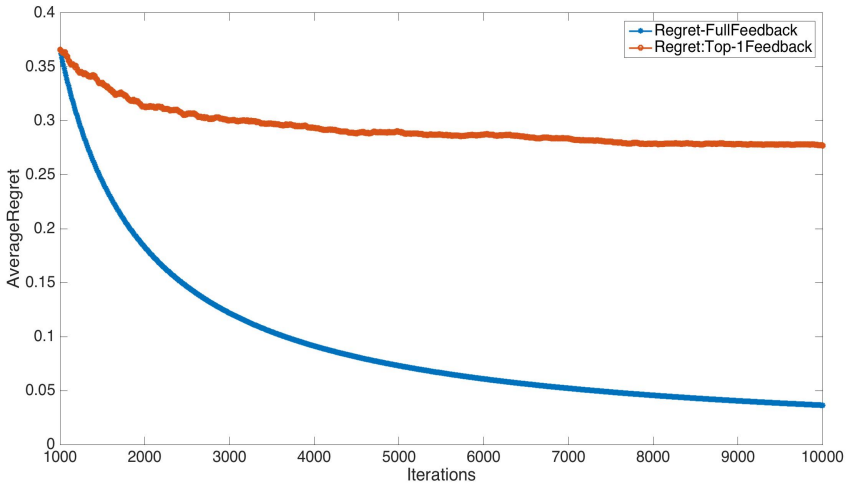


Figure 3.3: Comparison of average regret over time, for DCG, between top-1 feedback and full relevance vector feedback.

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 5, which learns from highly restricted feedback, with an algorithm which learns from full feedback and an algorithm which receives no feedback and learns nothing.*

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

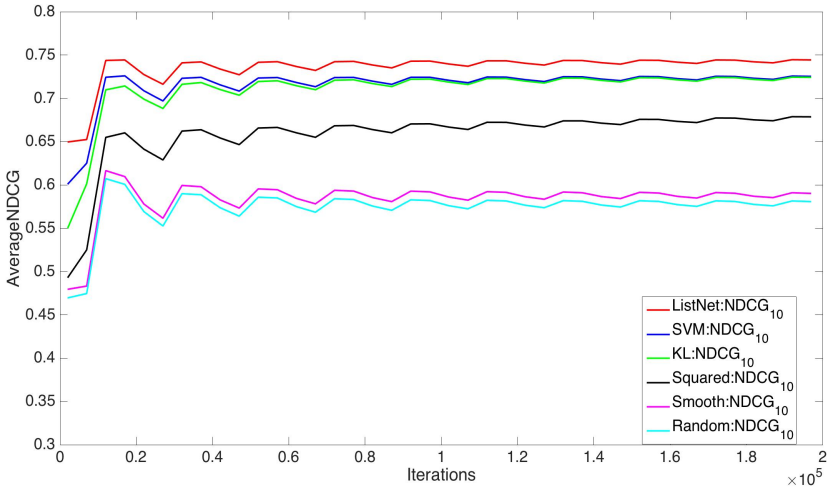


Figure 3.4: Average NDCG_{10} values of different algorithms, for Yahoo dataset. ListNet (in cyan) algorithm operates on full feedback and Random (in red) algorithm does not receive any feedback.

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 dataset multiple times). The reason for choosing different time horizons is that there are roughly double the number of queries in Yahoo dataset as compared to Yandex dataset. 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 5, 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 $\eta = O(T^{-1/2})$. 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 3.10). 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

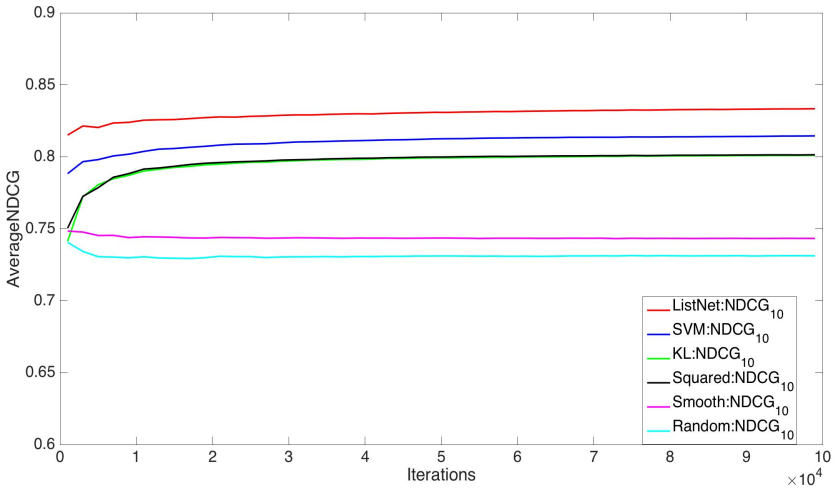


Figure 3.5: Average NDCG_{10} values of different algorithms, for Yahoo dataset. ListNet (in cyan) algorithm operates on full feedback and Random (in red) algorithm does not receive any feedback.

drawn permutation does not match the top ranked item of the permutation given by the deterministic score vector (Sec 3.3.4). 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 scaled up γ on the mismatch round by a constant, to remove the negative effect.

Results: Figure 5.2(b) and Figure 5.2(c) show that ListNet, with full information feedback at end of each round, has highest average NDCG value throughout, as expected. However, Algorithm 5, with the convex surrogates, produce competitive performance. In fact, in the Yahoo dataset, 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 dataset. *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.

3.5 Conclusion and Open Questions

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 datasets 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 4 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 5 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 25, 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 5, 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 feed-

back. What is the minimax regret for NDCG calibrated surrogates, with top k feedback, for $k > 1$?

CHAPTER IV

Generalization Error Bounds for Learning to Rank: Does the Length of Document Lists Matter?

4.1 Introduction

Training data in learning to rank consists of queries along with associated documents, where documents are represented as feature vectors. For each query, the documents are labeled with human relevance judgements. The goal at training time is to learn a ranking function that can, for a future query, rank its associated documents in order of their relevance to the query. The performance of ranking functions on test sets is evaluated using a variety of performance measures such as NDCG (*Järvelin and Kekäläinen*, 2002), ERR (*Chapelle et al.*, 2009) or Average Precision (*Yue et al.*, 2007).

The performance measures used for testing ranking methods cannot be directly optimized during training time as they lead to discontinuous optimization problems. As a result, researchers often minimize *surrogate* loss functions that are easier to optimize. For example, one might consider smoothed versions of, or convex upper bounds on, the target performance measure. However, as soon as one optimizes a surrogate loss, one has to deal with two questions (*Chapelle et al.*, 2011). First, does minimizing the surrogate on finite training data imply small expected surrogate loss on infinite unseen data? Second, does small expected surrogate loss on infinite unseen data imply small *target* loss on infi-

Table 4.1: A comparison of three bounds given in this chapter for Lipschitz loss functions. Criteria for comparison: algorithm bound applies to (OGD = Online Gradient Descent, [R]ERM = [Regularized] Empirical Risk Minimization), whether it applies to general (possibly non-convex) losses, and whether the constants involved are tight.

Bound	Applies to	Handles Nonconvex Loss	"Constant" hidden in $O(\cdot)$ notation
Theorem 35	OGD	No	Smallest
Theorem 36	RERM	No	Small
Theorem 38	ERM	Yes	Hides several logarithmic factors

nite unseen data? The first issue is one of *generalization error bounds* for empirical risk minimization (ERM) algorithms that minimize surrogate loss on training data. The second issue is one of *calibration*: does consistency in the surrogate loss imply consistency in the target loss?

This chapter deals with the former issue, viz. that of generalization error bounds for surrogate loss minimization. In pioneering works, *Lan et al.* (2008, 2009) gave generalization error bounds for learning to rank algorithms. However, while the former paper was restricted to analysis of pairwise approach to learning to rank, the later paper was limited to results on just three surrogates: ListMLE, ListNet and RankCosine. To the best of our knowledge, the most generally applicable bound on the generalization error of query-level learning to rank algorithms has been obtained by *Chapelle and Wu* (2010).

The bound of *Chapelle and Wu* (2010), while generally applicable, does have an explicit dependence on the *length* of the document list associated with a query. Our investigations begin with this simple question: is an explicit dependence on the length of document lists unavoidable in generalization error bounds for query-level learning to rank algorithms? We focus on the prevalent technique in literature where learning to rank algorithms learn linear scoring functions and obtain ranking by sorting scores in descending order. Our first contribution (Theorem 33) is to show that dimension of linear scoring functions that are *permutation invariant* (a necessary condition for being valid scoring functions for learning to rank) has no dependence on the length of document lists. Our second contribution

(Theorems 35, 36, 38) is to show that as long as one uses the "right" norm in defining the Lipschitz constant of the surrogate loss, we can derive generalization error bounds that have *no explicit dependence on the length of document lists*. The reason that the second contribution involves three bounds is that they all have different strengths and scopes of application (See Table 4.1 for a comparison). Our final contribution is to provide novel generalization error bounds for learning to rank in two previously unexplored settings: almost dimension independent bounds when using high dimensional features with ℓ_1 regularization (Theorem 40) and "optimistic" rates (that can be as fast as $O(1/n)$) when the loss function is smooth (Theorem 44). We also apply our results on popular convex and non-convex surrogates. All omitted proofs can be found in the appendix.

4.2 Preliminaries

In learning to rank (also called subset ranking to distinguish it from other related problems, e.g., bipartite ranking), a training example is of the form $((q, d_1, \ldots, d_m), y)$. Here q is a search query and $d_1, \ldots, d_m$ are m documents with varying degrees of *relevance* to the query. Human labelers provide the relevance vector $y \in \mathbb{R}^m$ where the entries in y contain the relevance labels for the m individual documents. Typically, y has integer-valued entries in the range $\{0, \ldots, Y_{\max}\}$ where $Y_{\max}$ is often less than 5. For our theoretical analysis, we get rid of some of these details by assuming that some feature map Ψ exists to map a query document pair (q, d) to $\mathbb{R}^d$. As a result, the training example $((q, d_1, \ldots, d_m), y)$ gets converted into (X, y) where $X = [\Psi(q, d_1), \ldots, \Psi(q, d_m)]^\top$ is an $m \times d$ matrix with the m query-document feature vector as rows. With this abstraction, we have an input space $\mathcal{X} \subseteq \mathbb{R}^{m \times d}$ and a label space $\mathcal{Y} \subseteq \mathbb{R}^m$.

A training set consists of iid examples $(X^{(1)}, y^{(1)}), \ldots, (X^{(n)}, y^{(n)})$ drawn from some underlying distribution D. To rank the documents in an instance $X \in \mathcal{X}$, often a score vector $s \in \mathbb{R}^m$ is computed. A ranking of the documents can then be obtained from s by sorting its entries in decreasing order. A common choice for the scoring function is to make

it *linear* in the input X and consider the following class of vector-valued functions:

$$\mathcal{F}_{\text{lin}} = \{X \mapsto Xw \ : \ X \in \mathbb{R}^{m\times d}, w \in \mathbb{R}^d\}. \tag{4.1}$$

Depending upon the regularization, we also consider the following two subclasses of $\mathcal{F}_{\text{lin}}$:

$$\mathcal{F}_2 := \{X \mapsto Xw \ : \ X \in \mathbb{R}^{m\times d}, w \in \mathbb{R}^d, \|w\|_2 \leq W_2\},$$
$$\mathcal{F}_1 := \{X \mapsto Xw \ : \ X \in \mathbb{R}^{m\times d}, w \in \mathbb{R}^d, \|w\|_1 \leq W_1\}.$$

In the input space $\mathcal{X}$, it is natural for the rows of X to have a bound on the appropriate dual norm. Accordingly, whenever we use $\mathcal{F}_2$, the input space is set to $\mathcal{X} = \{X \in \mathbb{R}^{m\times d} : \forall j \in [m],\ \|X_j\|_2 \leq R_X\}$ where X_j denotes jth row of X and $[m] := \{1, \ldots, m\}$. Similarly, when we use $\mathcal{F}_1$, we set $\mathcal{X} = \{X \in \mathbb{R}^{m\times d} \ : \ \forall j \in [m],\ \|X_j\|_\infty \leq \bar{R}_X\}$. These are natural counterparts to the following function classes studied in binary classification and regression:

$$\mathcal{G}_2 := \{x \mapsto \langle x, w\rangle \ : \ \|x\|_2 \leq R_X, w \in \mathbb{R}^d, \|w\|_2 \leq W_2\},$$
$$\mathcal{G}_1 := \{x \mapsto \langle x, w\rangle \ : \ \|x\|_\infty \leq \bar{R}_X, w \in \mathbb{R}^d, \|w\|_1 \leq W_1\}.$$

A key ingredient in the basic setup of the learning to rank problem is a loss function $\phi : \mathbb{R}^m \times \mathcal{Y} \to \mathbb{R}_+$ where $\mathbb{R}_+$ denotes the set of non-negative real numbers. Given a class $\mathcal{F}$ of vector-valued functions, a loss ϕ yields a natural loss class: namely the class of real-valued functions that one gets by composing ϕ with functions in $\mathcal{F}$:

$$\phi \circ \mathcal{F} := \{(X, y) \mapsto \phi(f(X), y) \ : \ X \in \mathbb{R}^{m\times d}, f \in \mathcal{F}\}.$$

For vector valued scores, the Lipschitz constant of ϕ depends on the norm $|||\cdot|||$ that we

decide to use in the score space ($|||\cdot|||_\star$ is dual of $|||\cdot|||$):

$$\forall y \in \mathcal{Y}, s, s' \in \mathbb{R}^m,\ |\phi(s_1, y) - \phi(s_2, y)| \leq G_\phi |||s_1 - s_2|||.$$

If ϕ is differentiable, this is equivalent to: $\forall y \in \mathcal{Y}, s \in \mathbb{R}^m,\ |||\nabla_s \phi(s, y)|||_\star \leq G_\phi$. Similarly, the smoothness constant H_ϕ of ϕ defined as: $\forall y \in \mathcal{Y}, s, s' \in \mathbb{R}^m$,

$$|||\nabla_s \phi(s_1, y) - \nabla_s \phi(s_2, y)|||_\star \leq H_\phi |||s_1 - s_2|||.$$

also depends on the norm used in the score space. If ϕ is twice differentiable, the above inequality is equivalent to

$$\forall y \in \mathcal{Y}, s \in \mathbb{R}^m,\ |||\nabla_s^2 \phi(s, y)|||_{\text{op}} \leq H_\phi$$

where $|||\cdot|||_{\text{op}}$ is the operator norm induced by the pair $|||\cdot|||, |||\cdot|||_\star$ and defined as $|||M|||_{\text{op}} := \sup_{v \neq 0} \frac{|||Mv|||_\star}{|||v|||}$. Define the expected loss of w under the distribution D $L_\phi(w) := \mathbb{E}_{(X,y)\sim D}\left[\phi(Xw, y)\right]$ and its empirical loss on the sample as $\hat{L}_\phi(w) := \frac{1}{n}\sum_{i=1}^n \phi(X^{(i)}w, y^{(i)})$. The minimizer of $L_\phi(w)$ (resp. $\hat{L}_\phi(w)$) over some function class (parameterized by w) will be denoted by $w^\star$ (resp. $\hat{w}$). We may refer to expectations w.r.t. the sample using $\widehat{\mathbb{E}}\left[\cdot\right]$. To reduce notational clutter, we often refer to (X, y) jointly by Z and $\mathcal{X} \times \mathcal{Y}$ by $\mathcal{Z}$. For vectors, $\langle u, v\rangle$ denotes the standard inner product $\sum_i u_i v_i$ and for matrices U, V of the same shape, $\langle U, V\rangle$ means $\text{Tr}(U^\top V) = \sum_{ij} U_{ij} V_{ij}$. The set of $m!$ permutation π of degree m is denoted by S_m. A vector of ones is denoted by $\mathbf{1}$.

4.3 Application to Specific Losses

To whet the reader's appetite for the technical presentation that follows, we will consider two loss functions, one convex and one non-convex, to illustrate the concrete improvements offered by our new generalization bounds. A generalization bound is of the

form: $L_\phi(\hat{w}) \leq L_\phi(w^\star)$+"complexity term". It should be noted that $w^\star$ is not available to the learning algorithm as it needs knowledge of underlying distribution of the data. The complexity term of *Chapelle and Wu* (2010) is $O(G_\phi^{CW} W_2 R_X \sqrt{m/n})$. The constant G_ϕ^{CW} is the Lipschitz constant of the surrogate ϕ (viewed as a function of the score vector s) w.r.t. ℓ_2 norm. Our bounds will instead be of the form $O(G_\phi W_2 R_X \sqrt{1/n})$, where G_ϕ is the Lipschitz constant of ϕ w.r.t. ℓ_∞ norm. Note that our bounds are free of any explicit m dependence. Also, by definition, $G_\phi \leq G_\phi^{CW}\sqrt{m}$ but the former can be much smaller as the two examples below illustrate. In benchmark datasets (*Liu et al.*, 2007b), m can easily be in the 100-1000 range.

4.3.1 Application to ListNet

The ListNet ranking method (*Cao et al.*, 2007a) uses a *convex* surrogate, that is defined in the following way[1]. Define m maps from $\mathbb{R}^m$ to $\mathbb{R}$ as: $P_j(v) = \exp(v_j)/\sum_{i=1}^m \exp(v_i)$ for $j \in [m]$. Then, we have, for $s \in \mathbb{R}^m$ and $y \in \mathbb{R}^m$,

$$\phi_{\text{LN}}(s, y) = -\sum_{j=1}^{m} P_j(y) \log P_j(s).$$

An easy calculation shows that the Lipschitz (as well as smoothness) constant of ϕ_{LN} is m independent.

