Online Learning to Rank with Feedback at the Top

Sougata Chaudhuri
University of Michigan, Ann Arbor

Ambuj Tewari
University of Michigan, Ann Arbor

Abstract

We consider an online learning to rank setting in which, at each round, an oblivious adversary generates a list of m documents, pertaining to a query, and the learner ranks the documents according to assigned scores. The adversary then generates a relevance vector and the learner updates its ranker according to the feedback received. We consider the setting where the feedback is restricted to be the relevance levels of only the top k documents in the ranked list, for $k \ll m$. However, the performance of learner is judged based on the unrevealed full relevance vectors, using an appropriate learning to rank loss function. We develop efficient algorithms for well known losses in the pointwise, pairwise and listwise families. We also prove that no online algorithm can have sublinear regret, with top 1 feedback, for any loss that is calibrated with respect to NDCG. We apply our algorithms on benchmark datasets demonstrating efficient online learning of a ranking function from highly restricted feedback.

1 Introduction

In learning to rank for information retrieval, the objective is to rank lists of documents, pertaining to different queries, so that the documents that are more relevant to a query are ranked above those that are less relevant. Most learning to rank methods are based on supervised *batch* learning, i.e., rankers are trained on batch data consisting of instances and labels [Liu, 2011]. The instances are lists of documents, pertaining to different queries, and labels are in the form of relevance vectors. The accuracy of a ranked list, in comparison to the actual relevance of the documents, is measured by various ranking measures, such as NDCG, AP and ERR.

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 training data might not be possible at all [Sanderson, 2010]. Moreover, a ranker trained from batch data might not be able to satisfy changing user needs and preferences. Recent research has focused on online learning of ranking systems, where a ranker is updated on the fly. One direction of work deploys models which 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 displayed items might not be clickable, such as in certain mobile apps. 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]. Another direction of work deploys models which learn optimal ranking of a fixed list of items, for diverse user preferences [Radlinski et al., 2008, Chaudhuri and Tewari, 2015]. Specifically, the latter work assumes that a user generates a full relevance vector for the entire ranked list of items but gives feedback only on the top ranked item. Motivation for this feedback model comes from considerations of user burden constraints (users will feel burdensome to provide careful feedback on all items) and privacy concerns (users will be unwilling to provide feedback on all items if they are about sensitive issues such as medical conditions). However, the requirement of having a fixed set of items to rank severely limits the practical applicability of this line of work.

Our work extends the work of Chaudhuri and Tewari [2015], by combining query-level ranking, in an online manner, with explicit but restricted feedback. We formalize the problem as an online game played over T rounds, between a learner and an *oblivious* adversary. At each round, the adversary generates a document list of length m, pertaining to a query. The learner

Appearing in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51. Copyright 2016 by the authors.

sees the list and produces a real valued score vector to rank the documents. We assume that the ranking is generated by sorting the score vector in descending order of its entries. The adversary then generates a relevance vector but the learner gets to see the relevance of only the top-k items of the ranked list, where $k \ll m$ is a small constant, like 1 or 2. The learner's loss in each round, based on the learner's score vector and the *full* relevance vector, is measured by some continuous learning to rank loss function. We focus on continuous surrogates losses, e.g., the cross entropy surrogate in ListNet [Cao et al., 2007] and hinge surrogate in RankSVM [Joachims, 2002], instead of discontinuous ranking measures like NDCG, AP, or ERR because the latter lead to intractable optimization problems. We note that the top-k feedback model is distinct from the full and bandit feedback models since neither the full relevance vector nor the loss at end of each round is revealed to the learner. Technically, the problem is an instance of *partial monitoring* [Cesa-Bianchi et al., 2006, Bartok et al., 2014], *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.

We make two main contributions in this paper. First, we propose a general, efficient algorithm for online learning to rank with top-k feedback and show that it works in conjunction with a number of ranking surrogates. We characterize the minimum feedback required, i.e., the value of k, for the algorithm to work with a particular surrogate by formally relating the feedback mechanism with the structure of the surrogates. We then apply our general techniques to three convex ranking surrogates and one non-convex surrogate. The convex surrogates considered are from three major learning to ranking methods: squared loss from a *pointwise* method [Cossock and Zhang, 2008], hinge loss used in the *pairwise* RankSVM [Joachims, 2002] method, and (modified) cross-entropy surrogate used in the *listwise* ListNet [Cao et al., 2007] method. The non-convex surrogate considered is the SmoothDCG surrogate [Chapelle and Wu, 2010]. For the three convex surrogates, we establish an $O(T^{2/3})$ regret bound.

