An Alternative Cross Entropy Loss for Learning-to-Rank

Sebastian Bruch*
National Institutes of Health
Bethesda, MD, United States
sebastian.bruch@nih.gov

ABSTRACT

Listwise learning-to-rank methods form a powerful class of ranking algorithms that are widely adopted in applications such as information retrieval. These algorithms learn to rank a set of items by optimizing a loss that is a function of the entire set—as a surrogate to a typically non-differentiable ranking metric. Despite their empirical success, existing listwise methods are based on heuristics and remain theoretically ill-understood. In particular, none of the empirically successful loss functions are related to ranking metrics. In this work, we propose a cross entropy-based learning-to-rank loss function that is theoretically sound, is a convex bound on NDCG—a popular ranking metric—and is consistent with NDCG under learning scenarios common in information retrieval. Furthermore, empirical evaluation of an implementation of the proposed method with gradient boosting machines on benchmark learning-to-rank datasets demonstrates the superiority of our proposed formulation over existing algorithms in quality and robustness.

CCS CONCEPTS

• **Information systems → Learning to rank**.

KEYWORDS

Learning to Rank; Ranking Metric Optimization; Information Retrieval

1 INTRODUCTION

Learning-to-rank is a central problem in a range of applications including web search, recommendation systems, and question answering. The task is to learn a function that, conditioned on some context, arranges a set of items into an ordered list so as to maximize a given metric. In this work, without loss of generality, we take search as an example where a set of documents (items) are ranked by their relevance to a query (context).

Rather than directly working with permutations, learning-to-rank methods typically approach the ranking problem as one of

---

"score and sort." The objective is then to learn a "scoring" function to estimate query-document relevance. Subsequently, they sort documents in decreasing relevance to form a ranked list. Ideally, the resulting ranked list should maximize a ranking metric.

Popular ranking metrics are instances of the general class of *conditional linear rank statistics* [10] that summarize the Receiver Operator Characteristic (ROC) curve. Of particular interest are the ranking statistics that care mostly about the leftmost portion of the ROC curve, corresponding to the top of the ranked list. Mean Reciprocal Rank and Normalized Discounted Cumulative Gain [14] are two such metrics that are widely used in information retrieval.

Ranking metrics, as functions of learning-to-rank scores, are flat almost everywhere; a small perturbation of scores is unlikely to lead to a change in the metric. This property poses a challenge for gradient-based optimization algorithms, making a direct optimization of ranking metrics over a complex hypothesis space infeasible. Addressing this challenge has been the focus of a large body of research [20], with most considering smooth loss functions as surrogates to metrics.

The majority of existing loss functions [4–6, 15, 33], however, are only loosely related to ranking metrics such as NDCG. ListNet [6], as an example, projects labels and scores onto the probability simplex and minimizes the cross-entropy between the resulting distributions. LambdaMART [5, 32] (denoted as λMART) forgoes the loss function altogether and heuristically formulates the gradients.

The heuristic nature of learning-to-rank loss functions and a lack of theoretical justification for their use have hindered progress in the field. While λMART remains the state-of-the-art method to date, the fact that its loss function, presumed to be smooth, is unknown makes a theoretical analysis of the algorithm difficult. Empirical improvements over existing methods remain marginal for similar reasons.

In this work, we are motivated to help close the gap above. To that end, we present a construction of the cross-entropy loss which we dub xe_{NDCG}, that is only slightly different from the ListNet loss, but that enjoys strong theoretical properties. In particular, we prove that our construction is a convex bound on NDCG, thereby lending credence to its optimization for the purpose of learning ranking functions. Furthermore, we show that the generalization error of xe_{NDCG} compares favorably with that of λMART's. Experiments on benchmark learning-to-rank datasets further reveal the empirical superiority of our proposed method. We anticipate the theoretical soundness of our method and its strong connection to ranking metrics enable future research and progress.

Our contributions can be summarized as follows:

- We present a cross entropy-based loss function, dubbed xe_{NDCG}, for learning-to-rank and prove that it is a convex bound on negative (translated and log-transformed) mean NDCG;
- We compare model complexity between λMART and xe_{NDCG};

- We formulate an approximation to the inverse Hessian for xE_{NDCG} for optimization with second-order methods; and,
- We optimize xE_{NDCG} to learn Gradient Boosted Regression Trees (denoted by $\text{xE}_{\text{NDCG}}\text{MART}$) and compare its performance and robustness with λMART on benchmark learning-to-rank datasets through extensive experiments.

This article is organized as follows. Section 2 reviews existing work on learning-to-rank. In Section 3, we introduce our notation and formulate the problem. Section 4 presents a detailed description of our proposed learning-to-rank loss function and examines its theoretical properties, including a comparison of bounds on the generalization error. We empirically evaluate our method and report our findings in Section 5. We conclude this work in Section 6.