Proposition 1. *The Lipschitz (resp. smoothness) constant of ϕ_{LN} w.r.t. $\|\cdot\|_\infty$ satisfies $G_{\phi_{\text{LN}}} \leq 2$ (resp. $H_{\phi_{\text{LN}}} \leq 2$) for any $m \geq 1$.*

Since the bounds above are independent of m, so the generalization bounds resulting from their use in Theorem 38 and Theorem 44 will also be independent of m (up to logarithmic factors). We are not aware of prior generalization bounds for ListNet that do not scale with m. In particular, the results of *Lan et al.* (2009) have an $m!$ dependence since

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

they consider the top-m version of ListNet. However, even if the top-1 variant above is considered, their proof technique will result in at least a linear dependence on m and does not result in as tight a bound as we get from our general results. It is also easy to see that the Lipschitz constant $G^{CW}_{\phi_{\mathrm{LN}}}$ of ListNet loss w.r.t. ℓ_2 norm is also 2 and hence the bound of *Chapelle and Wu* (2010) necessarily has a $\sqrt{m}$ dependence in it. Moreover, generalization error bounds for ListNet exploiting its smoothness will interpolate between the pessimistic $1/\sqrt{n}$ and optimistic $1/n$ rates. These have never been provided before.

4.3.2 Application to Smoothed DCG@1

This example is from the work of *Chapelle and Wu* (2010). Smoothed DCG@1, a *non-convex* surrogate, is defined as:

$$\phi_{\mathrm{SD}}(s, y) = D(1) \sum_{i=1}^{m} G(y_i) \frac{\exp(s_i/\sigma)}{\sum_j \exp(s_j/\sigma)},$$

where $D(i) = 1/\log_2(1+i)$ is the "discount" function and $G(i) = 2^i - 1$ is the "gain" function. The amount of smoothing is controlled by the parameter $\sigma > 0$ and the smoothed version approaches DCG@1 as $\sigma \to 0$ (DCG stands for Discounted Cumulative Gain *Järvelin and Kekäläinen* (2002)).

Proposition 2. *The Lipschitz constant of ϕ_{SD} w.r.t. $\|\cdot\|_\infty$ satisfies $G_{\phi_{\mathrm{SD}}} \leq 2D(1)G(Y_{\max})/\sigma$ for any $m \geq 1$. Here $Y_{\max}$ is maximum possible relevance score of a document (usually less than 5).*

As in the ListNet loss case we previously considered, the generalization bound resulting from Theorem 38 will be independent of m. This is intuitively satisfying: DCG@1, whose smoothing we are considering, only depends on the document that is put in the top position by the score vector s (and not on the entire sorted order of s). Our generalization bound does not deteriorate as the total list size m grows. In contrast, the bound of *Chapelle and Wu* (2010) will necessarily deteriorate as $\sqrt{m}$ since the constant $G^{CW}_{\phi_{\mathrm{SD}}}$ is the same as $G_{\phi_{\mathrm{SD}}}$.

Moreover, it should be noted that even in the original SmoothedDCG paper, σ is present in the denominator of $G^{CW}_{\phi_{\text{SD}}}$, so our results are directly comparable. Also note that this example can easily be extended to consider DCG@k for case when document list length $m \gg k$ (a very common scenario in practice).

4.3.3 Application to RankSVM

RankSVM (*Joachims*, 2002) is another well established ranking method, which minimizes a *convex* surrogate based on pairwise comparisons of documents. A number of studies have shown that ListNet has better empirical performance than RankSVM. One possible reason for the better performance of ListNet over RankSVM is that the Lipschitz constant of RankSVM surrogate w.r.t $\|\cdot\|_\infty$ doe scale with document list size as $O(m^2)$. Due to lack of space, we give the details in the supplement.

4.4 Does The Length of Document Lists Matter?

Our work is directly motivated by a very interesting generalization bound for learning to rank due to *Chapelle and Wu* (2010, Theorem 1). They considered a Lipschitz continuous loss ϕ with Lipschitz constant G^{CW}_ϕ w.r.t. *the* ℓ_2 *norm*. They show that, with probability at least $1-\delta$,

$$\forall w \in \mathcal{F}_2, \quad L_\phi(w) \leq \hat{L}_\phi(w) + 3\, G^{CW}_\phi W_2 R_X \sqrt{\frac{m}{n}} + \sqrt{\frac{8\log(1/\delta)}{n}}.$$

The dominant term on the right is $O(G^{CW}_\phi W_2 R_X \sqrt{m/n})$. In the next three sections, we will derive improved bounds of the form $\tilde{O}(G_\phi W_2 R_X \sqrt{1/n})$ where $G_\phi \leq G^{CW}_\phi \sqrt{m}$ but can be much smaller. Before we do that, let us examine the dimensionality reduction in linear scoring function that is caused by a natural permutation invariance requirement.

4.4.1 Permutation invariance removes m dependence in dimensionality of linear scoring functions

As stated in Section 4.2, a ranking is obtained by sorting a score vector obtained via a linear scoring function f. Consider the space of *linear* scoring function that consists of *all* linear maps f that map $\mathbb{R}^{m\times d}$ to $\mathbb{R}^m$:

$$\mathcal{F}_{\text{full}} := \left\{ X \mapsto \left[\langle X, W_1\rangle, \ldots, \langle X, W_m\rangle\right]^\top \;:\; W_i \in \mathbb{R}^{m\times d} \right\}.$$

These linear maps are fully parameterized by matrices $W_1, \ldots, W_m$. Thus, a full parameterization of the linear scoring function is of dimension $m^2 d$. Note that the popularly used class of linear scoring functions $\mathcal{F}_{\text{lin}}$ defined in Eq. 4.1 is actually a low d-dimensional subspace of the full $m^2 d$ dimensional space of all linear maps. It is important to note that the dimension of $\mathcal{F}_{\text{lin}}$ is *independent of* m.

In learning theory, one of the factors influencing the generalization error bound is the richness of the class of hypothesis functions. Since the linear function class $\mathcal{F}_{\text{lin}}$ has dimension independent of m, we intuitively expect that, at least under some conditions, algorithms that minimize ranking losses using linear scoring functions should have an m independent complexity term in the generalization bound. The reader might wonder whether the dimension reduction from $m^2 d$ to d in going from $\mathcal{F}_{\text{full}}$ to $\mathcal{F}_{\text{lin}}$ is arbitrary. To dispel this doubt, we prove the lower dimensional class $\mathcal{F}_{\text{lin}}$ is the *only sensible choice* of linear scoring functions in the learning to rank setting. This is because scoring functions should satisfy a permutation invariance property. That is, if we apply a permutation $\pi \in S_m$ to the rows of X to get a matrix πX then the scores should also simply get permuted by π. That is, we should only consider scoring functions in the following class:

$$\mathcal{F}_{\text{perminv}} = \{f : \forall \pi \in S_m, \forall X \in \mathbb{R}^{m\times d}, \pi f(X) = f(\pi X)\}.$$

The permutation invariance requirement, in turn, forces a reduction from dimension $m^2 d$

to just $2d$ (which has no dependence on m).

Theorem 33. *The intersection of the function classes $\mathcal{F}_{\text{full}}$ and $\mathcal{F}_{\text{perminv}}$ is the $2d$-dimensional class:*

$$\mathcal{F}'_{\text{lin}} = \{X \mapsto Xw + (\mathbf{1}^\top Xv)\mathbf{1} \ : \ w, v \in \mathbb{R}^d\}. \tag{4.2}$$

Note that the extra degree of freedom provided by the v parameter in Eq. 4.2 is useless for ranking purposes since adding a constant vector (i.e., a multiple of $\mathbf{1}$) to a score vector has no effect on the sorted order. This is why we said that $\mathcal{F}_{\text{lin}}$ is the only sensible choice of linear scoring functions.

4.5 Online to Batch Conversion

In this section, we build some intuition as to why it is natural to use $\|\cdot\|_\infty$ in defining the Lipschitz constant of the loss ϕ. To this end, consider the following well known online gradient descent (OGD) regret guarantee. Recall that OGD refers to the simple online algorithm that makes the update $w_{i+1} \leftarrow w_i - \eta \nabla_{w_i} f_i(w_i)$ at time i. If we run OGD to generate w_i's, we have, for all $\|w\|_2 \leq W_2$:

$$\sum_{i=1}^{n} f_i(w_i) - \sum_{i=1}^{n} f_i(w) \leq \frac{W_2^2}{2\eta} + \eta G^2 n$$

where G is a bound on the maximum ℓ_2-norm of the gradients $\nabla_{w_i} f_i(w_i)$ and f_i's have to be *convex*. If $(X^{(1)}, y^{(1)}), \ldots, (X^{(n)}, y^{(n)})$ are iid then by setting $f_i(w) = \phi(X^{(i)}w, y^{(i)})$, $1 \leq i \leq n$ we can do an "online to batch conversion". That is, we optimize over η, take expectations and use Jensen's inequality to get the following excess risk bound:

$$\forall \|w\|_2 \leq W_2, \ \mathbb{E}L_\phi(\hat{w}_{\text{OGD}}) - L_\phi(w) \leq W_2 G\sqrt{\frac{2}{n}}$$

where $\hat{w}_{\mathrm{OGD}} = \frac{1}{n}\sum_{i=1}^{n} w_i$ and G has to satisfy (noting that $s = X^{(i)}w_i$)

$$G \geq \|\nabla_{w_i} f_i(w_i)\|_2 = \|(X^{(i)})^\top \nabla_s \phi(X^{(i)}w_i, y^{(i)})\|_2$$

where we use the chain rule to express ∇_w in terms of ∇_s. Finally, we can upper bound

$$\begin{aligned}
&\|(X^{(i)})^\top \nabla_s \phi(X^{(i)}w_i, y^{(i)})\|_2 \\
&\leq \|(X^{(i)})^\top\|_{1\to 2} \cdot \|\nabla_s \phi(X^{(i)}w_i, y^{(i)})\|_1 \\
&\leq R_X \|\nabla_s \phi(X^{(i)}w_i, y^{(i)})\|_1
\end{aligned}$$

as $R_X \geq \max_{j=1}^{m} \|X_j\|_2$ and because of the following lemma.

Lemma 34. *For any $1 \leq p \leq \infty$,*

$$\begin{aligned}
\|X\|_{p\to q} &= \sup_{v\neq 0} \frac{\|Xv\|_q}{\|v\|_p} \\
\|X^\top\|_{1\to p} &= \|X\|_{q\to\infty} = \max_{j=1}^{m} \|X_j\|_p \,,
\end{aligned}$$

where q is the dual exponent of p (i.e., $\frac{1}{q} + \frac{1}{p} = 1$).

Thus, we have shown the following result.

Theorem 35. *Let ϕ be convex and have Lipschitz constant G_ϕ w.r.t. $\|\cdot\|_\infty$. Suppose we run online gradient descent (with appropriate step size η) on $f_i(w) = \phi(X^{(i)}w, y^{(i)})$ and return $\hat{w}_{\mathrm{OGD}} = \frac{1}{T}\sum_{i=1}^{n} w_i$. Then we have,*

$$\forall \|w\|_2 \leq W_2,\ \mathbb{E}L_\phi(\hat{w}_{\mathrm{OGD}}) - L_\phi(w) \leq G_\phi\, W_2\, R_X \sqrt{\frac{2}{n}}.$$

The above excess risk bound has no explicit m dependence. This is encouraging but there are two deficiencies of this approach based on online regret bounds. First, the result applies to the output of a specific algorithm that may not be the method of choice for

practitioners. For example, the above argument does not yield uniform convergence bounds that could lead to excess risk bounds for ERM (or regularized versions of it). Second, there is no way to generalize the result to Lipschitz, but *non-convex* loss functions. It may noted here that the original motivation for *Chapelle and Wu* (2010) to prove their generalization bound was to consider the non-convex loss used in their SmoothRank method. We will address these issues in the next two sections.

4.6 Stochastic Convex Optimization

We first define the regularized empirical risk minimizer:

$$\hat{w}_\lambda = \operatorname*{argmin}_{\|w\|_2 \leq W_2} \frac{\lambda}{2}\|w\|_2^2 + \hat{L}_\phi(w). \tag{4.3}$$

We now state the main result of this section.

Theorem 36. *Let the loss function ϕ be convex and have Lipschitz constant G_ϕ w.r.t. $\|\cdot\|_\infty$. Then, for an appropriate choice of $\lambda = O(1/\sqrt{n})$, we have*

$$\mathbb{E}L_\phi(\hat{w}_\lambda) \leq L_\phi(w^\star) + 2\, G_\phi\, R_X\, W_2 \left(\frac{8}{n} + \sqrt{\frac{2}{n}}\right).$$

This result applies to a batch algorithm (regularized ERM) but unfortunately requires the regularization parameter λ to be set in a particular way. Also, it does not apply to non-convex losses and does not yield uniform convergence bounds. In the next section, we will address these deficiencies. However, we will incur some extra logarithmic factors that are absent in the clean bound above.

4.7 Bounds for Non-convex Losses

The above discussion suggests that we have a possibility of deriving tighter, possibly m-independent, generalization error bounds by assuming that ϕ is Lipschitz continuous w.r.t.

$\|\cdot\|_\infty$. The standard approach in binary classification is to appeal to the Ledoux-Talagrand contraction principle for establishing Rademacher complexity (*Bartlett and Mendelson*, 2003). It gets rid of the loss function and incurs a factor equal to the Lipschitz constant of the loss in the Rademacher complexity bound. Since the loss function takes scalar argument, the Lipschitz constant is defined for only one norm, i.e., the absolute value norm. It is not immediately clear how such an approach would work when the loss takes vector valued arguments and is Lipschitz w.r.t. $\|\cdot\|_\infty$. We are not aware of an appropriate extension of the Ledoux-Talagrand contraction principle. Note that Lipschitz continuity w.r.t. the Euclidean norm $\|\cdot\|_2$ does not pose a significant challenge since Slepian's lemma can be applied to get rid of the loss. Several authors have already exploited Slepian's lemma in this context (*Bartlett and Mendelson*, 2003; *Chapelle and Wu*, 2010). We take a route involving covering numbers and define the data-dependent (pseudo-)metric:

$$d_\infty^{Z^{(1:n)}}(w, w') := \max_{i=1}^{n} \left|\phi(X^{(i)}w, y^{(i)}) - \phi(X^{(i)}w', y^{(i)})\right|$$

Let $\mathcal{N}_\infty(\epsilon, \phi \circ \mathcal{F}, Z^{(1:n)})$ be the covering number at scale ϵ of the composite class $\phi \circ \mathcal{F} = \phi \circ \mathcal{F}_1$ or $\phi \circ \mathcal{F}_2$ w.r.t. the above metric. Also define

$$\mathcal{N}_\infty(\epsilon, \phi \circ \mathcal{F}, n) := \max_{Z^{(1:n)}} \mathcal{N}_\infty(\epsilon, \phi \circ \mathcal{F}, Z^{(1:n)}).$$

With these definitions in place, we can state our first result on covering numbers.

Proposition 3. *Let the loss ϕ be Lipschitz w.r.t. $\|\cdot\|_\infty$ with constant G_ϕ. Then following covering number bound holds:*

$$\log_2 \mathcal{N}_\infty(\epsilon, \phi \circ \mathcal{F}_2, n) \leq \left\lceil \frac{G_\phi^2 W_2^2 R_X^2}{\epsilon^2} \right\rceil \log_2(2mn + 1).$$

Proof. Note that

$$\begin{aligned}&\max_{i=1}^{n} \left|\phi(X^{(i)}w, y^{(i)}) - \phi(X^{(i)}w', y^{(i)})\right| \\ &\leq G_\phi \cdot \max_{i=1}^{n} \max_{j=1}^{m} \left|\left\langle X_j^{(i)}, w \right\rangle - \left\langle X_j^{(i)}, w' \right\rangle\right|.\end{aligned}$$

This immediately implies that if we have a cover of the class $\mathcal{G}_2$ (Sec.4.2) at scale ϵ/G_ϕ w.r.t. the metric

$$\max_{i=1}^{n} \max_{j=1}^{m} \left|\left\langle X_j^{(i)}, w \right\rangle - \left\langle X_j^{(i)}, w' \right\rangle\right|$$

then it is also a cover of $\phi \circ \mathcal{F}_2$ w.r.t. $d_\infty^{Z^{(1:n)}}$, at scale ϵ. Now comes a simple, but crucial observation: *from the point of view of the scalar valued function class $\mathcal{G}_2$, the vectors $(X_j^{(i)})_{j=1:m}^{i=1:n}$ constitute a data set of size mn.* Therefore,

$$\mathcal{N}_\infty(\epsilon, \phi \circ \mathcal{F}_2, n) \leq \mathcal{N}_\infty(\epsilon/G_\phi, \mathcal{G}_2, mn). \tag{4.4}$$

Now we appeal to the following bound due to *Zhang* (2002, Corollary 3) (and plug the result into (4.4)):

$$\log_2 \mathcal{N}_\infty(\epsilon/G_\phi, \mathcal{G}_2, mn) \leq \left\lceil \frac{G_\phi^2 W_2^2 R_X^2}{\epsilon^2} \right\rceil \log_2(2mn+1)$$

□

Covering number $\mathcal{N}_2(\epsilon, \phi \circ \mathcal{F}, Z^{(1:n)})$ uses pseudo-metric:

$$d_2^{Z^{(1:n)}}(w, w') := \left(\sum_{i=1}^{n} \frac{1}{n} \left(\phi(X^{(i)}w, y^{(i)}) - \phi(X^{(i)}w', y^{(i)})\right)^2\right)^{1/2}$$

It is well known that a control on $\mathcal{N}_2(\epsilon, \phi \circ \mathcal{F}, Z^{(1:n)})$ provides control on the empirical Rademacher complexity and that $\mathcal{N}_2$ covering numbers are smaller than $\mathcal{N}_\infty$ ones. For us, it will be convenient to use a more refined version[2] due to *Mendelson* (2002). Let $\mathcal{H}$ be a

[2]We use a further refinement due to Srebro and Sridharan available at `http://ttic.uchicago.`

class of functions, with $\mathcal{H} : \mathcal{Z} \mapsto \mathbb{R}$, uniformly bounded by B. Then, we have following bound on empirical Rademacher complexity

$$\begin{aligned}
&\widehat{\mathfrak{R}}_n(\mathcal{H}) \\
&\leq \inf_{\alpha>0} \left(4\alpha + 10 \int_{\alpha}^{\sup_{h\in\mathcal{H}} \sqrt{\widehat{\mathbb{E}}[h^2]}} \sqrt{\frac{\log_2 \mathcal{N}_2(\epsilon, \mathcal{H}, Z^{(1:n)})}{n}} d\epsilon \right) \qquad (4.5) \\
&\leq \inf_{\alpha>0} \left(4\alpha + 10 \int_{\alpha}^{B} \sqrt{\frac{\log_2 \mathcal{N}_2(\epsilon, \mathcal{H}, Z^{(1:n)})}{n}} d\epsilon \right). \qquad (4.6)
\end{aligned}$$

Here $\widehat{\mathfrak{R}}_n(\mathcal{H})$ is the empirical Rademacher complexity of the class $\mathcal{H}$ defined as

$$\widehat{\mathfrak{R}}_n(\mathcal{H}) := \mathbb{E}_{\sigma_{1:n}} \left[\sup_{h\in\mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(Z_i) \right],$$

where $\sigma_{1:n} = (\sigma_1, \ldots, \sigma_n)$ are iid Rademacher (symmetric Bernoulli) random variables.