The convex surrogates we mentioned above are widely used but are known to fail to be calibrated with respect to NDCG [Ravikumar et al., 2011]. Our second contribution is to show that for the entire class of NDCG calibrated surrogates, no online algorithm can have sublinear (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 rather surprising result is non-trivial and relies on exploiting a connection between the construction of optimal adversary strategies for hopeless *finite action* partial monitoring games [Piccolboni and Schindelhauer, 2001] and the structure of NDCG calibrated surrogates. We only focus on NDCG calibrated surrogates for the *impossibility* results since no (convex) surrogate can be calibrated for AP and ERR [Calauzenes et al., 2012]. This impossibility result is the first of its kind for a natural partial monitoring problem with side information when the learner's action space is infinite. Note, however, that there does exist work on partial monitoring problems with continuous learner actions, but without side information [Kleinberg and Leighton, 2003, Cesa-Bianchi et al., 2006], and vice versa [Bartók and Szepesvári, 2012, Gentile and Orabona, 2014].

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

2 Preliminaries

In learning to rank, an instance is a matrix $X \in \mathbb{R}^{m \times d}$, consisting of a list of m documents, each represented as a feature vector in $\mathbb{R}^d$, with each list pertaining to a single query. The supervision is in form of a relevance vector $R = \{0, 1, \dots, n\}^m$, 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. $X_{i:}$ denotes ith row of X and R_i denotes ith component of R. *The subscript t is exclusively used to denote time t.* Thus, R_t denotes relevance vector generated at time t and $R_{t,i}$ denotes ith component of R_t. We assume feature vectors representing documents are bounded by R_D in ℓ_2 norm.

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_i^w 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. 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 in π while $\pi^{-1}(i) = j$ means document i is placed at position j. S_m denote the set of $m!$ different permutations of $[m]$ where $[m] = \{1, 2 \dots, m\}$.

Various ranking measures, like NDCG and AP, judge the quality of a ranking function, by comparing the ranked lists produced by the ranking function and the relevance vector, respectively. Formally, NDCG, cut off at $\ell \leq m$ for a query with m documents, with relevance vector R and score vector s induced by a ranking function, is defined as follows: $\text{NDCG}_\ell(s,R) = \frac{1}{Z_\ell(R)} \sum_{i=1}^{\ell} G(R_{\pi_s(i)})D(i)$. Here, $G(r) = 2^r - 1$, $D(i) = \frac{1}{\log_2{(i+1)}}$, $Z_\ell(R) = \max\limits_{\pi \in S_m} \sum_{i=1}^{\ell} G(R_{\pi(i)})D(i)$. $\pi_s = \text{argsort}(s)$ is the permutation induced by score vector s in descending order. Since optimization of the discontinuous ranking measures is an NP-hard problem, most ranking methods are based on minimizing *surrogate* losses, which can be optimized more 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 target ranking measures.

3 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 according to it. The adversary then generates a relevance vector R_t but only reveals the relevance 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, $\phi(\tilde{s}_t, R_t)$. *Note that since the learner's objective is to produce good ranking at every round, learner's performance is measured w.r.t. to entire relevance vector R_t whereas it only gets restricted feedback on 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: Alg. 1 is our general algorithm for learning a ranking function, online, from partial feedback. The key step in Alg. 1 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 $\mathbb{E}_t\left[\cdot\right]$ 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: $\mathbb{E}_t\left[\tilde{z}_t\right] = \nabla_{w=w_t}\phi(X_t w, R_t)$. We note that conditioned on the past, the score vector $s_t^{w_t} = X_t w_t$ is deterministic. We start with a general result relating feedback to the construction of unbiased estimator of a vector valued function. Let $\mathbb{P}$ 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,