2 RELATED WORK

A large class of learning-to-rank methods attempt to optimize pairwise misranking error—a popular ranking statistic in many prioritization problems—by learning to correctly classify pairwise preferences. Examples include RankSVM [15] and AdaRank [34] which learn margin classifiers, RankNet [4] which optimizes a probabilistic loss function, and the P-Norm Push method [26] which extends the problem to settings where we mostly care about the top of the ranked list. While the so-called "pairwise" methods typically optimize convex upper-bounds of the misranking error, direct optimization methods based on mathematical programming have also been proposed [27] albeit for linear hypothesis spaces.

Pairwise learning-to-rank methods, while generally effective, optimize loss functions that are misaligned with more complex ranking statistics such as Expected Reciprocal Rank [8] or NDCG [14]. This discrepancy has given rise to the so-called "listwise" methods, where the loss function under optimization is defined over the entire list of items, not just pairs.

Listwise learning-to-rank methods either derive a smooth approximation to ranking metrics or use heuristics to construct smooth surrogate loss functions. Algorithms that represent the first class are SoftRank [28] which takes every score to be the mean of a Gaussian distribution, and ApproxNDCG [24] which approximates the indicator function—used in the computation of ranks given scores—with a generalized sigmoid.

The other class of listwise learning-to-rank methods include ListMLE [33], ListNet [6], and λMART [5, 32]. ListMLE maximizes the log-likelihood based on the Plackett-Luce probabilistic model, a loss function that is disconnected from ranking metrics. ListNet minimizes the cross-entropy between the ground-truth and score distributions. Though a recent work [2] establishes a link between the ListNet loss function and NDCG under strict conditions—requiring binary relevance labels—in a general setting, its loss is only loosely related to ranking metrics.

λMART is a gradient boosting machine [13] that forgoes the loss function altogether and, instead, directly designs the gradients of its unknown loss function using heuristics. While a recent work [31] claims to have found λMART's loss function, it overlooks an important detail: The reported loss function in [31] is not differentiable.

There is abundant evidence to suggest listwise methods are empirically superior to pairwise methods where MRR, ERR, or NDCG determines ranking quality [3, 20, 31]. However, unlike pairwise

methods, listwise algorithms remain theoretically ill-understood. Past studies have examined the generalization error bounds for existing surrogate loss functions [9, 18, 29], but little attention has been paid to the validity of such functions which could shed light on their empirical performance.

3 PRELIMINARIES

In this section, we formalize the problem and introduce our notation. To simplify exposition, we write vectors in bold and use subscripts to index their elements (e.g., $\gamma_i \in \boldsymbol{\gamma}$).

Let $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^m \times \mathcal{Y}^m$ be a training example comprising of m items and relevance labels where $\mathcal{X} \subset \mathbb{R}^d$ is the bounded space of items or item-context pairs represented by d-dimensional feature vectors, and $\mathcal{Y} \subset \mathbb{R}_+$ is the space of nonnegative relevance labels. For consistency with existing work on listwise learning-to-rank, we refer to each $x_i \in \boldsymbol{x}$, $1 \le i \le m$ as a "document." Note, however, that x_i could be the representation of any general item or item-context pair. We assume the training set Ψ consists of n such examples.

We denote a learning-to-rank scoring function by $f : \mathcal{X} \to \mathbb{R}$ and assume $f \in \mathcal{F}$ where $\mathcal{F}$ is a compact hypothesis space of bounded functions endowed with the uniform norm. For brevity, we denote $f(x_i)$ by f_i and, with a slight abuse of notation, define $f(\boldsymbol{x}) = (f_1, f_2, \dots, f_m)$, the vector of scores for m documents in $\boldsymbol{x}$.

As noted in earlier sections, the goal is to learn a scoring function f that minimizes the empirical risk:

$$\mathcal{L}(f) = \frac{1}{|\Psi|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \Psi} \ell(\boldsymbol{y}, f(\boldsymbol{x})), \tag{1}$$

where $\ell(\cdot)$ is by assumption a smooth loss function.