Corollary 37. *Let ϕ be Lipschitz w.r.t. $\|\cdot\|_\infty$ and uniformly bounded*[3] *by B for $w \in \mathcal{F}_2$. Then the empirical Rademacher complexities of the class $\phi \circ \mathcal{F}_2$ is bounded as*

$$\begin{aligned}
\widehat{\mathfrak{R}}_n(\phi \circ \mathcal{F}_2) \leq 10 G_\phi W_2 R_X \sqrt{\frac{\log_2(3mn)}{n}} \\
\times \log \frac{6B\sqrt{n}}{5 G_\phi W_2 R_X \sqrt{\log_2(3mn)}}.
\end{aligned}$$

Proof. This follows by simply plugging in estimates from Proposition 3 into (4.6) and choosing α optimally. □

Control on the Rademacher complexity immediately leads to uniform convergence bounds and generalization error bounds for ERM. The informal $\tilde{O}$ notation hides factors logarithmic in $m, n, B, G_\phi, R_X, W_1$. Note that all hidden factors are small and computable

edu/~karthik/dudley.pdf

[3] A uniform bound on the loss easily follows under the (very reasonable) assumption that $\forall y, \exists s_y$ s.t. $\phi(s_y, y) = 0$. Then $\phi(Xw, y) \leq G_\phi \|Xw - s_y\|_\infty \leq G_\phi(W_2 R_X + \max_{y\in\mathcal{Y}} \|s_y\|_\infty) \leq G_\phi(2W_2 R_X)$.

from the results above.

Theorem 38. *Suppose ϕ is Lipschitz w.r.t. $\|\cdot\|_\infty$ with constant G_ϕ and is uniformly bounded by B as w varies over $\mathcal{F}_2$. With probability at least $1-\delta$,*

$$\forall w \in \mathcal{F}_2, \quad L_\phi(w) \leq \hat{L}_\phi(w) + \tilde{O}\left(G_\phi W_2 R_X \sqrt{\frac{1}{n}} + B\sqrt{\frac{\log(1/\delta)}{n}}\right)$$

and therefore with probability at least $1-2\delta$,

$$L_\phi(\hat{w}) \leq L_\phi(w^\star) + \tilde{O}\left(G_\phi W_2 R_X \sqrt{\frac{1}{n}} + B\sqrt{\frac{\log(1/\delta)}{n}}\right).$$

where $\hat{w}$ is an empirical risk minimizer over $\mathcal{F}_2$.

Proof. Follows from standard bounds using Rademacher complexity. See, for example, *Bartlett and Mendelson* (2003). □

As we said before, ignoring logarithmic factors, the bound for $\mathcal{F}_2$ is an improvement over the bound of *Chapelle and Wu* (2010).

4.8 Extensions

We extend the generalization bounds above to two settings: a) high dimensional features and b) smooth losses.

4.8.1 High-dimensional features

In learning to rank situations involving high dimensional features, it may not be appropriate to use the class $\mathcal{F}_2$ of ℓ_2 bounded predictors. Instead, we would like to consider the class $\mathcal{F}_1$ of ℓ_1 bounded predictors. In this case, it is natural to measure size of the input matrix X in terms of a bound $\bar{R}_X$ on the maximum ℓ_∞ norm of each of its row. The following analogue of Proposition 3 can be shown.

Proposition 4. *Let the loss ϕ be Lipschitz w.r.t. $\|\cdot\|_\infty$ with constant G_ϕ. Then the following covering number bound holds:*

$$\log_2 \mathcal{N}_\infty(\epsilon, \phi \circ \mathcal{F}_1, n) \leq \left\lceil \frac{288\, G_\phi^2\, W_1^2\, \bar{R}_X^2\, (2 + \log d)}{\epsilon^2} \right\rceil \times \log_2 \left(2 \left\lceil \frac{8 G_\phi W_1 \bar{R}_X}{\epsilon} \right\rceil mn + 1 \right).$$

Using the above result to control the Rademacher complexity of $\phi \circ \mathcal{F}_1$ gives the following bound.

Corollary 39. *Let ϕ be Lipschitz w.r.t. $\|\cdot\|_\infty$ and uniformly bounded by B for $w \in \mathcal{F}_1$. Then the empirical Rademacher complexities of the class $\phi \circ \mathcal{F}_1$ is bounded as*

$$\widehat{\mathfrak{R}}_n(\phi \circ \mathcal{F}_1) \leq 120\sqrt{2} G_\phi W_1 \bar{R}_X \sqrt{\frac{\log(d)\, \log_2(24 mn G_\phi W_1 \bar{R}_X)}{n}} \times \log^2 \frac{B + 24 mn G_\phi W_1 \bar{R}_X}{40\sqrt{2} G_\phi W_1 \bar{R}_X \sqrt{\log(d)\, \log_2(24 mn G_\phi W_1 \bar{R}_X)}}.$$

As in the previous section, control of Rademacher complexity immediately yields uniform convergence and ERM generalization error bounds.

Theorem 40. *Suppose ϕ is Lipschitz w.r.t. $\|\cdot\|_\infty$ with constant G_ϕ and is uniformly bounded by B as w varies over $\mathcal{F}_1$. With probability at least $1 - \delta$,*

$$\forall w \in \mathcal{F}_1, \quad L_\phi(w) \leq \hat{L}_\phi(w) + \tilde{O}\left(G_\phi W_1 \bar{R}_X \sqrt{\frac{\log d}{n}} + B\sqrt{\frac{\log(1/\delta)}{n}} \right)$$

and therefore with probability at least $1 - 2\delta$,

$$L_\phi(\hat{w}) \leq L_\phi(w^\star) + \tilde{O}\left(G_\phi W_1 \bar{R}_X \sqrt{\frac{\log d}{n}} + B\sqrt{\frac{\log(1/\delta)}{n}} \right)$$

where $\hat{w}$ is an empirical risk minimizer over $\mathcal{F}_1$.

As can be easily seen from Theorem. 40, the generalization bound is *almost* independent of the dimension of the document feature vectors. We are not aware of existence of such a result in learning to rank literature.

4.8.2 Smooth losses

We will again use online regret bounds to explain why we should expect "optimistic" rates for smooth losses before giving more general results for smooth but possibly non-convex losses.

4.8.3 Online regret bounds under smoothness

Let us go back to OGD guarantee, this time presented in a slightly more refined version. If we run OGD with learning rate η then, for all $\|w\|_2 \leq W_2$:

$$\sum_{i=1}^{n} f_i(w_i) - \sum_{i=1}^{n} f_i(w) \leq \frac{W_2^2}{2\eta} + \eta \sum_{i=1}^{n} \|g_i\|_2^2$$

where $g_i = \nabla_{w_i} f_i(w_i)$ (if f_i is not differentiable at w_i then we can set g_i to be an arbitrary subgradient of f_i at w_i). Now assume that all f_i's are non-negative functions and are smooth w.r.t. $\|\cdot\|_2$ with constant H. Lemma 3.1 of *Srebro et al.* (2010) tells us that any non-negative, smooth function $f(w)$ enjoy an important *self-bounding* property for the gradient:

$$\|\nabla_w f_i(w)\|_2 \leq \sqrt{4H f_i(w)}$$

which bounds the magnitude of the gradient of f at a point in terms of the value of the function itself at that point. This means that $\|g_i\|_2^2 \leq 4H f_i(w_i)$ which, when plugged into the OGD guarantee, gives:

$$\sum_{i=1}^{n} f_i(w_i) - \sum_{i=1}^{n} f_i(w) \leq \frac{W_2^2}{2\eta} + 4\eta H \sum_{i=1}^{n} f_i(w_i)$$

Again, setting $f_i(w) = \phi(X^{(i)}w, y^{(i)})$, $1 \leq t \leq n$, and using the online to batch conversion technique, we can arrive at the bound: for all $\|w\|_2 \leq W_2$:

$$\mathbb{E}L_\phi(\hat{w}) \leq \frac{L_\phi(w)}{(1 - 4\eta H)} + \frac{W_2^2}{2\eta(1 - 4\eta H)n}$$

At this stage, we can fix $w = w^\star$, the optimal ℓ_2-norm bounded predictor and get optimal η as:

$$\eta = \frac{W_2}{4HW_2 + 2\sqrt{4H^2W_2^2 + 2HL_\phi(w^\star)n}}. \tag{4.7}$$

After plugging this value of η in the bound above and some algebra (see Section C), we get the upper bound

$$\mathbb{E}L_\phi(\hat{w}) \leq L_\phi(w^\star) + 2\sqrt{\frac{2HW_2^2 L_\phi(w^\star)}{n}} + \frac{8HW_2^2}{n}. \tag{4.8}$$

Such a rate interpolates between a $1/\sqrt{n}$ rate in the "pessimistic" case ($L_\phi(w^\star) > 0$) and the $1/n$ rate in the "optimistic" case ($L_\phi(w^\star) = 0$) (this terminology is due to *Panchenko* (2002)).

Now, assuming ϕ to be twice differentiable, we need H such that

$$H \geq \|\nabla_w^2 \phi(X^{(i)}w, y^{(i)})\|_{2\to 2} = \|X^\top \nabla_s^2 \phi(X^{(i)}w, y^{(i)})X\|_{2\to 2}$$

where we used the chain rule to express ∇_w^2 in terms of ∇_s^2. Note that, for OGD, we need smoothness in w w.r.t. $\|\cdot\|_2$ which is why the matrix norm above is the operator norm corresponding to the pair $\|\cdot\|_2, \|\cdot\|_2$. In fact, when we say "operator norm" without mentioning the pair of norms involved, it is this norm that is usually meant. It is well known that this norm is equal to the largest singular value of the matrix. But, just as before, we can bound this in terms of the smoothness constant of ϕ w.r.t. $\|\cdot\|_\infty$ (see Section C in the appendix):

$$\begin{aligned} &\|(X^{(i)})^\top \nabla_s^2 \phi(X^{(i)}w, y^{(i)})X^{(i)}\|_{2\to 2} \\ &\leq R_X^2 \|\nabla_s^2 \phi(X^{(i)}w, y^{(i)})\|_{\infty\to 1}. \end{aligned}$$

where we used Lemma 34 once again. This result using online regret bounds is great for building intuition but suffers from the two defects we mentioned at the end of Section 4.5.

In the smoothness case, it additionally suffers from a more serious defect: the correct choice of the learning rate η requires knowledge of $L_\phi(w^\star)$ which is seldom available.

4.8.4 Generalization error bounds under smoothness

Once again, to prove a general result for possibly non-convex smooth losses, we will adopt an approach based on covering numbers. To begin, we will need a useful lemma from *Srebro et al.* (2010, Lemma A.1 in the Supplementary Material). Note that, for functions over real valued predictions, we do not need to talk about the norm when dealing with smoothness since essentially the only norm available is the absolute value.

Lemma 41. *For any h-smooth non-negative function $f : \mathbb{R} \to \mathbb{R}_+$ and any $t, r \in \mathbb{R}$ we have*

$$(f(t) - f(r))^2 \leq 6h(f(t) + f(r))(t - r)^2.$$

We first provide an extension of this lemma to the vector case.

Lemma 42. *If $\phi : \mathbb{R}^m \to \mathbb{R}_+$ is a non-negative function with smoothness constant H_ϕ w.r.t. a norm $|||\cdot|||$ then for any $s_1, s_2 \in \mathbb{R}^m$ we have*

$$(\phi(s_1) - \phi(s_2))^2 \leq 6H_\phi \cdot (\phi(s_1) + \phi(s_2)) \cdot |||s_1 - s_2|||^2.$$

Using the basic idea behind local Rademacher complexity analysis, we define the following loss class:

$$\mathcal{F}_{\phi,2}(r) := \{(X, y) \mapsto \phi(Xw, y) \ : \ \|w\|_2 \leq W_2, \hat{L}_\phi(w) \leq r\}.$$

Note that this is a random subclass of functions since $\hat{L}_\phi(w)$ is a random variable.

Proposition 5. *Let ϕ be smooth w.r.t. $\|\cdot\|_\infty$ with constant H_ϕ. The covering numbers of*

$\mathcal{F}_{\phi,2}(r)$ in the $d_2^{Z^{(1:n)}}$ metric defined above are bounded as follows:

$$\log_2 \mathcal{N}_2(\epsilon, \mathcal{F}_{\phi,2}(r), Z^{(1:n)}) \leq \left\lceil \frac{12 H_\phi W_2^2 R_X^2 r}{\epsilon^2} \right\rceil \log_2(2mn+1).$$

Control of covering numbers easily gives a control on the Rademacher complexity of the random subclass $\mathcal{F}_{\phi,2}(r)$.

Corollary 43. *Let ϕ be smooth w.r.t. $\|\cdot\|_\infty$ with constant H_ϕ and uniformly bounded by B for $w \in \mathcal{F}_2$. Then the empirical Rademacher complexity of the class $\mathcal{F}_{\phi,2}(r)$ is bounded as*

$$\widehat{\mathfrak{R}}_n\left(\mathcal{F}_{\phi,2}(r)\right) \leq 4\sqrt{r} C \log \frac{3\sqrt{B}}{C}$$

where $C = 5\sqrt{3} W_2 R_X \sqrt{\frac{H_\phi \log_2(3mn)}{n}}$.

With the above corollary in place we can now prove our second key result.

Theorem 44. *Suppose ϕ is smooth w.r.t. $\|\cdot\|_\infty$ with constant H_ϕ and is uniformly bounded by B over $\mathcal{F}_2$. With probability at least $1-\delta$,*

$$\forall w \in \mathcal{F}_2,\ L_\phi(w) \leq \hat{L}_\phi(w) + \tilde{O}\left(\sqrt{\frac{L_\phi(w) D_0}{n}} + \frac{D_0}{n}\right)$$

where $D_0 = B\log(1/\delta) + W_2^2 R_X^2 H_\phi$. Moreover, with probability at least $1-2\delta$,

$$L_\phi(\hat{w}) \leq L_\phi(w^\star) + \tilde{O}\left(\sqrt{\frac{L_\phi(w^\star) D_0}{n}} + \frac{D_0}{n}\right)$$

where $\hat{w}, w^\star$ are minimizers of $\hat{L}_\phi(w)$ and $L_\phi(w)$ respectively (over $w \in \mathcal{F}_2$).

4.9 Conclusion

We showed that it is not necessary for generalization error bounds for query-level learning to rank algorithms to deteriorate with increasing length of document lists associated

with queries. The key idea behind our improved bounds was defining Lipschitz constants w.r.t. ℓ_∞ norm instead of the "standard" ℓ_2 norm. As a result, we were able to derive much tighter guarantees for popular loss functions such as ListNet and Smoothed DCG@1 than previously available.

Our generalization analysis of learning to rank algorithms paves the way for further interesting work. One possibility is to use these bounds to design active learning algorithms for learning to rank with formal label complexity guarantees. Another interesting possibility is to consider other problems, such as multi-label learning, where functions with vector-valued outputs are learned by optimizing a joint function of those outputs.

CHAPTER V

Personalized Advertisement Recommendation: A Ranking Approach to Address the Ubiquitous Click Sparsity Problem

5.1 Introduction

Personalized advertisement recommendation (PAR) system is intrinsic to many major tech companies like Google, Yahoo, Facebook and others. The particular PAR setting we study here consists of a policy that displays one of the K possible ads/offers, when a user visits the system. The user's profile is represented as a context vector, consisting of relevant information like demographics, geo-location, frequency of visits, etc. Depending on whether user clicks on the ad, the system gets a reward of value 1, which in practice translates to dollar revenue. The policy is (continuously) updated from historical data, which consist of tuples of the form $\{$*user context*, *displayed ad*, *reward*$\}$. We will, in this chapter, concern ourselves with PAR systems that are geared towards maximizing total number of clicks.

The plethora of papers written on the PAR problem makes it impossible to provide an exhaustive list. Interested readers may refer to a recent paper by a team of researchers in Google (*McMahan et al.*, 2013) and references therein. While the techniques in different papers differ in their details, the majority of them can be be analyzed under the umbrella

framework of *contextual bandits* (*Langford and Zhang*, 2008). The term *bandit* refers to the fact that the system only gets to see the user's feedback on the ad that was displayed, and not on any other ad. Bandit information leads to difficulty in estimating the expected reward of a new or a relatively unexplored ad (the *cold start* problem). Thus, contextual bandit algorithms, during prediction, usually balance between *exploitation* and *exploration*. Exploitation consists of predicting according to the current recommendation policy, which usually selects the ad with the maximum estimated reward, and exploration consists of systematically choosing some other ad to display, to gather more information about it.