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

Lemma 1. *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:*

$$F(x) = \sum_{(i_1,i_2,\ldots,i_\ell) \in \ {}^mP_\ell} h_{i_1,i_2,\ldots,i_\ell}(x_{i_1}, x_{i_2}, \ldots, x_{i_\ell}) \tag{2}$$

where $\ell \leq k$, ${}^mP_\ell$ is ℓ permutations of m and $h : \mathbb{R}^\ell \mapsto \mathbb{R}^a$ (the subscripts in h is used to denote possibly different functions). Moreover, when $F(x)$ can be written in form of Eq 2 , with $\ell = k$, an unbiased estimator of $F(x)$, based on $\{\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}\}$, is,

$$g(\sigma, x_{\sigma(1)}, \ldots, x_{\sigma(k)}) = \frac{\sum\limits_{(j_1,j_2,\ldots,j_k) \in S_k} h_{\sigma(j_1),\ldots,\sigma(j_k)}(x_{\sigma(j_1)}, \ldots, x_{\sigma(j_k)})}{\sum\limits_{(j_1,\ldots,j_k) \in S_k} p(\sigma(j_1), \ldots, \sigma(j_k))} \tag{3}$$

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

Algorithm 1 Ranking with Top-k Feedback (RTop-kF)

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

Illustrative Examples: We provide simple examples to concretely illustrate the abstract functions in Lemma 1. 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 $F(x)$: $g(\sigma, x_{\sigma(1)}) = \dfrac{x_{\sigma(1)} e_{\sigma(1)}}{p(\sigma(1))}$, where $p(\sigma(1)) = \sum\limits_{\pi \in S_m} \mathbb{P}(\pi)\mathbb{1}(\pi(1) = \sigma(1))$. 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: $g(\sigma, x_{\sigma(1)}) = \dfrac{h_{\sigma(1)}(x_{\sigma(1)})}{p(\sigma(1))}$.

4 Unbiased Estimators of Gradients of Surrogates

Alg. 1 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. 1 with $\mathbb{P} = \mathbb{P}_t$ where $\mathbb{P}_t$ is the distribution in round t in Alg. 1.

4.1 Convex Surrogates

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

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

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

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

We note that the unbiased estimator was constructed from top-2 feedback. The following lemma, in conjunction with the necessary condition of Lemma 1 shows that it is the minimum information required to construct the unbiased estimator.

Lemma 2. *The gradient of RankSVM surrogate, i.e., $\phi_{svm}(s,R)$ cannot be decomposed over 1 coordinate of R at a time.*

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

The authors of the ListNet method use a cross-entropy surrogate on two probability distributions on permutations, induced by score and relevance vector respectively. More formally, the surrogate is defined as follows[1]. 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, $\phi_{\mathrm{LN}}(s,R) = -\sum_{i=1}^m P_i(R)\log P_i(s)$ and $\nabla_s \phi_{\mathrm{LN}}(s,R) = \sum_{i=1}^m \left(-\frac{\exp(R_i)}{\sum_{j=1}^m \exp(R_j)} + \frac{\exp(s_i)}{\sum_{j=1}^m \exp(s_j)}\right) e_i$. We have the following lemma about the gradient of ϕ_{LN}.

Lemma 3. *The gradient of ListNet surrogate $\phi_{LN}(s,R)$ cannot be decomposed over k, for $k = 1,2$, coordinates of R at a time.*

In fact, an examination of the proof of the above lemma reveals that decomposability at any $k < m$ does not hold for the gradient of LisNet surrogate, though we only prove it for $k = 1,2$ (since feedback for top k items with $k > 2$ does not seem practical). Due to Lemma 1, this means that if we want to run Alg. 1 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 $\phi_{\mathrm{LN}}(s,R)$ 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.

[1]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.

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

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

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

Now, from Lemma 1, 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 Alg. 1 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.

4.1.1 Non-convex Surrogate

We provide an example of a non-convex surrogate for which Alg. 1 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:

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

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

Using Lemma 1, 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:

$$\tilde{\mathbf{z}} = X^\top \left(\frac{G(R_{\sigma(1)})}{p(\sigma(1))} (*) \right)$$

$$(*) = \sum_{j=1}^{m} \frac{1}{\epsilon} [\frac{\exp(s_{\sigma(1)}/\epsilon)}{\sum_{j'} \exp(s_{j'}/\epsilon)} \mathbb{1}_{(\sigma(1)=j)} - \frac{\exp((s_{\sigma(1)} + s_j)/\epsilon)}{(\sum_{j'} \exp(s_{j'}/\epsilon))^2}] e_j$$

4.2 Computational Complexity of Algorithm 1

Three of the four key steps governing the complexity of Alg. 1, 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.