Most contextual bandit algorithms aim to learn a policy that is essentially some form of multi-class classifier. For example, one important class of contextual bandit algorithms learn a classifier per ad from the batch of data (*Richardson et al.*, 2007; *McMahan et al.*, 2013; *He et al.*, 2014) and convert it into a policy, that displays the ad with the highest classifier score to the user (exploitation). Some exploration techniques, like explicit ϵ-greedy (*Koh and Gupta*, 2014; *Theocharous et al.*, 2015) or implicit Bayesian type sampling from the posterior distribution maintained on classifier parameters (*Chapelle and Li*, 2011) are sometimes combined with this exploitation strategy. Other, more theoretically sophisticated online bandit algorithms, essentially learn a cost-sensitive multi-class classifier by updating after every round of user-system interaction (*Dudik et al.*, 2011; *Agarwal et al.*, 2014).

Despite the fact that PAR has always been mentioned as one of the main applications of CB algorithms, there has not been much investigation into the practical issues raised in using classifier-based policies for PAR. The potential difficulty in using such policies in PAR stems from the problem of *click sparsity*, i.e., very few users actually ever click on online ads and this lack of positive feedback makes it difficult to learn good classifiers. Our main objective here is to study this important practical issue and we list our contributions:

- We detail the framework of contextual bandit algorithms and discuss the problem associated with click sparsity.

- We suggest a simple ranker-based policy to overcome the click sparsity problem. The rankers are learnt by optimizing the *Area Under Curve* (AUC) ranking loss via *stochastic gradient descent* (SGD) (*Shamir and Zhang*, 2013), leading to a highly scalable algorithm. The rankers are then combined to create a recommendation policy.

- We conduct extensive experiments to illustrate the improvement provided by our suggested method over both linear and ensemble classifier-based policies for the PAR problem. Our first set of experiments compare *deterministic policies* on publicly available classification datasets, that are converted to bandit datasets following standard techniques. Our second set of experiments compare *stochastic policies* on three proprietary bandit datasets, for which we employ a high confidence offline contextual bandit evaluation technique.

5.2 Contextual Bandit (CB) Approach to PAR

The main contextual bandit algorithms can be largely divided into two classes: those that make specific parametric assumption about the reward generation process and those that simply assume that context and rewards are generated i.i.d. from some distribution. The two major algorithms in the first domain are LinUCB (*Chu et al.*, 2011) and Thompson sampling (*Agrawal and Goyal*, 2013). Both algorithms assume that the reward of each ad (arm) is a continuous linear function of some unknown parameter, which is not a suitable assumption for click-based *binary* reward in PAR. Moreover, both algorithms assume that there is context information available for each ad, while we assume availability of only user context in our setting. Thus, from now on, we focus on the second class of the contextual bandit algorithms. We provide a formal description of the framework of contextual bandits suited to the PAR setting, and then discuss the problem that arises due to click sparsity.

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $[K] = \{1, 2, \ldots, K\}$ denote the user context space and K different

ads/arms. At each round, it is assumed that a pair (x, r) is drawn i.i.d. from some unknown joint distribution D over $\mathcal{X} \times \{0,1\}^K$. Here, $x \in \mathcal{X}$ and $r \in \{0,1\}^K$ represent the user context vector and the full reward vector, i.e., the user's true preference for all the ads (the full reward vector is unknown to the algorithm). Π is the space of policies such that for any $\pi \in \Pi$, $\pi : \mathcal{X} \mapsto [K]$. Contextual bandit algorithms have the following steps:

- At each round t, the context vector x_t is revealed, i.e., a user visits the system.
- The system selects ad a_t according to the current policy $\pi_t \in \Pi$ (exploitation strategy). Optionally, an exploration strategy is sometimes added, creating a distribution $p_t(\cdot)$ over the ads and a_t is drawn from $p_t(\cdot)$. Policy π_t and distribution p_t are sometimes used synonymously by considering π_t to be a stochastic policy.
- Reward r_{t,a_t} is revealed and the new policy $\pi_{t+1} \in \Pi$ is computed, using information $\{x_t, a_t, r_{t,a_t}\}$. We *emphasize* that the system does not get to know $r_{t,a'}, \forall\, a' \neq a_t$.

Assuming the user-system interaction happens over T rounds, the objective of the system is to maximize its cumulative reward, i.e., $\sum_{t=1}^{T} r_{t,a_t}$. Note that since rewards are assumed to be binary, $\sum_{t=1}^{T} r_{t,a_t}$ is precisely the total number of clicks and $\frac{\sum_{t=1}^{T} r_{t,a_t}}{T}$ is the overall CTR of the recommendation algorithm. Theoretically, performance of a bandit algorithm is analyzed via the concept of *regret*, i.e.,

$$\text{Regret(T)} = \underbrace{\sum_{t=1}^{T} r_{t,\pi^*(x_t)}}_{\text{optimal policy's cumulative reward}} - \underbrace{\sum_{t=1}^{T} r_{t,a_t}}_{\text{algorithm's cumulative reward}}$$

where $\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \sum_{t=1}^{T} r_{t,\pi(x_t)}$. The desired property of any contextual bandit algorithm is to have a sublinear (in T) bound on Regret(T) (in expectation or high probability), i.e., Regret(T) $\leq$ o(T). *This guarantees that, at least, the algorithm converges to the optimal policy π^* asymptotically*.

5.3 Practical Issues with CB Policy Space

Policy space Π considered for major contextual bandit algorithms are based on classifiers. They can be tuples of binary classifiers, with one classifier per ad, or global cost-sensitive multi-class classifier, depending on the nature of the bandit algorithm. Since clicks on the ads are rare and small improvement in click-through rate can lead to significant reward, it is vital for the policy space to have good policies that can identify the correct ads for the rare users who are highly likely to click on them. Extreme click sparsity makes it *very practically challenging* to design a classifier-based policy space, where policies can identify the correct ads for rare users. *Crucially, contextual bandit algorithms are only concerned with converging as fast as possible to the best policy π^* in the policy space Π and do not take into account the nature of the policies. Hence, if the optimal policy in the policy space does a poor job in identifying correct ads, then the bandit algorithm will have very low cumulative reward, regardless of its sophistication.* We discuss how click sparsity hinders in the design of different types of classifier-based policies.

5.3.1 Binary Classifier Based Policies

Contextual bandit algorithms are traditionally presented as online algorithms, with continuous update of policies. Usually, in industrial PAR systems, it is highly impractical to update policies continuously, due to thousands of users visiting a system in a small time frame. Thus, policy update happens, i.e. new policy is learnt, after intervals of time, using the bandit data produced from the interaction between the current policy and users, collected in batch. It is convenient to learn a binary classifier per ad in such a setting. To explain the process concisely, we note that the bandit data consists of tuples of the form $\{x, a, r_a\}$. For each ad a, the users who had not clicked on the ad (r_a=0) would be considered as negative examples and the users who had clicked on the ad (r_a=1) would be considered as positive examples, creating a binary training set for ad a. The K binary classifiers are converted into a recommendation policy using a "one-vs-all" method (*Rifkin and*

Klautau, 2004). *Thus, each policy in policy space* Π *can be considered to be a tuple of* K *binary classifiers*.

A number of research publications show that researchers consider binary linear classifiers, that are learnt by optimizing the logistic loss (*Richardson et al.*, 2007), while ensemble classifiers, like random forests, are also becoming popular (*Koh and Gupta*, 2014). We note that the majority of the papers that learn a logistic linear classifier focus on feature selection (*He et al.*, 2014), novel regularizations to tackle high-dimensional context vectors (*McMahan et al.*, 2013), or propose clever combinations of logistic classifiers (*Agarwal et al.*, 2009).

Click sparsity poses difficulty in design of accurate binary classifiers in the following way: for an ad a, there will be very few clicks on the ad as compared to the number of users who did not click on the ad. A binary classifier learnt in such setting will almost always predict that its corresponding ad will not be clicked by a user, failing to identify the rare, but very important, users who are likely to click on the ad. This is colloquially referred to as "class imbalance problem" in binary classification (*Japkowicz and Stephen*, 2002). Due to the extreme nature of the imbalance problem, tricks like under-sampling of negative examples or oversampling of positive examples (*Chawla et al.*, 2004) are not very useful. More sophisticated techniques like cost-sensitive svms require prior knowledge about importance of each class, which is not generally available in the PAR setting.

Note- Some of the referenced papers do not have explicit mention of CBs because the focus in those papers is on the issues related to classifier learning process, involving type of regularization, overcoming curse of dimensionality, scalability etc. The important issue of extreme class imbalance has not received sufficient attention (Sec 6.2, (*He et al.*, 2014)). When the classifiers are used to predict ads, the technique is a particular CB algorithm (the exact exploration+ exploitation mix is often not revealed).

5.3.2 Cost Sensitive Multi-Class Classifier Based Policies

Another type of policy space consist of cost-sensitive multi-class classifiers (*Langford and Zhang*, 2008; *Dudik et al.*, 2011; *Agarwal et al.*, 2014). They can be cost-sensitive multi-class svms (*Cao et al.*, 2013), multi-class logistic classifiers or filter trees (*Beygelzimer et al.*, 2007). Click sparsity poses slightly different kind of problem in practically designing a policy space of such classifiers.

Cost sensitive multi-class classifier works as follows: assume a context-reward vector pair (x,r) is generated as described in the PAR setting. The classifier will try to select a class (ad) a such that the reward r_a is maximum among all choices of $r_{a'}$, $\forall\ a' \in [K]$ (we consider reward maximizing classifiers, instead of cost minimizing classifiers). Unlike in traditional multi-class classification, where one entry of r is 1 and all other entries are 0; in cost sensitive classification, r can have any combination of 1 and 0. Now consider the reward vectors r_ts generated over T rounds. A poor quality classifier π^p, which fails to identify the correct ad for most users x_t, will have very low average reward, i.e.,$\frac{\sum_{t=1}^{T} r_{t,\pi^p(x_t)}}{T} \sim O(\epsilon)$, with $\epsilon \sim 0$. From the model perspective, extreme click sparsity translates to almost all reward vectors r_t being $\vec{\mathbf{0}}$. Thus, even a very good classifier π^g, which can identify the correct ad a_t for almost all users, will have very low average reward, i.e., $\frac{\sum_{t=1}^{T} r_{t,\pi^g(x_t)}}{T} \sim O(\epsilon)$. From a practical perspective, it is difficult to distinguish between the performance of a good and poor classifier, in face of extreme sparsity, and thus, cost sensitive multi-class classifiers are not ideal policies for contextual bandits addressing the PAR problem.

5.4 AUC Optimized Ranker

We propose a ranking-based alternative to learning a classifier per ad, in the offline setting, that is capable of overcoming the click sparsity problem. We learn a ranker per ad by optimizing the Area Under the Curve (AUC) loss, and use a ranking score normalization technique to create a policy mapping context to ad. We note that AUC is a popular measure

used to evaluate a classifier on an imbalanced dataset. However, our objective is to explicitly use the loss to learn a ranker that overcomes the imbalance problem and then create a context to ad mapping policy.

Ranker Learning Technique: For an ad a, let $S^+ = \{x : r_a = 1\}$ and $S^- = \{x : r_a = 0\}$ be the set of positive and negative instances, respectively. Let f_w be a linear ranking function parameterized by $w \in \mathbb{R}^d$, i.e., $f_w(x) = w \cdot x$ (inner product). AUC-based loss (AUCL) is a ranking loss that is minimized when positive instances get higher scores than negative instances, i.e., the positive instances are ranked higher than the negatives when instances are sorted in descending order of their scores (*Cortes and Mohri*, 2004b). Formally, we define empirical AUCL for function $f_w(\cdot)$

$$\text{AUCL} = \frac{1}{|S^+||S^-|} \sum_{x^+ \in S^+} \sum_{x^- \in S^-} \mathbb{1}(\underbrace{f_w(x^+) - f_w(x^-)}_{t} < 0).$$

Direct optimization of AUCL is a NP-hard problem, since AUCL is sum of discontinuous indicator functions. To make the objective function computationally tractable, the indicator functions are replaced by a continuous, convex surrogate $\ell(t)$. Examples include hinge $\ell(t) = [1 - t]_+$ and logistic $\ell(t) = \log(1 + exp(-t))$ surrogates. Thus, the final objective function to optimize is

$$L(w) = \frac{1}{|S^+||S^-|} \sum_{x^+ \in S^+} \sum_{x^- \in S^-} \ell(\underbrace{f_w(x^+) - f_w(x^-)}_{t}). \tag{5.1}$$

Note: Since AUCL is a ranking loss, the concept of class imbalance ceases to be a problem. Irrespective of the number of positive and negative instances in the training set, the position of a positive instance w.r.t to a negative instance in the final ranked list is the only matter of concern in AUCL calculation.

5.4.1 Optimization Procedure

The objective function (5.1) is a convex function and can be efficiently optimized by *stochastic gradient descent* (SGD) procedure (*Shamir and Zhang*, 2013). One computa-

tional issue associated with AUCL is that it pairs every positive and negative instance, effectively squaring the training set size.The SGD procedure easily overcomes this computational issue. At every step of SGD, a positive and a negative instance are randomly selected from the training set, followed by a gradient descent step. This makes the training procedure memory-efficient and mimics full gradient descent optimization on the entire loss. We also note that the rankers for the ads can be trained in parallel and any regularizer like $\|w\|_1$ and $\|w\|_2$ can be added to (5.1), to introduce sparsity or avoid overfitting. Lastly, powerful non-linear kernel ranking functions can be learnt in place of linear ranking functions, but at the cost of memory efficiency, and the rankers can even be learnt online, from streaming data (*Zhao et al.*, 2011).

5.4.2 Constructing Policy from Rankers

Similar to learning a classifier per ad, a separate ranking function $f_{w_a}(\cdot)$ is learnt for each ad a from the bandit batch data. Then the following technique is used to convert the K separate ranking functions into a recommendation policy. First, a threshold score s_a is learnt for each action $a \in [K]$ separately (see the details below), and then for a new user x, the combined policy π works as follows:

$$\pi(x) = \underset{a \in [K]}{\operatorname{argmax}} \, (f_{w_a}(x) - s_a). \tag{5.2}$$

Thus, π maps x to ad a with maximum "normalized score". This normalization negates the inherent scoring bias that might exist for each ranking function. That is, a ranking function for an action $a \in [K]$ might learn to score all instances (both positive and negative) higher than a ranking function for an action $b \in [K]$. Therefore, for a new instance x, ranking function for a will always give a higher score than the ranking function for b, leading to possible incorrect predictions.

Learning Threshold Score s_a: After learning the ranking function $f_{w_a}(\cdot)$ from the training data, the threshold score s_a is learnt by maximizing some classification measure like precision, recall, or F-score on the same training set. That is, score of each (positive or negative) instance in the training set is calculated and the classification measure corresponding to different thresholds are compared. The threshold that gives the maximum measure value is assigned to s_a.

5.5 Competing Policies and Evaluation Techniques

To support our hypothesis that ranker based policies address the click-sparsity problem better than classifier based policies, we set up two sets of experiments. We a) compared deterministic policies (only "exploitation") on full information (classification) datasets and b) compared stochastic policies ("exploitation + exploration") on bandit datasets, with a specific offline evaluation technique. Both of our experiments were designed for batch learning setting, with policies constructed from separate classifiers/rankers per ad. The classifiers considered were **linear** and **ensemble RandomForest** classifiers and ranker considered was the **AUC optimized ranker**.

Deterministic Policies: Policies from the trained classifiers were constructed using the "one-vs-all" technique, i.e., for a new user x, the ad with the maximum score according to the classifiers was predicted. For the policy constructed from rankers, the ad with the maximum shifted score according to the rankers was predicted, using Eq. 5.2. Deterministic policies are "exploit only" policies.

Stochastic Policies: Stochastic policies were constructed from deterministic policies by adding an ϵ-greedy exploration technique on top. Briefly, let one of the stochastic policies be denoted by π_e and let $\epsilon \in [0, 1]$. For a context x in the test set, $\pi_e(a|x) = 1 - \epsilon$, if a was the offer with the maximum score according to the underlying deterministic policy, and $\pi_e(a|x) = \frac{\epsilon}{K-1}$, otherwise ($K$ is the total number of offers). Thus, π_e is a probability distribution over the offers. Stochastic policies are "exploit+ explore" policies.

5.5.1 Evaluation on Full Information Classification Data

Benchmark bandit data are usually hard to obtain in public domains. So, we compared the deterministic policies on benchmark K-class classification data, converted to K-class bandit data, using the technique in (*Langford et al.*, 2011). Briefly, the standard conversion technique is as follows: A K-class dataset is randomly split into training set X_{train} and test set X_{test} (in our experiments, we used $70 - 30$ split). *Only the labeled training set is converted into bandit data, as per procedure*. Let $\{x, a\}$ be an instance and the corresponding class in the training set. A class $a' \in [K]$ is selected uniformly at random. If $a = a'$, a reward of 1 is assigned to x; otherwise, a reward of 0 is assigned. The new bandit instance is of the form $\{x, a', 1\}$ or $\{x, a', 0\}$, and the true class a is hidden. The bandit data is then divided into K separate binary class training sets, as detailed in the section "Binary Classifier based Policies".

Evaluation Technique: We compared the *deterministic policies* by calculating the CTR of each policy. For a policy π, CTR on a test set of cardinality n is measured as:

$$\frac{1}{n} \sum_{(x,a) \in \text{test set}} \mathbb{1}(\pi(x) = a) \tag{5.3}$$

Note that we can calculate the true CTR of a policy π because the correct class a for an instance x is known in the test set.

5.5.2 Evaluation on Bandit Information Data

Bandit datasets have both training and test sets in bandit form, and datasets we use are industry proprietary in nature.

Evaluation Technique: We compared the *stochastic policies* on bandit datasets. Comparison of policies on bandit test set comes with the following unique challenge: for a stochastic policy π, the expected reward is $\rho(\pi) = \mathbb{E}_{a \sim \pi(\cdot|x)} r_a = \sum_a r_a \pi(a|x)$, for a test context x (with the true CTR of π being average of expected reward over entire test set).

Since the bandit form of test data does not give any information about rewards for offers which were not displayed, it is not possible to calculate the expected reward!

We evaluated the policies using a particular offline contextual bandit policy evaluation technique. There exist various such evaluation techniques in the literature, with adequate discussion about the process (*Li et al.*, 2011). We used one of the importance weighted techniques as described in (*Theocharous et al.*, 2015). The reason was that we could give high confidence lower bound on the performance of the policies. We provide the mathematical details of the technique.

The bandit test data was logged from the interaction between users and a fully random policy π_u, over an interaction window. The random policy produced the following distribution over offers: $\pi_u(a|x) = \frac{1}{K}, \forall\ a \in [K]$. For an instance $\{x, a', r_{a'}\}$ in the test set, the importance weighted reward of evaluation policy π is computed as $\hat{\rho}(\pi) = r_{a'}\frac{\pi(a'|x)}{\pi_u(a'|x)}$. The importance weighted reward is an unbiased estimator of the true expected reward of π, i.e., $\mathbb{E}_{a'\sim\pi_u(\cdot|x)}\hat{\rho}(\pi) = \rho(\pi)$.

Let the cardinality of the test set be n. The **importance weighted CTR of** π is defined as

$$\frac{1}{n}\sum_{i=1}^{n} r_{a_i}\frac{\pi(a_i|x_i)}{\pi_u(a_i|x_i)} \tag{5.4}$$

Since (x, r) are assumed to be generated i.i.d., the importance weighted CTR is an unbiased estimator of the true CTR of π. Moreover, it is possible to construct a *t-test* based **lower confidence bound** on the expected reward, using the unbiased estimator, as follows: let $X_i = r_{a_i}\frac{\pi(a_i|x_i)}{\pi_u(a_i|x_i)}$, $\hat{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, and $\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \hat{X})^2}$. Then, $\hat{X} =$ importance weighted CTR and

$$\hat{X} - \frac{\sigma}{\sqrt{n}}t_{1-\delta,n-1} \tag{5.5}$$

is a $1-\delta$ **lower confidence bound** on the true CTR. Thus, during evaluation, we plotted the importance weighted CTR and lower confidence bounds for the competing policies.

Table 5.1: All datasets were obtained from UCI repository (https://archive.ics.uci.edu/ml/). Five different datasets were selected. In the table, size represents the number of examples in the complete dataset. Features indicate the dimension of the instances. Avg. % positive gives the number of positive instances per class, divided by the total number of instances for that class, in the bandit training set, averaged over all classes. The lower the value, the more is the imbalance per class during training.

	OptDigits	Isolet	Letter	PenDigits	Movementlibras
Size	5620	7797	20000	10992	360
Features	64	617	16	16	91
Classes	10	26	26	10	15
Avg. % positive	10	4	4	10	7

5.6 Empirical Results

We detail the parameters and results of our experiments.

Linear Classifiers and Ranker: For each ad a, a linear classifier was learnt by optimizing the logistic surrogate, while a linear ranker was learnt by optimizing the objective function (5.1), with $\ell(\cdot)$ being the logistic surrogate. Since we did not have the problem of sparse high-dimensional features in our datasets, we added an ℓ_2 regularizer instead of ℓ_1 regularizer. We applied SGD with1 million iterations; varied the parameter λ of the ℓ_2 regularizer in the set $\{0.01, 0.1, 1, 10\}$ and recorded the best result.

Ensemble Classifiers: We learnt a RandomForest classifier for each ad a. The RandomForests were composed of 200 trees, both for computational feasibility and for the more theoretical reason outlined in (*Koh and Gupta*, 2014).

5.6.1 Comparison of Deterministic Policies

Datasets: The multi-class datasets are detailed in Table 5.1.

Evaluation: To compare the deterministic policies , we conducted two sets of experiments; one without under-sampling of negative classes during training (i.e., no class balancing) and another with heavy under-sampling of negative classes (artificial class balancing).

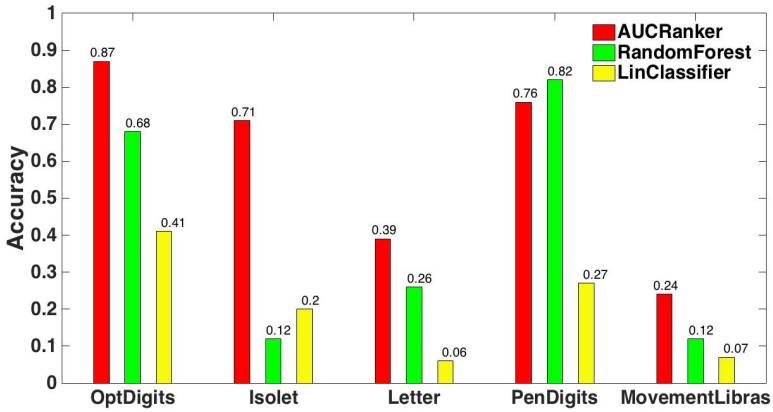
Training and testing were repeated 10 times for each dataset to account for the randomness introduced during conversion of classification training data to bandit training data, and the average accuracy over the runs are reported. Figure 5.1 top and bottom show performance of various policies learnt without and with under-sampling during training, respectively. Under-sampling was done to make positive:negative ratio as 1:2 for every class (this basically means that Avg $\%$ positive was 33%). *The ratio of 1:2 generally gave the best results*.

Observations: **a**) With heavy under-sampling, the performance of classifier-based policies improve significantly during training. Ranker-based policy is not affected, validating that class imbalance does not affect ranking loss, **b)** The linear ranker-based policy performs uniformly better than the linear classifier-based policy, with or without under-sampling. This shows that restricting to same class of functions (linear), rankers handles class-imbalance much better than classifiers **c)** The linear ranker-based policy does better than more complex RandomForest (RF) based policy, when no under-sampling is done during training, and is competitive when under-sampling is done, and **d)** Complex classifiers like RFs are relatively robust to moderate class imbalance. However, as we will see in real datasets, when class imbalance is extreme, gain from using a ranker-based policy becomes prominent. Moreover, growing big forests may be infeasible due to memory constraints,

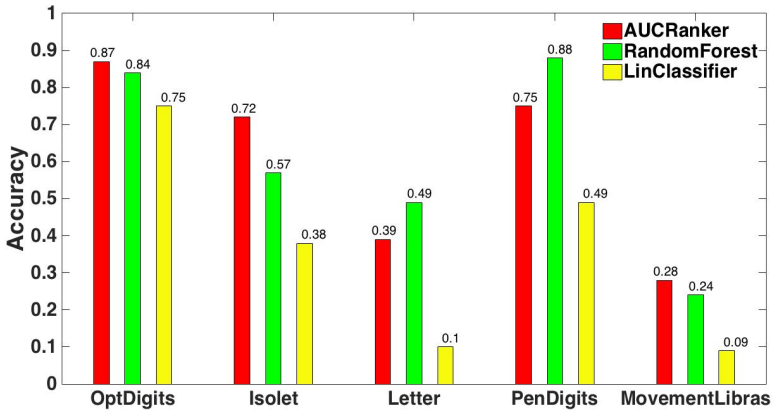
5.6.2 Comparison of Stochastic Policies

Our next set of experiments were conducted on three different datasets that are property of a major technology company.

Datasets: Two of the datasets were collected from campaigns run by two major banks and another from campaign run by a major hotel chain. When a user visited the campaign website, she was either assigned to a targeted policy or a purely random policy. The targeted policy was some specific ad serving policy, particular to the campaign. The data was collected in the form $\{x, a, r_a\}$, where x denotes the user context, a denotes the offer dis-



(a) Hotel- **moderate click sparsity**



(b) Bank 1- **extreme click sparsity**

Figure 5.1: Comparison of classification accuracy of various policies for different datasets, Top- without under-sampling and Bottom- with under-sampling.

played, and $r_a \in \{0, 1\}$ denotes the reward received. We trained our competing policies on data collected from the targeted policy and testing was done on the data collected from the random policy. We focused on the top-5 offers by number of impressions in the training set. Table 5.2 provides information about the training sets collected from the hotel and one of the bank's campaigns. The second bank's campaign has similar training set as the first one. *As can be clearly observed, each offer in the bank's campaign suffers from extreme click sparsity*.

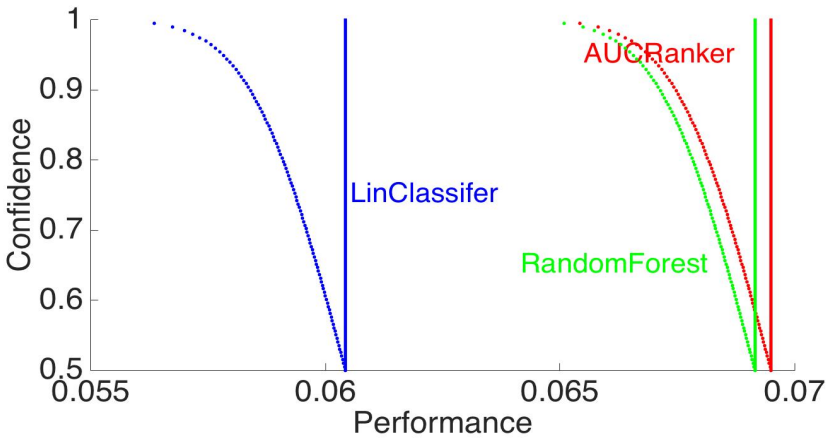
Table 5.2: Information about the training sets

Domain	Offer	Impressions	Avg. % Positive (Clicks/Impressions)
Hotel			
	1	36164	8.1
	2	37944	8.2
	3	30871	7.8
	4	32765	7.7
	5	20719	5.5
Bank 1			
	1	37750	0.17
	2	38254	0.40
	3	182191	0.45
	4	168789	0.30
	5	17291	0.23

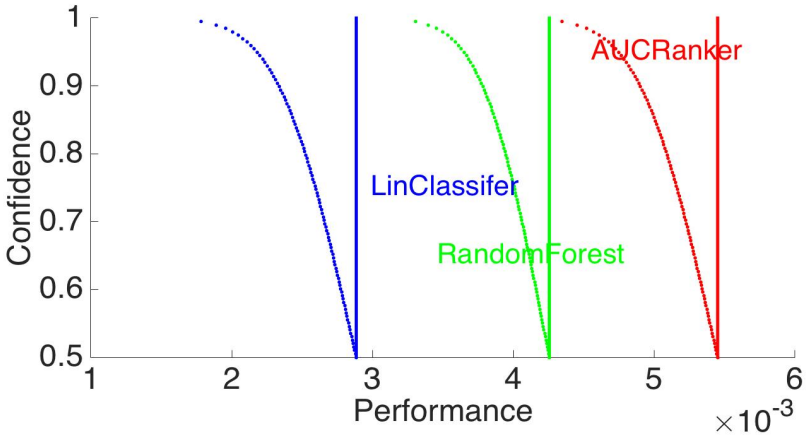
Feature Selection: We used a feature selection strategy to select around 20% of the users' features, as some of the features were of poor quality and led to difficulty in learning. We used the *information gain* criteria to select features (*Cheng et al.*, 2012).

Results: Figures 5.2(a), 5.2(b), and 5.2(c) show the results of our experiments. We used heavy under-sampling of negative examples, at the ratio 1:1 for positive:negative examples per offer, while training the classifiers. During evaluation, $\epsilon = 0.2$ was used as exploration parameter. Taking $\epsilon < 0.2$, meaning more exploitation, did not yield better results.

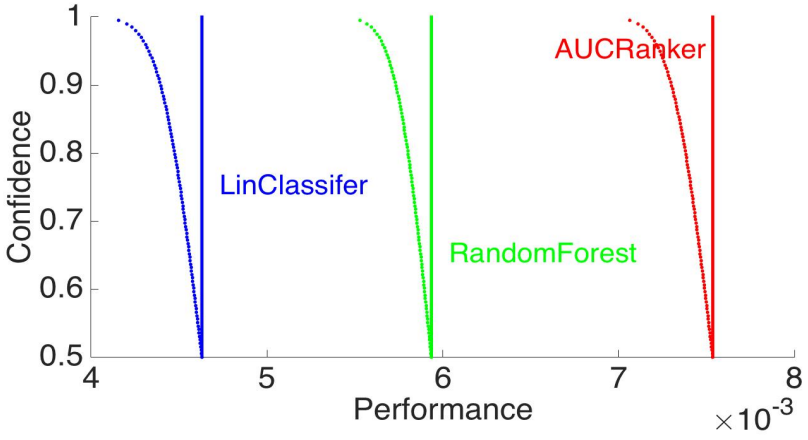
Observations: **a)** The ranker-based policy generally performed better than the classifier-based policies. **b)** For the bank campaigns, *where the click sparsity problem is extremely severe, it can be stated with high confidence that the ranker-based policy performed significantly better than classifier based policies*. This shows that the ranker-based policy can handle class imbalance better than the classifier policies.



(a) Hotel- **moderate click sparsity**



(b) Bank 1- **extreme click sparsity**



(c) Bank 2- **extreme click sparsity**

Figure 5.2: Importance weighted CTR and lower confidence bounds for policies. Ranking based approach gains 15-25 % in importance weighted CTR over RF based approach on data with extreme click sparsity. Classifiers were trained after class balancing.

APPENDICES

APPENDIX A

Proof(s) of Chapter II

Proof of Theorem.4

Proof for AP: As stated previously, documents pertaining to every query are sorted according to relevance labels. We point out another critical property of AP (for that matter any ranking measure). AP is only affected when scores of 2 documents, which have different relevance levels, are not consistent with the relevance levels, as long as ranking is obtained by sorting scores in descending order. That is, if $R_i = R_j$, then it does not matter whether $s_i > s_j$ or $s_i < s_j$. So, without loss of generality, we can always assume that within same relevance class, the documents are sorted according to scores. That is, if $R_i = R_j$ with $i < j$, then $s_i \geq s_j$. The without loss of generality holds because SLAM is calculated with knowledge of relevance and score vector. Thus, within same relevance class, we can sort the documents according to their scores (effectively exchanging document identities), without affecting SLAM loss.

Let $R \in \mathbb{R}^m$ be an arbitrary binary relevance vector, with r relevant documents and $m-r$ irrelevant documents in a list. AP loss is only incurred if at least 1 irrelevant document is placed above at least 1 relevant document. With reference to ϕ^v_{SLAM} in Eq. (2.9), for any $i \geq r+1$ and $\forall\, j > i$, we have $\mathbb{1}(R_i > R_j) = 0$, since $R_i = R_j = 0$. For any $i \geq r+1$ and $\forall\, j < i$, $\mathbb{1}(R_i > R_j) = 0$ since documents are sorted according to relevance labels and

$R_i = 0, R_j = 1$. Thus, w.l.o.g., we can take $v_{r+1}, ..., v_m = 0$, since indicator in SLAM loss will never turn on for $i \geq r + 1$.

Let a score vector s be such that an irrelevant document j has the highest score among m documents. Then, $\phi^v_{SLAM} = v_1(1 + s_j - s_1) + v_2(1 + s_j - s_2) + ... + v_r(1 + s_j - s_r)$. The maximum possible AP induced loss in case at least one irrelevant document has higher score than all relevant documents is when all irrelevant documents outscore all relevant documents. The AP loss in that case is: $1 - \frac{1}{r}(\frac{1}{m-r+1} + \frac{2}{m-r+2} + .. + \frac{r}{m-r+r})$. Since ϕ^v_{SLAM} has to upper bound AP $\forall s$ (for each R) and since s_j can be infinitesimally greater than all other score components (thus, $1 + s_j - s_i \sim 1, \ \forall \ i = 1, \ldots, r$), we need the following equation for upper bound property to hold:

$v_1 + v_2 + ... + v_r \geq 1 - \frac{1}{r}(\frac{1}{m-r+1} + \frac{2}{m-r+2} + .. + \frac{r}{m-r+r})$.

Similarly, let a score vector s be such that an irrelevant document j has higher score than all but the 1st relevant document. Then $\phi^v_{SLAM} = v_2(1 + s_j - s_2) + v_3(1 + s_j - s_3) + ... + v_r(1 + s_j - s_r)$. The maximum possible AP induced loss in this case occurs when all irrelevant documents are placed above all relevant documents except the first relevant document. The AP loss in that case is: $1 - \frac{1}{r}(1 + \frac{2}{m-r+2} + \frac{3}{m-r+3} + .. + \frac{r}{m-r+r})$. Following same line of logic for upper bounding as before, we get

$v_2 + v_3 + ... + v_r \geq 1 - \frac{1}{r}(1 + \frac{2}{m-r+2} + \frac{3}{m-r+3} + .. + \frac{r}{m-r+r})$.

Likewise, if we keep repeating the logic, we get sequence of inequalities, with the last inequality being

$v_r \geq 1 - \frac{1}{r}(r - 1 + \frac{r}{m-r+r})$.

Now, it can be easily seen that our definition of v^{AP} satisfies the inequalities.