4.3 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:

$$\mathbb{E}\left[\sum_{t=1}^{T} \phi(X_t w_t, R_t)\right] \leq \min_{w: \|w\|_2 \leq U} \sum_{t=1}^{T} \phi(X_t w, R_t) + \frac{U^2}{2\eta} + \frac{\eta}{2} \mathbb{E}\left[\sum_{t=1}^{T} \|\tilde{z}_t\|_2^2\right] \tag{4}$$

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 Alg. 1, 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 [2]

$$\mathbb{E}\left[\sum_{t=1}^{T} \phi(\tilde{s}_t, R_t)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T} \phi(X_t w_t, R_t)\right] + O(\gamma T) \tag{5}$$

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 4.1 , 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 4. *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 5. *For parameter γ in Alg. 1 , R_D being the bound on ℓ_2 norm of the feature vectors (rows of document matrix X), m being the upper bound on number of documents per query, U being the radius of the Euclidean ball denoting the space of ranking parameters and $R_{\max}$ being the maximum possible relevance value (in practice always ≤ 5), let $C^\phi \in \{C^{sq}, C^{svm}, C^{KL}\}$ be polynomial functions of R_D, m, U, R_{max}, where the degrees of the polynomials depend on the surrogate ($\phi_{sq}, \phi_{svm}, \phi_{KL}$), with no degree ever greater than four. Then we have,*

$$\mathbb{E}_t \left[\|\tilde{z}_t\|_2^2\right] \leq \frac{C^\phi}{\gamma} \tag{6}$$

Plugging Eq. 6 and Eq. 5 in Eq. 4, 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 4.1. *For any sequence of instances and labels $(X_t, R_t)_{\{t \in [T]\}}$, applying Alg. 1 with top-1 feedback*

[2]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.

for ϕ_{sq} and ϕ_{KL} and top-2 feedback for ϕ_{svm}, will produce the following bound:

$$\mathbb{E}\left[\sum_{t=1}^{T}\phi(\tilde{s}_t, R_t)\right] - \min_{w:\|w\|_2\leq U}\sum_{t=1}^{T}\phi(X_t w, R_t) \leq C^{\phi} O\left(T^{2/3}\right) \tag{7}$$

where C^{ϕ} is a surrogate dependent function, as described in Lemma 5 , 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 $O(T^{3/4})$. Surprisingly, Alg. 1, 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.

5 Impossibility of Sublinear Regret for NDCG Calibrated Surrogates

Learning to rank methods optimize surrogates to learn a ranking function, even though performance is measured by target measures like NDCG. This is done because direct optimization of the measures lead to NP-hard optimization problems. One of the most desirable properties of any surrogate is *calibration*, i.e., the surrogate should be calibrated w.r.t the target [Bartlett et al., 2006]. Intuitively, it means that a function with small expected surrogate loss on unseen data should have small 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 $G(\mathbf{R}) = (G(R_1),\ldots,G(R_m))^{\top}$.

Theorem 5.1. *[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*

$$s^*_{\phi}(\eta) = g\left(\mathbb{E}_{R\sim\eta}\left[\frac{G(\mathbf{R})}{Z_m(R)}\right]\right). \tag{8}$$

Informally, Eq. 8 states that $\operatorname{argsort}(s^*_{\phi}(\eta)) \subseteq \operatorname{argsort}(\mathbb{E}_{R\sim\eta}\left[\frac{G(\mathbf{R})}{Z_m(R)}\right])$ 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 $\|s - \frac{G(\mathbf{R})}{Z_m(R)}\|_2^2$.

We now state the *impossibility* result for the class of NDCG calibrated surrogates with top-1 feedback.

Theorem 5.2. *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 s.t. 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 [2001]. An examination of their proof of Thm. 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 (Thm. 5.1) to explicitly construct two such probability distributions. This proves the result. □

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

6 Empirical Results

Objective: We conducted experiments on benchmark datasets to demonstrate the performance of ranking functions that are learnt from partial feedback. As stated before, though Alg. 1 is designed to minimize surrogate based regret, the users only care about the ranking presented to them, and indeed the algorithm interacts with users only through ranked lists. We tested the quality of the ranked lists, and hence the performance of the evolving ranking functions, against

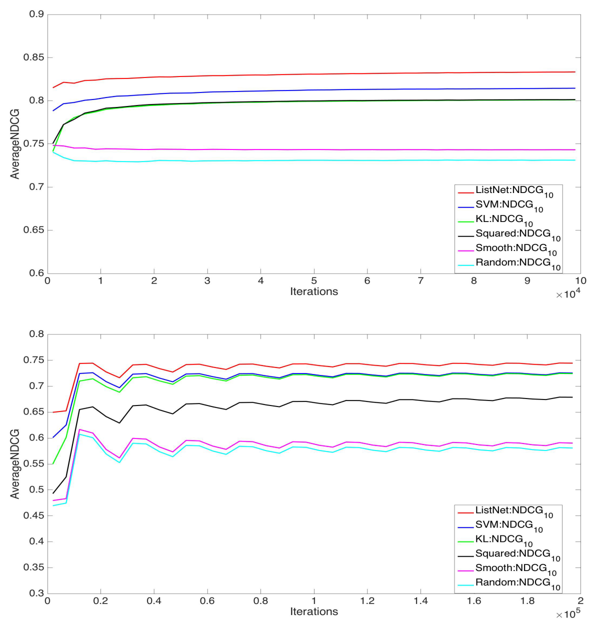

Figure 1: Average NDCG$_{10}$ values for different algorithms, for **Yandex (top)** and **Yahoo (bottom)**. ListNet:NDCG$_{10}$ (in cyan) is a **full feedback** algorithm and Random:NDCG$_{10}$ (in red) is **no feedback** algorithm.

the full relevance vectors via NDCG$_{10}$.

Ranking functions compared: We applied Alg. 1, with top-1 feedback, on Squared, KL and SmoothDCG surrogates, and with top-2 feedback, on the RankSVM surrogate. *Since our work is based on a novel feedback model, the performance of Alg. 1 could not be directly compared with any published baseline.* So, based on the objective of our work, we selected two different ranking methods for comparison. The first one is the online version of ListNet ranking algorithm, with full relevance vector revealed at end of every round. ListNet is not only one of the most cited ranking algorithms (over 700 citations according to Google Scholar), but also one of the most validated algorithms [Tax et al., 2015]. We emphasize that some of the ranking algorithms, which have shown better empirical performance than ListNet, are usually based on non-convex surrogates with complex ranking functions. These algorithms cannot usually be converted into online algorithms which learn from streaming data. The second one is a fully random algorithm which outputs fully random ranking of documents at each round. *We are comparing Alg. 1, which learns from highly restricted feedback, with an algorithm which learns from full feedback and an algorithm which receives no feedback.*

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, 2011] 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.

Setting of experiments: We selected time horizon $T = 200000$ (Yahoo) and $T = 100000$ (Yandex) iterations for our experiments (thus, each algorithm went over each dataset multiple times). 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 Alg. 1, we selected $\eta = \frac{0.01}{t^{2/3}}$ and $\gamma = \frac{0.1}{t^{1/3}}$, for round t (we saw considerable improvement in performance when η was selected much smaller than γ). 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.

Observations: In both the datasets, ListNet, with full information, has highest average NDCG value throughout. However, Alg. 1, with the convex surrogates, produce competitive performance. In fact, in the Yahoo dataset, our algorithms, with RankSVM and KL, are very close to the performance of ListNet. RanSVM does better than the other surrogates, since the estimator of RankSVM gradient is constructed from top-2 feedback, leading to lower variance. KL, being listwise in nature, does better than squared loss. *Crucially, our algorithms, based on all three convex surrogates, perform significantly better than the purely random algorithm, and are much closer to full feedback 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.

Acknowledgement

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

References

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

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

Gabor Bartok et al. Partial monitoring–classification, regret bounds, and algorithms. *Mathematics of Operations Research*, 39(4):967–997, 2014.

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

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

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

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

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

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

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

Sougata Chaudhuri and Ambuj Tewari. Online ranking with top-1 feedback. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 129–137. ACM, 2015.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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