Proof for NDCG:

We once again remind that $\pi^{-1}(i)$ means position of document i in permutation π. Thus, if document i is placed at position j in π, then $\pi^{-1}(i) = j$. Moreover, like AP, we assume that $R_1 \geq R_2 \geq \ldots \geq R_m$ and that within same relevance class, documents are sorted according to score. We have a modified definition of NDCG, for $k = m$, which is

required for the proof:

$$\text{NDCG}(s, R) = \frac{1}{Z(R)} \sum_{i=1}^{m} G(R_i) D(\pi_s^{-1}(i)) \tag{A.1}$$

where $G(r) = 2^r - 1$, $D(i) = \frac{1}{\log_2(i+1)}$, $Z(R) = \max_{\pi} \sum_{i=1}^{m} G(R_i) D(\pi^{-1}(i))$.
We begin the proof:

$$\begin{aligned}
& 1 - \text{NDCG}(s, R) \\
&= \frac{1}{Z(R)} \sum_{i=1}^{m} G(R_i) D(i) - \frac{1}{Z(R)} \sum_{i=1}^{m} G(R_i) D(\pi_s^{-1}(i)) \\
&= \frac{1}{Z(R)} \sum_{i=1}^{m} G(R_i) \left(D(i) - D(\pi_s^{-1}(i)) \right)
\end{aligned}$$

Now, $D(i) = \frac{1}{\log_2(1+i)}$ is a decreasing function of i. $D(i) - D(\pi_s^{-1}(i))$ is positive only if $i < \pi_s^{-1}(i)$. This means that document i in the original list, is placed at position $\pi_s^{-1}(i)$, which comes after i, by sorted order of score vector s. By the assumption that indices of documents within same relevance class are sorted according to their scores, this means that document i is outscored by another document (say with index k) with lower relevance level. At that point, the function $\max(0, \max_{j=1,\ldots,m} \{\mathbb{1}(R_i > R_j)(1 + s_j - s_i)\})$ turns on with value at least 1 (i.e., $(1 + s_k - s_i > 1)$) and with weight vector $v_i^{\text{NDCG}} = \frac{G(R_i)D(i)}{Z(R)}$. We can now easily see the upper bound property.

APPENDIX B

Proof(s) of Chapter III

Proof of Theorem 15:

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 3.1 are neighboring actions, using Lemma 14 . 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 3.1 and 3.2, we have,

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

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

Proof of Theorem 19:

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

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

(dot product between 2 vectors of length $m!$).

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

$$\begin{aligned}
\mathbb{E}\sum_{t=1}^{T}[SumLoss(\sigma_t, R_t)] &= \sum_{i=1}^{K}\mathbb{E}_{\sigma_t\sim\rho_i}[\sum_{t\in B_i} SumLoss(\sigma_t, R_t)] \\
&= \sum_{i=1}^{K}\sum_{t\in B_i}\rho_i \cdot [SumLoss(\sigma_1, R_t), \ldots, SumLoss(\sigma_{m!}, R_t)] \\
&= \sum_{i=1}^{K}\sum_{t\in B_i}\rho_i \cdot [\sigma_1^{-1}\cdot R_t, \ldots, \sigma_{m!}^{-1}\cdot R_t)] \text{ By defn. of SumLoss} \\
&= \frac{T}{K}\sum_{i=1}^{K}\rho_i \cdot [\sigma_1^{-1}\cdot R_i^{avg}, \ldots, \sigma_{m!}^{-1}\cdot R_i^{avg}] \\
&= \frac{T}{K}\sum_{i=1}^{K}\mathbb{E}_{\sigma_i\sim\rho_i}[SumLoss(\sigma_i, R_i^{avg})] \\
&= \frac{T}{K}\mathbb{E}_{\sigma_1\sim\rho_1,\ldots,\sigma_K\sim\rho_K}\sum_{i=1}^{K} SumLoss(\sigma_i, R_i^{avg}) \qquad \text{(B.2)}
\end{aligned}$$

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:

$$\begin{aligned}
\mathbb{E}_{\sigma_1\sim\rho_1,\ldots,\sigma_K\sim\rho_K}\sum_{i=1}^{K} SumLoss(\sigma_i, R_i^{avg}) &\leq \min_{\sigma}\sum_{i=1}^{K} SumLoss(\sigma, R_i^{avg}) + C\sqrt{K} \\
&= \min_{\sigma}\sum_{i=1}^{K}\sigma^{-1}\cdot R_i^{avg} + C\sqrt{K} \qquad \text{(B.3)} \\
&= \min_{\sigma}\sum_{t=1}^{T}\sigma^{-1}\cdot\frac{R_t}{T/K} + C\sqrt{K}
\end{aligned}$$

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. B.2 and Eq. B.3, we get:

$$\sum_{t=1}^{T} \mathbb{E}_{\sigma_t \in \rho_i}[SumLoss(\sigma_t, R_t)] \leq \min_{\sigma} \sum_{t=1}^{T} SumLoss(\sigma, R_t) + C\frac{T}{\sqrt{K}}. \tag{B.4}$$

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 25. 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 4). 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:

$$\begin{aligned}
&\mathbb{E}_{\sigma_t \sim \rho_i(\hat{R}_1, \hat{R}_2, .., \hat{R}_{i-1})} \sum_{t \in [B_i]} SumLoss(\sigma_t, R_t) \\
&\qquad = \frac{T}{K} \mathbb{E}_{\sigma_i \sim \rho_i(\hat{R}_1, \hat{R}_2, .., \hat{R}_{i-1})} SumLoss(\sigma_i, R_i^{avg}) \\
&\qquad \text{(From Eq. B.2)} \\
&\qquad = \frac{T}{K} \mathbb{E}_{\sigma_i \sim \rho_i(\hat{R}_1, \hat{R}_2, .., \hat{R}_{i-1})} \mathbb{E}_{\hat{R}_i} SumLoss(\sigma_i, \hat{R}_i) \\
&\qquad (\because \text{SumLoss is linear in both arguments and } \hat{R}_i \text{ is unbiased}) \\
&\qquad = \frac{T}{K} \mathbb{E}_{\hat{R}_i} \mathbb{E}_{\sigma_i \sim \rho_i(\hat{R}_1, \hat{R}_2, .., \hat{R}_{i-1})} SumLoss(\sigma_i, \hat{R}_i).
\end{aligned}$$

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,

$$\mathbb{E}_{\hat{R}_1,...,\hat{R}_{i-1}} \mathbb{E}_{\sigma_t \sim \rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})} \sum_{t \in [B_i]} SumLoss(\sigma_t, R_t) = \frac{T}{K} \mathbb{E}_{\hat{R}_1,...,\hat{R}_{i-1},\hat{R}_i} E_{\sigma_i \sim \rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})} SumLoss(\sigma_i, \hat{R}_i). \tag{B.5}$$

Thus,

$$\begin{aligned}
\mathbb{E} \sum_{t=1}^{T} SumLoss(\sigma_t, R_t) &= \mathbb{E} \sum_{i=1}^{K} \sum_{t \in [B_i]} SumLoss(\sigma_t, R_t) \\
&= \sum_{i=1}^{K} E_{\hat{R}_1,...,\hat{R}_{i-1}} E_{\sigma_t \sim \rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})} \sum_{t \in [B_i]} SumLoss(\sigma_t, R_t) \\
&= \frac{T}{K} \sum_{i=1}^{K} \mathbb{E}_{\hat{R}_1,...,\hat{R}_{i-1},\hat{R}_i} \mathbb{E}_{\sigma_i \sim \rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})} SumLoss(\sigma_i, \hat{R}_i) \\
&\quad \text{(From Eq. B.5)} \\
&= \frac{T}{K} \mathbb{E}_{\hat{R}_1,...,\hat{R}_K} \mathbb{E}_{\sigma_i \sim \rho_i(\hat{R}_1,\hat{R}_2,..,\hat{R}_{i-1})} \sum_{i=1}^{K} SumLoss(\sigma_i, \hat{R}_i)
\end{aligned}$$

Now using Eq. B.3, 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 :

$$\mathbb{E}\left[\sum_{t=1}^{T} SumLoss(\sigma_t, R_t)\right] - \min_{\sigma}\sum_{t=1}^{T} SumLoss(\sigma, R_t) \leq C^I\lceil m/k\rceil K + C\frac{T}{\sqrt{K}}. \quad \text{(B.6)}$$

Now we optimize over K, to get $K = C^{II}\dfrac{T^{2/3}}{\lceil m/k\rceil^{2/3}}$, and finally:

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

□

Proof of Corollary 20:

Proof. We only need to instantiate the constants C, C^I, C^{II} and C^{III}, from Theorem 19, 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 $O(\sqrt{T})$ regret over T rounds when $\epsilon = \sqrt{\frac{D}{RAT}}$. Then, $C = 2\sqrt{DRA}$, $C^I = R, C^{II} = \dfrac{(DA)^{1/3}}{R^{1/3}}, C^{III} = (DA)^{1/3}R^{2/3}$. Substituting the values of D, R, A, we conclude. □

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 give pointers in the in proving 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 4 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 13, 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 14 shows that it is directly applicable to DCG, in light of extension of Lemma 13 to DCG.

Finally, just like in SumLoss, simple calculations with $m = 3$ and $n = 1$, in light of Lemma 13 and 14, show that local observability condition fails to hold.

We show the calculations:

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

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

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

Proof of Corrolary 21

For DCG, the parameters of FTPL are: $R = \sum_{i=1}^m f^s(\sigma^{-1}(i))g^s(R(i)) = O(m(2^n - 1)), D = \sum_{i=1}^m f^s(\sigma^{-1}(i)) = O(m), A = \sum_{i=1}^m g^s(R(i)) = O(m(2^n - 1))$. The proof follows from substituting the values in C^{III} (can be followed from proof of Corollary 20).

Precision@n:

Proof of Corrolary 22

For Precision@n, the parameters of FTPL are: $D = \sum_{i=1}^m f^s(\sigma^{-1}(i)) = O(n), R = \sum_{i=1}^m f^s(\sigma^{-1}(i))g^s(R(i)) = O(n), A = \sum_{i=1}^m g^s(R(i)) = O(m)$. The proof follows from substituting the values in C^{III} (can be followed from proof of Corollary 20).

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.3.2.2). 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 23

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

NDCG:

The first and last row of Table 3.1, 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 3.1, 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 3.1 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.

Proof of Lemma 25: We restate the lemma before giving the proof, for ease of reading:

Lemma 25: 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 3.9 , with $\ell = k$, an unbiased estimator of $F(x)$, based on

$\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$, is,

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

where S_k is the set of $k!$ permutations of [k] and $p(\sigma(1), \ldots, \sigma(k))$ is as in Eq 3.8 .

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. 25 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. 25 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 29 :

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 28 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 5)

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 5)).

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. 5)).

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 26 :

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 27 :

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. 32:

Proof. We will first fix the setting of the online game. We consider $m = 3$ and fixed the document matrix $X \in \mathbb{R}^{3\times3}$ 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 $s_p^* = \operatorname{argmin}_s \mathbb{E}_{R\sim p}\phi(s, R)$. 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* (2001b) 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 $\forall j \in [m]$,

Table B.1: 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

$\mathbb{P}_{R\sim p}(R(j)=1) = \mathbb{P}_{R\sim\tilde{p}}(R(j)=1)$ (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 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 31, 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 $\operatorname{argsort}\left(\mathbb{E}_{R\sim p}\left[\frac{G(\mathbf{R})}{Z(R)}\right]\right) \neq \operatorname{argsort}\left(\mathbb{E}_{R\sim\tilde{p}}\left[\frac{G(\mathbf{R})}{Z(R)}\right]\right)$, 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. B.1.

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

□

APPENDIX C

Proof(s) of Chapter IV

Proof of Proposition 1

Proof. Let e_j's denote standard basis vectors. We have

$$\nabla_s \phi_{\mathrm{LN}}(s, y) = -\sum_{j=1}^{m} P_j(y) e_j + \sum_{j=1}^{m} \frac{\exp(s_j)}{\sum_{j'=1}^{m} \exp(s_{j'})} e_j$$

Therefore,

$$\begin{aligned} \|\nabla_s \phi_{\mathrm{LN}}(s, y)\|_1 &\leq \sum_{j=1}^{m} P_j(y) \|e_j\|_1 + \sum_{j=1}^{m} \frac{\exp(s_j)}{\sum_{j'=1}^{m} \exp(s_{j'})} \|e_j\|_1 \\ &= 2. \end{aligned}$$

We also have

$$[\nabla_s^2 \phi_{\mathrm{LN}}(s, y)]_{j,k} = \begin{cases} -\frac{\exp(2s_j)}{(\sum_{j'=1}^{m} \exp(s_{j'}))^2} + \frac{\exp(s_j)}{\sum_{j'=1}^{m} \exp(s_{j'})} & \text{if } j = k \\ -\frac{\exp(s_j + s_k)}{(\sum_{j'=1}^{m} \exp(s_{j'}))^2} & \text{if } j \neq k \,. \end{cases}$$

Moreover,

$$\begin{aligned}
\|\nabla_s^2 \phi_{\mathrm{LN}}(s,y)\|_{\infty\to 1} &\leq \sum_{j=1}^{m}\sum_{k=1}^{m} |[\nabla_s^2 \phi_{\mathrm{LN}}(s,y)]_{j,k}| \\
&\leq \sum_{j=1}^{m}\sum_{k=1}^{m} \frac{\exp(s_j+s_k)}{(\sum_{j'=1}^{m}\exp(s_{j'}))^2} + \sum_{j=1}^{m}\frac{\exp(s_j)}{\sum_{j'=1}^{m}\exp(s_{j'})} \\
&= \frac{(\sum_{j=1}^{m}\exp(s_j))^2}{(\sum_{j'=1}^{m}\exp(s_{j'}))^2} + \frac{\sum_{j=1}^{m}\exp(s_j)}{\sum_{j'=1}^{m}\exp(s_{j'})} \\
&= 2
\end{aligned}$$

□

Proof of Proposition 2

Proof. Let $1_{(\text{condition})}$ denote an indicator variable. We have

$$[\nabla_s \phi_{\mathrm{SD}}(s,y)]_j = D(1)\left(\sum_{i=1}^{m} G(r_i)\left[\frac{1}{\sigma}\frac{\exp(s_i/\sigma)}{\sum_{j'}\exp(s_{j'}/\sigma)}1_{(i=j)} - \frac{1}{\sigma}\frac{\exp((s_i+s_j)/\sigma)}{(\sum_{j'}\exp(s_{j'}/\sigma))^2}\right]\right)$$

Therefore,

$$\begin{aligned}
\frac{\|\nabla_s \phi_{\mathrm{SD}}(s,y)\|_1}{D(1)G(Y_{\max})} &\leq \sum_{j=1}^{m}\left(\sum_{i=1}^{m}\left[\frac{1}{\sigma}\frac{\exp(s_i/\sigma)}{\sum_{j'}\exp(s_{j'}/\sigma)}1_{(i=j)} + \frac{1}{\sigma}\frac{\exp((s_i+s_j)/\sigma)}{(\sum_{j'}\exp(s_{j'}/\sigma))^2}\right]\right) \\
&= \frac{1}{\sigma}\left(\frac{\sum_j \exp(s_j/\sigma)}{\sum_{j'}\exp(s_{j'}/\sigma)} + \frac{(\sum_j \exp(s_j/\sigma))^2}{(\sum_{j'}\exp(s_{j'}/\sigma))^2}\right) \\
&= \frac{2}{\sigma}.
\end{aligned}$$

□

RankSVM

The RankSVM surrogate is defined as:

$$\phi_{RS}(s,y) = \sum_{i=1}^{m}\sum_{j=1}^{m}\max(0, 1_{(y_i>y_j)}(1+s_j-s_i))$$

It is easy to see that $\nabla_s \phi_{RS}(s, y) = \sum_{i=1}^{m} \sum_{j=1}^{m} \max(0, 1_{(y_i > y_j)}(1 + s_j - s_i))(e_j - e_i)$. Thus, the ℓ_1 norm of gradient is $O(m^2)$.

Proof of Theorem 33

Proof. It is straightforward to check that $\mathcal{F}'_{\text{lin}}$ is contained in both $\mathcal{F}_{\text{full}}$ as well as $\mathcal{F}_{\text{perminv}}$. So, we just need to prove that any f that is in both $\mathcal{F}_{\text{full}}$ and $\mathcal{F}_{\text{perminv}}$ has to be in $\mathcal{F}'_{\text{lin}}$ as well.

Let P_π denote the $m \times m$ permutation matrix corresponding to a permutation π. Consider the full linear class $\mathcal{F}_{\text{full}}$. In matrix notation, the permutation invariance property means that, for any π, X, we have $P_\pi[\langle X, W_1\rangle, \ldots, \langle X, W_m\rangle\rangle]^\top = [\langle P_\pi X, W_1\rangle, \ldots, \langle P_\pi X, W_m\rangle\rangle]^\top$.

Let $\rho_1 = \{P_\pi : \pi(1) = 1\}$, where $\pi(i)$ denotes the index of the element in the ith position according to permutation π. Fix any $P \in \rho_1$. Then, for any X, $\langle X, W_1\rangle = \langle PX, W_1\rangle$. This implies that, for all X, $\text{Tr}({W_1}^\top X) = \text{Tr}({W_1}^\top PX)$. Using the fact that $\text{Tr}(A^\top X) = \text{Tr}(B^\top X), \forall X$ implies $A = B$, we have that ${W_1}^\top = {W_1}^\top P$. Because $P^\top = P^{-1}$, this means $PW_1 = W_1$. This shows that all rows of W_1, other than 1st row, are the same but perhaps different from 1st row. By considering $\rho_i = \{P_\pi : \pi(i) = i\}$ for $i > 1$, the same reasoning shows that, for each i, all rows of W_i, other than ith row, are the same but possibly different from ith row.

Let $\rho_{1\leftrightarrow 2} = \{P_\pi : \pi(1) = 2, \pi(2) = 1\}$. Fix any $P \in \rho_{1\leftrightarrow 2}$. Then, for any X, $\langle X, W_2\rangle = \langle PX, W_1\rangle$ and $\langle X, W_1\rangle = \langle PX, W_2\rangle$. Thus, we have $W_2^\top = W_1^\top P$ as well as $W_1^\top = W_2^\top P$ which means $PW_2 = W_1, PW_1 = W_2$. This shows that row 1 of W_1 and row 2 of W_2 are the same. Moreover, row 2 of W_1 and row 1 of W_2 are the same. Thus, for some $u, u' \in \mathbb{R}^d$, W_1 is of the form $[u|u'|u'|\ldots|u']^\top$ and W_2 is of the form $[u'|u|u'|\ldots|u']^\top$. Repeating this argument by considering $\rho_{1\leftrightarrow i}$ for $i > 2$ shows that W_i is of the same form (u in row i and u' elsewhere).

Therefore, we have proved that any linear map that is permutation invariant has to be

of the form:

$$X \mapsto \left(u^\top X_i + (u')^\top \sum_{j \neq i} X_j \right)_{i=1}^m .$$

We can reparameterize above using $w = u - u'$ and $v = u'$ which proves the result. □

Proof of Lemma 34

Proof. The first equality is true because

$$\begin{aligned} \|X^\top\|_{1\to p} &= \sup_{v\neq 0} \frac{\|X^\top v\|_p}{\|v\|_1} = \sup_{v\neq 0} \sup_{u\neq 0} \frac{\langle X^\top v, u\rangle}{\|v\|_1 \|u\|_q} \\ &= \sup_{u\neq 0} \sup_{v\neq 0} \frac{\langle v, Xu\rangle}{\|v\|_1 \|u\|_q} = \sup_{u\neq 0} \frac{\|Xu\|_\infty}{\|u\|_q} = \|X\|_{q\to\infty}. \end{aligned}$$

The second is true because

$$\begin{aligned} \|X\|_{q\to\infty} &= \sup_{u\neq 0} \frac{\|Xu\|_\infty}{\|u\|_q} = \sup_{u\neq 0} \max_{j=1}^m \frac{|\langle X_j, u\rangle|}{\|u\|_q} \\ &= \max_{j=1}^m \sup_{u\neq 0} \frac{|\langle X_j, u\rangle|}{\|u\|_q} = \max_{j=1}^m \|X_j\|_p. \end{aligned}$$

□

Proof of Theorem 36 Our theorem is developed from the "expectation version" of Theorem 6 of *Shalev-Shwartz et al.* (2009) that was originally given in probabilistic form. The expected version is as follows.

Let $\mathcal{Z}$ be a space endowed with a probability distribution generating iid draws $Z_1, \ldots, Z_n$. Let $\mathcal{W} \subseteq \mathbb{R}^d$ and $f : \mathcal{W} \times \mathcal{Z} \to \mathbb{R}$ be λ-strongly convex[1] and G-Lipschitz (w.r.t. $\|\cdot\|_2$) in w for every z. We define $F(w) = \mathbb{E} f(w, Z)$ and let

$$\begin{aligned} w^\star &= \operatorname*{argmin}_{w\in\mathcal{W}} \; F(w), \\ \hat{w} &= \operatorname*{argmin}_{w\in\mathcal{W}} \; \frac{1}{n} \sum_{i=1}^n f(w, Z_i). \end{aligned}$$

[1]Recall that a function is called λ-strongly convex (w.r.t. $\|\cdot\|_2$) iff $f - \frac{\lambda}{2}\|\cdot\|_2^2$ is convex.

Then $\mathbb{E}F(\hat{w}) - F(w^\star) \leq \frac{4G^2}{\lambda n}$, where the expectation is taken over the sample. The above inequality can be proved by carefully going through the proof of Theorem 6 proved by *Shalev-Shwartz et al.* (2009).

We now derive the "expectation version" of Theorem 7 of *Shalev-Shwartz et al.* (2009). Define the regularized empirical risk minimizer as follows:

$$\hat{w}_\lambda = \underset{w \in \mathcal{W}}{\operatorname{argmin}} \ \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} f(w, Z_i). \tag{C.1}$$

The following result gives optimality guarantees for the regularized empirical risk minimizer.

Theorem 45. *Let $\mathcal{W} = \{w \ : \ \|w\|_2 \leq W_2\}$ and let $f(w,z)$ be convex and G-Lipschitz (w.r.t. $\|\cdot\|_2$) in w for every z. Let $Z_1, ..., Z_n$ be iid samples and let $\lambda = \sqrt{\frac{\frac{4G^2}{n}}{\frac{W_2^2}{2} + \frac{4W_2^2}{n}}}$. Then for $\hat{w}_\lambda$ and $w^\star$ as defined above, we have*

$$\mathbb{E}F(\hat{w}_\lambda) - F(w^\star) \leq 2\,G\,W_2\left(\frac{8}{n} + \sqrt{\frac{2}{n}}\right). \tag{C.2}$$

Proof. Let $r_\lambda(w,z) = \frac{\lambda}{2}\|w\|_2^2 + f(w,z)$. Then r_λ is λ-strongly convex with Lipschitz constant $\lambda W_2 + G$ in $\|\cdot\|_2$. Applying "expectation version" of Theorem 6 of *Shalev-Shwartz et al.* (2009) to r_λ, we get

$$\mathbb{E}\frac{\lambda}{2}\|\hat{w}_\lambda\|_2^2 + F(\hat{w}_\lambda) \leq \min_{w \in \mathcal{W}}\left\{\frac{\lambda}{2}\|w\|_2^2 + F(w)\right\} + \frac{4(\lambda W_2 + G)^2}{\lambda n} \leq \frac{\lambda}{2}\|w^\star\|_2^2 + F(w^*) + \frac{4(\lambda W_2 + G)^2}{\lambda n}.$$

Thus, we get

$$\mathbb{E}F(\hat{w}_\lambda) - F(w^\star) \leq \frac{\lambda W_2^2}{2} + \frac{4(\lambda W_2 + G)^2}{\lambda n}.$$

Minimizing the upper bound w.r.t. λ, we get $\lambda = \sqrt{\frac{4G^2}{n}}\sqrt{\frac{1}{\frac{W_2^2}{2} + \frac{4W_2^2}{n}}}$. Plugging this choice back in the equation above and using the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ finishes the proof of Theorem 45. □

We now have all ingredients to prove Theorem 36.

Proof of Theorem 36. Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $f(w, z) = \phi(Xw, y)$ and apply Theorem 45. Finally note that if ϕ is G_ϕ-Lipschitz w.r.t. $\|\cdot\|_\infty$ and every row of $X \in \mathbb{R}^{m\times d}$ has Euclidean norm bounded by R_X then $f(\cdot, z)$ is $G_\phi R_X$-Lipschitz w.r.t. $\|\cdot\|_2$ in w. □

Proof of Theorem 40

Proof. Following exactly the same line of reasoning (reducing a sample of size n, where each prediction is $\mathbb{R}^m$-valued, to an sample of size mn, where each prediction is real valued) as in the beginning of proof of Proposition 3, we have

$$\mathcal{N}_\infty(\epsilon, \phi \circ \mathcal{F}_1, n) \leq \mathcal{N}_\infty(\epsilon/G_\phi, \mathcal{G}_1, mn). \tag{C.3}$$

Plugging in the following bound due to *Zhang* (2002, Corollary 5):

$$\log_2 \mathcal{N}_\infty(\epsilon/G_\phi, \mathcal{G}_1, mn) \leq \left\lceil \frac{288\, G_\phi^2\, W_1^2\, \bar{R}_X^2\, (2 + \ln d)}{\epsilon^2} \right\rceil \times \log_2 \left(2\lceil 8 G_\phi W_1 \bar{R}_X/\epsilon \rceil mn + 1\right)$$

into (C.3) respectively proves the result. □

Calculations involved in deriving Equation (4.8)

Plugging in the value of η from (4.7) into the expression

$$\frac{L_\phi(w^\star)}{(1 - 4\eta H)} + \frac{W_2^2}{2\eta(1 - 4\eta H)n}$$

yields (using the shorthand $L^\star$ for $L_\phi(w^\star)$)

$$L^\star + \frac{2HW_2L^\star}{\sqrt{4H^2W_2^2 + 2HL^\star n}} + \frac{W_2}{n}\left[\frac{4H^2W_2^2}{\sqrt{4H^2W_2^2 + 2HL^\star n}} + \sqrt{4H^2W_2^2 + 2HL^\star n} + 4HW_2\right]$$

Denoting HW_2^2/n by x, this simplifies to

$$L^\star + \frac{2\sqrt{x}L^\star + 4x\sqrt{x}}{\sqrt{4x + 2L^\star}} + \sqrt{x}\sqrt{4x + 2L^\star} + 4x.$$

Using the arithmetic mean-geometric mean inequality to upper bound the middle two terms gives

$$L^\star + 2\sqrt{2xL^\star + 4x^2} + 4x.$$

Finally, using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, we get our final upper bound

$$L^\star + 2\sqrt{2xL^\star} + 8x.$$

Calculation of smoothness constant

$$\begin{aligned}
&\|(X^{(i)})^\top \nabla_s^2 \phi(X^{(i)}w, y^{(i)}) X^{(i)}\|_{2\to 2} = \sup_{v \neq 0} \frac{\|(X^{(i)})^\top \nabla_s^2 \phi(X^{(i)}w, y^{(i)}) X^{(i)} v\|_2}{\|v\|_2} \\
&\leq \sup_{v \neq 0} \frac{\|(X^{(i)})^\top\|_{1\to 2} \|\nabla_s^2 \phi(X^{(i)}w, y^{(i)}) X^{(i)} v\|_1}{\|v\|_2} \\
&\leq \sup_{v \neq 0} \frac{\|(X^{(i)})^\top\|_{1\to 2} \cdot \|\nabla_s^2 \phi(X^{(i)}w, y^{(i)})\|_{\infty\to 1} \cdot \|X^{(i)} v\|_\infty}{\|v\|_2} \\
&\leq \sup_{v \neq 0} \frac{\|(X^{(i)})^\top\|_{1\to 2} \cdot \|\nabla_s^2 \phi(X^{(i)}w, y^{(i)})\|_{\infty\to 1} \cdot \|X^{(i)}\|_{2\to\infty} \cdot \|v\|_2}{\|v\|_2} \\
&\leq \left(\max_{j=1}^{m} \|X_j^{(i)}\| \right)^2 \cdot \|\nabla_s^2 \phi(X^{(i)}w, y^{(i)})\|_{\infty\to 1} \\
&\leq R_X^2 \|\nabla_s^2 \phi(X^{(i)}w, y^{(i)})\|_{\infty\to 1}.
\end{aligned}$$

Proof of Lemma 42

Proof. Consider the function

$$f(t) = \phi((1-t)s_1 + ts_2).$$

It is clearly non-negative. Moreover

$$\begin{aligned}
|f'(t_1) - f'(t_2)| &= |\left\langle \nabla_s \phi(s_1 + t_1(s_2 - s_1)) - \nabla_s \phi(s_1 + t_2(s_2 - s_1)), s_2 - s_1 \right\rangle| \\
&\leq |||\nabla_s \phi(s_1 + t_1(s_2 - s_1)) - \nabla_s \phi(s_1 + t_2(s_2 - s_1))|||_\star \cdot |||s_2 - s_1||| \\
&\leq H_\phi \left| t_1 - t_2 \right| |||s_2 - s_1|||^2
\end{aligned}$$

and therefore it is smooth with constant $h = H_\phi |||s_2 - s_1|||^2$. Appealing to Lemma 41 now gives

$$(f(1) - f(0))^2 \leq 6H_\phi |||s_2 - s_1|||^2 (f(1) + f(0))(1 - 0)^2$$

which proves the lemma since $f(0) = \phi(s_1)$ and $f(1) = \phi(s_2)$. □

Proof of Proposition 5

Proof. Let $w, w' \in \mathcal{F}_{\phi,2}(r)$. Using Lemma 42

$$\begin{aligned}
&\sum_{i=1}^n \frac{1}{n} \left(\phi(X^{(i)}w, y^{(i)}) - \phi(X^{(i)}w', y^{(i)})\right)^2 \\
&\leq 6H_\phi \sum_{i=1}^n \frac{1}{n} \left(\phi(X^{(i)}w, y^{(i)}) + \phi(X^{(i)}w', y^{(i)})\right) \\
&\quad \cdot \|X^{(i)}w - X^{(i)}w'\|_\infty^2 \\
&\leq 6H_\phi \cdot \max_{i=1}^n \|X^{(i)}w - X^{(i)}w'\|_\infty^2 \\
&\quad \cdot \sum_{i=1}^n \frac{1}{n} \left(\phi(X^{(i)}w, y^{(i)}) + \phi(X^{(i)}w', y^{(i)})\right) \\
&= 6H_\phi \cdot \max_{i=1}^n \|X^{(i)}w - X^{(i)}w'\|_\infty^2 \cdot \left(\hat{L}_\phi(w) + \hat{L}_\phi(w')\right) \\
&\leq 12H_\phi r \cdot \max_{i=1}^n \|X^{(i)}w - X^{(i)}w'\|_\infty^2.
\end{aligned}$$

where the last inequality follows because $\hat{L}_\phi(w) + \hat{L}_\phi(w') \leq 2r$.

This immediately implies that if we have a cover of the class $\mathcal{G}_2$ at scale $\epsilon/\sqrt{12H_\phi r}$

w.r.t. the metric

$$\max_{i=1}^{n} \max_{j=1}^{m} \left| \left\langle X_j^{(i)}, w \right\rangle - \left\langle X_j^{(i)}, w' \right\rangle \right|$$

then it is also a cover of $\mathcal{F}_{\phi,2}(r)$ w.r.t. $d_2^{Z^{(1:n)}}$. Therefore, we have

$$\mathcal{N}_2(\epsilon, \mathcal{F}_{\phi,2}(r), Z^{(1:n)}) \leq \mathcal{N}_\infty(\epsilon/\sqrt{12 H_\phi r}, \mathcal{G}_2, mn). \tag{C.4}$$

Appealing once again to a result by *Zhang* (2002, Corollary 3), we get

$$\log_2 \mathcal{N}_\infty(\epsilon/\sqrt{12 H_\phi r}, \mathcal{G}_2, mn) \leq \left\lceil \frac{12 H_\phi \, W_2^2 \, R_X^2 \, r}{\epsilon^2} \right\rceil \times \log_2(2mn+1)$$

which finishes the proof. □

Proof of Corollary 43

Proof. We plug in Proposition 5's estimate into (4.5):

$$\begin{aligned} \widehat{\mathfrak{R}}_n \left(\mathcal{F}_{\phi,2}(r) \right) &\leq \inf_{\alpha > 0} \left(4\alpha + 10 \int_{\alpha}^{\sqrt{Br}} \sqrt{\frac{\left\lceil \frac{12 H_\phi \, W_2^2 \, R_X^2 \, r}{\epsilon^2} \right\rceil \log_2(2mn+1)}{n}} d\epsilon \right) \\ &\leq \inf_{\alpha > 0} \left(4\alpha + 20\sqrt{3} W_2 R_X \sqrt{\frac{r H_\phi \log_2(3mn)}{n}} \int_{\alpha}^{\sqrt{Br}} \frac{1}{\epsilon} d\epsilon \right) . \end{aligned}$$

Now choosing $\alpha = C\sqrt{r}$ where $C = 5\sqrt{3} W_2 R_X \sqrt{\frac{H_\phi \log_2(3mn)}{n}}$ gives us the upper bound

$$\widehat{\mathfrak{R}}_n \left(\mathcal{F}_{\phi,2}(r) \right) \leq 4\sqrt{r} C \left(1 + \log \frac{\sqrt{B}}{C} \right) \leq 4\sqrt{r} C \log \frac{3\sqrt{B}}{C}.$$

□

Proof of Theorem 44

Proof. We appeal to Theorem 6.1 of *Bousquet* (2002) that assumes there exists an upper bound

$$\widehat{\mathfrak{R}}_n\left(\mathcal{F}_{2,\phi}(r)\right) \leq \psi_n(r)$$

where $\psi_n : [0,\infty) \to \mathbb{R}_+$ is a non-negative, non-decreasing, non-zero function such that $\psi_n(r)/\sqrt{r}$ is non-increasing. The upper bound in Corollary 43 above satisfies these conditions and therefore we set $\psi_n(r) = 4\sqrt{r}C\log\frac{3\sqrt{B}}{C}$ with C as defined in Corollary 43. From Bousquet's result, we know that, with probability at least $1-\delta$,

$$\begin{aligned}\forall w \in \mathcal{F}_2,\ L_\phi(w) \leq \hat{L}_\phi(w) + 45r_n^\star + \sqrt{8r_n^\star L_\phi(w)} \\ + \sqrt{4r_0 L_\phi(w)} + 20r_0\end{aligned}$$

where $r_0 = B(\log(1/\delta) + \log\log n)/n$ and $r_n^\star$ is the largest solution to the equation $r = \psi_n(r)$. In our case, $r_n^\star = \left(4C\log\frac{3\sqrt{B}}{C}\right)^2$. This proves the first inequality.

Now, using the above inequality with $w = \hat{w}$, the empirical risk minimizer and noting that $\hat{L}_\phi(\hat{w}) \leq \hat{L}_\phi(w^\star)$, we get

$$\begin{aligned}L_\phi(\hat{w}) \leq \hat{L}_\phi(w^\star) + 45r_n^\star + \sqrt{8r_n^\star L_\phi(\hat{w})} \\ + \sqrt{4r_0 L_\phi(\hat{w})} + 20r_0\end{aligned}$$

The second inequality now follows after some elementary calculations detailed below. □

Details of some calculations in the proof of Theorem 44 Using Bernstein's inequality, we have, with probability at least $1-\delta$,

$$\begin{aligned}\hat{L}_\phi(w^\star) &\leq L_\phi(w^\star) + \sqrt{\frac{4\mathrm{Var}[\phi(Xw^\star, y)]\log(1/\delta)}{n}} + \frac{4B\log(1/\delta)}{n} \\ &\leq L_\phi(w^\star) + \sqrt{\frac{4BL_\phi(w^\star)\log(1/\delta)}{n}} + \frac{4B\log(1/\delta)}{n} \\ &\leq L_\phi(w^\star) + \sqrt{4r_0 L_\phi(w^\star)} + 4r_0.\end{aligned}$$

Set $D_0 = 45r_n^\star + 20r_0$. Putting the two bounds together and using some simple upper bounds, we have, with probability at least $1 - 2\delta$,

$$L_\phi(\hat{w}) \leq \sqrt{D_0 \hat{L}_\phi(w^\star)} + D_0,$$
$$\hat{L}_\phi(w^\star) \leq \sqrt{D_0 L_\phi(w^\star)} + D_0.$$

which implies that

$$L_\phi(\hat{w}) \leq \sqrt{D_0}\sqrt{\sqrt{D_0 L_\phi(w^\star)} + D_0} + D_0.$$

Using $\sqrt{ab} \leq (a+b)/2$ to simplify the first term on the right gives us

$$L_\phi(\hat{w}) \leq \frac{D_0}{2} + \frac{\sqrt{D_0 L_\phi(w^\star)} + D_0}{2} + D_0 = \frac{\sqrt{D_0 L_\phi(w^\star)}}{2} + 2D_0 \,.$$

BIBLIOGRAPHY

BIBLIOGRAPHY

Agarwal, A., D. Hsu, S. Kale, J. Langford, L. Li, and R. Schapire (2014), Taming the monster: A fast and simple algorithm for contextual bandits, in *Proceedings of the International Conference on Machine Learning*, pp. 1638–1646.

Agarwal, D., E. Gabrilovich, R. Hall, V. Josifovski, and R. Khanna (2009), Translating relevance scores to probabilities for contextual advertising, in *Proceedings of the ACM conference on Information and knowledge management*, pp. 1899–1902, ACM.

Agrawal, R., S. Gollapudi, A. Halverson, and S. Ieong (2009), Diversifying search results, in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pp. 5–14, ACM.

Agrawal, S., and N. Goyal (2013), Thompson sampling for contextual bandits with linear payoffs, in *Proceedings of the International Conference on Machine Learning*, pp. 127–135.

Ailon, N. (2014), Improved bounds for online learning over the permutahedron and other ranking polytopes, in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 29–37.

Baeza-Yates, R., and B. Ribeiro-Neto (1999), *Modern information retrieval*, vol. 463, ACM Press.

Bartlett, P. L., and S. Mendelson (2003), Rademacher and Gaussian complexities: Risk bounds and structural results, *The Journal of Machine Learning Research*, *3*, 463–482.

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

Bartók, G., and C. Szepesvári (2012), Partial monitoring with side information, in *Algorithmic Learning Theory*, pp. 305–319.

Bartok, G., D. P. Foster, D. Pal, A. Rakhlin, and C. Szepesvari (2014), Partial monitoringclassification, regret bounds, and algorithms, *Mathematics of Operations Research*, *39*(4), 967–997.

Beygelzimer, A., J. Langford, and P. Ravikumar (2007), Multiclass classification with filter trees, *Preprint, June*, *2*.

Bhaskara, A., and A. Vijayaraghavan (2011), Approximating matrix p-norms, in *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pp. 497–511, SIAM.

Blum, A., and Y. Mansour (2007), Learning, regret minimization, and equilibria, in *Algorithmic game theory*, edited by N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, Cambridge University Press.

Bousquet, O. (2002), Concentration inequalities and empirical processes theory applied to the analysis of learning algorithms, Ph.D. thesis, Ecole Polytechnique.

Bubeck, S., and R. Eldan (2015), Multi-scale exploration of convex functions and bandit convex optimization, *arXiv preprint arXiv:1507.06580*.

Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender (2005), Learning to rank using gradient descent, in *Proceedings of International Conference on Machine Learning*, pp. 89–96.

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

Cao, P., D. Zhao, and O. Zaiane (2013), An optimized cost-sensitive svm for imbalanced data learning, in *Advances in Knowledge Discovery and Data Mining*, pp. 280–292, Springer.

Cao, Z., T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li (2007a), Learning to rank: from pairwise approach to listwise approach, in *Proceedings of the International Conference on Machine Learning*, pp. 129–136.

Cao, Z., T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li (2007b), Learning to rank: from pairwise approach to listwise approach, in *Proceedings of the International Conference on Machine learning*, pp. 129–136, ACM.

Cavallanti, G., N. Cesa-Bianchi, and C. Gentile (2007), Tracking the best hyperplane with a simple budget perceptron, *Machine Learning*, *69*(2-3), 143–167.

Cesa-Bianchi, N. (2006), *Prediction, learning, and games*, Cambridge University Press.

Chakrabarti, S., R. Khanna, U. Sawant, and C. Bhattacharyya (2008), Structured learning for non-smooth ranking losses, in *Proceedings of Conference on Knowledge Discovery and Data Mining*, pp. 88–96.

Chapelle, O., and Y. Chang (2011a), Yahoo! learning to rank challenge overview., in *Yahoo! Learning to Rank Challenge*, pp. 1–24.

Chapelle, O., and Y. Chang (2011b), Yahoo! learning to rank challenge overview., *Journal of Machine Learning Research-Proceedings Track*, pp. 1–24.

Chapelle, O., and L. Li (2011), An empirical evaluation of thompson sampling, in *Advances in Neural Information Processing Systems*, pp. 2249–2257.

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

Chapelle, O., Q. Le, and A. Smola (2007), Large margin optimization of ranking measures, in *Neural Information Processing Systems Workshop: Machine Learning for Web Search*.

Chapelle, O., D. Metlzer, Y. Zhang, and P. Grinspan (2009), Expected reciprocal rank for graded relevance, in *Proceedings of the ACM Conference on Information and Knowledge Management*, pp. 621–630, ACM.

Chapelle, O., Y. Chang, and T.-Y. Liu (2011), Future directions in learning to rank, in *Proceedings of the Yahoo! Learning to Rank Challenge June 25, 2010, Haifa, Israel*, Journal of Machine Learning Research Workshop and Conference Proceedings, pp. 91–100.

Chawla, N., N. Japkowicz, and A. Kotcz (2004), Editorial: special issue on learning from imbalanced data sets, *ACM SIGKDD Explorations Newsletter*, *6*(1), 1–6.

Cheng, T., Y. Wang, and S. Bryant (2012), FSelector: a ruby gem for feature selection, *Bioinformatics*, *28*(21), 2851–2852.

Chu, W., L. Li, L. Reyzin, and R. Schapire (2011), Contextual bandits with linear payoff functions, in *International Conference on Artificial Intelligence and Statistics*, pp. 208–214.

Cortes, C., and M. Mohri (2004a), AUC optimization vs. error rate minimization, *Advances in Neural Information Processing Systems*, *16*(16), 313–320.

Cortes, C., and M. Mohri (2004b), AUC optimization vs. error rate minimization, *Advances in Neural Information Processing Systems*, *16*(16), 313–320.

Cossock, D., and T. Zhang (2006a), Subset ranking using regression, in *Proceedings of Conference on Learning Theory*, pp. 605–619.

Cossock, D., and T. Zhang (2006b), Subset ranking using regression, in *Proceedings of Conference on Learning Theory*, pp. 605–619.

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

Crammer, K., and Y. Singer (2001), Pranking with ranking., in *Advances in Neural Information Processing Systems*, pp. 641–647.

Crammer, K., and Y. Singer (2002), A new family of online algorithms for category ranking, in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 151–158, ACM.

Dekel, O., S. Shalev-Shwartz, and Y. Singer (2008), The forgetron: A kernel-based perceptron on a budget, *SIAM Journal on Computing*, *37*(5), 1342–1372.

Dong, A., Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz (2010), Towards recency ranking in web search, in *Proceedings of the third ACM international conference on Web search and data mining*, pp. 11–20, ACM.

Duchi, J. C., L. W. Mackey, and M. I. Jordan (2010), On the consistency of ranking algorithms, in *Proceedings of the International Conference on Machine Learning*, pp. 327–334.

Dudik, M., D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang (2011), Efficient optimal learning for contextual bandits, *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, 2011*.

Elsas, J. L., V. R. Carvalho, and J. G. Carbonell (2008), Fast learning of document ranking functions with the committee perceptron, in *Proceedings of the International Conference on Web Search and Data Mining*, pp. 55–64, ACM.

Flaxman, A. D., A. T. Kalai, and H. B. McMahan (2005), Online convex optimization in the bandit setting., in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 385–394.

Foster, D. P., and A. Rakhlin (2012), No internal regret via neighborhood watch, in *International Conference on Artificial Intelligence and Statistics*, pp. 382–390.

Freund, Y., and R. E. Schapire (1999), Large margin classification using the perceptron algorithm, *Machine learning*, pp. 277–296.

Freund, Y., R. Iyer, R. E. Schapire, and Y. Singer (2003), An efficient boosting algorithm for combining preferences, *J. Mach. Learn. Res.*, *4*, 933–969.

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

Harrington, E. F. (2003), Online ranking/collaborative filtering using the perceptron algorithm, in *Proceedings of International Conference on Machine Learning*, vol. 20, pp. 250–257.

He, X., et al. (2014), Practical lessons from predicting clicks on ads at facebook, in *Proceedings of 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1–9, ACM.

Henneges, C., G. Hinselmann, S. Jung, J. Madlung, W. Schütz, A. Nordheim, and A. Zell (2011), Ranking methods for the prediction of frequent top scoring peptides from proteomics data, *Journal of Proteomics & Bioinformatics*, *2009*.

Herbrich, R., T. Graepel, and K. Obermayer (1999), Large margin rank boundaries for ordinal regression, *Advances in Neural Information Processing Systems*, pp. 115–132.

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

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

Jain, P., N. Natarajan, and A. Tewari (2015), A family of online algorithms for general prediction problems, in *Advances in Neural Information Processing Systems*.

Japkowicz, N., and S. Stephen (2002), The class imbalance problem: A systematic study, *Intelligent data analysis*, *6*(5), 429–449.

Järvelin, K., and J. Kekäläinen (2000), 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*, pp. 41–48, ACM.

Järvelin, K., and J. Kekäläinen (2002), Cumulated gain-based evaluation of IR techniques, *ACM Transactions on Information Systems*, *20*(4), 422–446.

Joachims, T. (2002), Optimizing search engines using clickthrough data, in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 133–142, ACM.

Kalai, A., and S. Vempala (2005), Efficient algorithms for online decision problems, *Journal of Computer and System Sciences*, pp. 291–307.

Karatzoglou, A., L. Baltrunas, and Y. Shi (2013), Learning to rank for recommender systems, in *Proceedings of the 7th ACM Conference on Recommender systems*, pp. 493–494, ACM.

Kleinberg, R., and T. Leighton (2003), The value of knowing a demand curve: Bounds on regret for online posted-price auctions, in *Foundations of Computer Science, 2003*, pp. 594–605.

Koh, E., and N. Gupta (2014), An empirical evaluation of ensemble decision trees to improve personalization on advertisement, in *Proceedings of KDD 14 Second Workshop on User Engagement Optimization*.

Lan, Y., T.-Y. Liu, T. Qin, Z. Ma, and H. Li (2008), Query-level stability and generalization in learning to rank, in *Proceedings of the International Conference on Machine Learning*, pp. 512–519, ACM.

Lan, Y., T.-Y. Liu, Z. Ma, and H. Li (2009), Generalization analysis of listwise learning-to-rank algorithms, in *Proceedings of the International Conference on Machine Learning*, pp. 577–584.

Langford, J., and T. Zhang (2008), The epoch-greedy algorithm for multi-armed bandits with side information, in *Advances in Neural Information Processing Systems*, pp. 817–824.

Langford, J., L. Li, and M. Dudik (2011), Doubly robust policy evaluation and learning, in *Proceedings of the 28th International Conference on Machine Learning*, pp. 1097–1104.

Le, Q., T. Sarlós, and A. Smola (2013), Fastfoodapproximating kernel expansions in log-linear time, in *Proceedings of International Conference on Machine Learning*.

Li, L., W. Chu, J. Langford, and X. Wang (2011), Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms, in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 297–306, ACM.

Lin, T., B. Abrahao, R. Kleinberg, and J. Lui (2014), Combinatorial partial monitoring game with linear feedback and its applications, in *Proceedings of the International Conference on Machine Learning*, pp. 901–909, ACM.

Liu, T.-Y. (2011), *Learning to rank for information retrieval*, Springer Science & Business Media.

Liu, T.-Y., J. Xu, T. Qin, W. Xiong, and H. Li (2007a), 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*, pp. 3–10.

Liu, T.-y., J. Xu, T. Qin, W. Xiong, and H. Li (2007b), 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*, pp. 3–10.

McMahan, H., et al. (2013), Ad click prediction: a view from the trenches, in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1222–1230, ACM.

Mendelson, S. (2002), Rademacher averages and phase transitions in Glivenko-Cantelli classes, *IEEE Transactions on Information Theory*, *48*(1), 251–263.

Mohri, M., A. Rostamizadeh, and A. Talwalkar (2012), *Foundations of Machine Learning*, MIT Press.

Ni, W., and Y. Huang (2008), Online ranking algorithm based on perceptron with margins, in *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pp. 814–819, IEEE.

Panchenko, D. (2002), Some extensions of an inequality of Vapnik and Chervonenkis., *Electronic Communications in Probability*, *7*, 55–65.

Piccolboni, A., and C. Schindelhauer (2001a), Discrete prediction games with arbitrary feedback and loss, in *Computational Learning Theory*, pp. 208–223, Springer.

Piccolboni, A., and C. Schindelhauer (2001b), Discrete prediction games with arbitrary feedback and loss, in *In Proceedings of Conference on Learning Theory*, pp. 208–223, Springer.

Qin, T., and T.-Y. Liu (2006), Microsoft letor website, `http://research.microsoft.com/en-us/um/beijing/projects/letor/`.

Radlinski, F., R. Kleinberg, and T. Joachims (2008), Learning diverse rankings with multi-armed bandits, in *Proceedings of the International conference on Machine learning*, pp. 784–791, ACM.

Radlinski, F., P. N. Bennett, B. Carterette, and T. Joachims (2009), Redundancy, diversity and interdependent document relevance, in *ACM SIGIR*, pp. 46–52, ACM.

Rahimi, A., and B. Recht (2007), Random features for large-scale kernel machines, in *Advances in Neural Information Processing Systems*, pp. 1177–1184.

Ravikumar, P., A. Tewari, and E. Yang (2011), On NDCG consistency of listwise ranking methods, in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 618–626.

Richardson, M., E. Dominowska, and R. Ragno (2007), Predicting clicks: estimating the click-through rate for new ads, in *Proceedings of the 16th international conference on World Wide Web*, pp. 521–530, ACM.

Rifkin, R., and A. Klautau (2004), In defense of one-vs-all classification, *The Journal of Machine Learning Research*, *5*, 101–141.

Rosenblatt, F. (1958), The perceptron: a probabilistic model for information storage and organization in the brain., *Psychological review*, *65*(6), 386.

Sanderson, M. (2010), *Test collection based evaluation of information retrieval systems*, vol. 13, Now Publishers Inc.

Schapire, R. E., and Y. Freund (2012), *Boosting: Foundations and Algorithms*, MIT Press (MA).

Shalev-Shwartz, S. (2011), Online learning and online convex optimization, *Foundations and Trends in Machine Learning*, pp. 107–194.

Shalev-Shwartz, S., O. Shamir, N. Srebro, and K. Sridharan (2009), Stochastic convex optimization, in *Proceedings of the 22nd Annual Conference on Learning Theory*.

Shamir, O., and T. Zhang (2013), Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes, in *Proceedings of the International Conference on Machine Learning*, pp. 71–79.

Shen, L., and A. K. Joshi (2005), Ranking and reranking with perceptron, *Machine Learning*, *60*(1-3), 73–96.

Srebro, N., K. Sridharan, and A. Tewari (2010), Smoothness, low noise, and fast rates, in *Advances in Neural Information Processing Systems 23*, pp. 2199–2207.

Tax, N., S. Bockting, and D. Hiemstra (2015), A cross-benchmark comparison of 87 learning to rank methods, *Information Processing and Management*, pp. 757–772.

Theocharous, G., P. Thomas, and M. Ghavamzadeh (2015), Ad recommendation systems for life-time value optimization, in *Proceedings of the 24th International Conference on World Wide Web Companion*, pp. 1305–1310.

Vapnik, V. N. (1999), *The Nature of Statistical Learning Theory*, second ed., Springer.

Wang, J., J. Wan, Y. Zhang, and S. C. Hoi (2015), Solar: Scalable online learning algorithms for ranking, in *Proceedings of Association for Computational Linguistics*.

Wen, Z., B. Kveton, and A. Ashkan (2014), Efficient learning in large-scale combinatorial semi-bandits, in *Proceedings of the International Conference on Machine Learning*.

Xu, J., and H. Li (2007), Adarank: a boosting algorithm for information retrieval, in *Proceedings of SIGIR*, pp. 391–398.

Yue, Y., T. Finley, F. Radlinski, and T. Joachims (2007), A support vector method for optimizing average precision, in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 271–278.

Zhang, T. (2002), Covering number bounds of certain regularized linear function classes, *The Journal of Machine Learning Research*, *2*, 527–550.

Zhao, P., R. Jin, T. Yang, and S. C. Hoi (2011), Online auc maximization, in *Proceedings of the International Conference on Machine Learning*, pp. 233–240.

Zhao, P., J. Wang, P. Wu, R. Jin, and S. C. Hoi (2012), Fast bounded online gradient descent algorithms for scalable kernel-based online learning, *arXiv preprint arXiv:1206.4633*.

Zinkevich, M. (2003), Online convex programming and generalized infinitesimal gradient ascent, in *Proceedings of the International Conference on Machine Learning*, pp. 928–936.