ListNet: The loss ℓ in ListNet [6] first projects labels $\boldsymbol{y}$ and scores $f(\boldsymbol{x})$ onto the probability simplex to form distributions ϕ_{ListNet} and ρ_{ListNet}, respectively. Given the two distributions, the loss is their distance as measured by cross entropy:

$$\ell(\boldsymbol{y}, f(\boldsymbol{x})) \triangleq - \sum_{i=1}^{m} \phi_{\text{ListNet}}(y_i) \log \rho_{\text{ListNet}}(f_i). \tag{2}$$

The distributions ϕ_{ListNet} and ρ_{ListNet} may be understood as encoding the likelihood of document x_i appearing at the top of the ranked list, referred to as "top one" probability, according to the labels and scores respectively. In the original publication [6], ϕ_{ListNet} and ρ_{ListNet} are defined as follows:

$$\phi_{\text{ListNet}}(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{m} e^{y_j}}, \quad \rho_{\text{ListNet}}(f_i) = \frac{e^{f_i}}{\sum_{j=1}^{m} e^{f_j}}. \tag{3}$$

λMART: The loss ℓ in λMART is unknown but its gradients with respect to the scoring function are designed as follows:

$$\frac{\partial \ell}{\partial f_i} = \sum_{y_i > y_j} \frac{\partial \ell_{ij}}{\partial f_i} + \sum_{y_k > y_i} \frac{\partial \ell_{ki}}{\partial f_i}, \tag{4}$$

where

$$\frac{\partial \ell_{mn}}{\partial f_m} = \frac{-\sigma |\Delta_{\text{NDCG}_{mn}}|}{1 + e^{\sigma(f_m - f_n)}} = -\frac{\partial \ell_{nm}}{\partial f_m}, \tag{5}$$

where σ is a hyperparameter and $\Delta_{\text{NDCG}_{mn}}$ is the change in NDCG if documents at ranks m and n are swapped. Finally, NDCG is

defined as follows:

$$\text{NDCG}(\boldsymbol{\pi}_f, \boldsymbol{y}) = \frac{\text{DCG}(\boldsymbol{\pi}_f, \boldsymbol{y})}{\text{DCG}(\boldsymbol{\pi}_\boldsymbol{y}, \boldsymbol{y})}, \tag{6}$$

where $\boldsymbol{\pi}_f$ is a ranked list induced by f on $\boldsymbol{x}$, $\boldsymbol{\pi}_\boldsymbol{y}$ is the ideal ranked list (where $\boldsymbol{x}$ is sorted by $\boldsymbol{y}$), and DCG is defined as follows:

$$\text{DCG}(\boldsymbol{\pi}, \boldsymbol{y}) = \sum_{i=1}^{m} \frac{2^{y_i} - 1}{\log_2(1 + \boldsymbol{\pi}[i])}, \tag{7}$$

with $\boldsymbol{\pi}[i]$ denoting the rank of x_i.

4 PROPOSED METHOD

In this section, we show how a slight modification to the ListNet loss function equips the loss with interesting theoretical properties. To avoid conflating implementation details with the loss function itself, we name our proposed loss function XE_{NDCG}.

DEFINITION 1. *For a training example $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^m \times \mathcal{Y}^m$ and scores $f(\boldsymbol{x}) \in \mathbb{R}^m$, we define XE_{NDCG} as the cross entropy between score distribution ρ and a parameterized class of label distributions ϕ defined as follows:*

$$\rho(f_i) = \frac{e^{f_i}}{\sum_{j=1}^{m} e^{f_j}}, \quad \phi(y_i; \boldsymbol{\gamma}) = \frac{2^{y_i} - \gamma_i}{\sum_{j=1}^{m} 2^{y_j} - \gamma_j}$$

where $\boldsymbol{\gamma} \in [0, 1]^m$.

In effect, the distribution ϕ allocates a mass in the interval $[2^{y_r} - 1, 2^{y_r}]$ for each document indexed with r. As we will explain later, the vector $\boldsymbol{\gamma}$ plays an important role in certain theoretical properties of our proposed loss function. Note that in general, $\boldsymbol{\gamma}$ may be unique to each training example $(\boldsymbol{x}, \boldsymbol{y})$.

4.1 Relationship to NDCG

The difference between XE_{NDCG} and ListNet is minor but consequential: The change to the definition of ϕ leads to our main result.

THEOREM 1. *XE_{NDCG} is an upper-bound on negative (translated and log-transformed) mean Normalized Discounted Cumulative Gain.*

Theorem 1 asserts that XE_{NDCG} is a convex proxy to minimizing negative NDCG (where we turn NDCG which is a utility to a cost by negation). No such analytical link exists between the λMART, List-Net, or other listwise learning-to-rank loss functions and ranking metrics.

In proving Theorem 1 we make use of Jensen's inequality when applied to the log function:

$$\log \mathbb{E}[X] \geq \mathbb{E}[\log X], \tag{8}$$

where X is a random variable and $\mathbb{E}[\cdot]$ denotes expectation. We also use the following bound on ranks that was originally derived in [2]:

$$\boldsymbol{\pi}[r] = 1 + \sum_{i \neq r} \mathbb{1}_{f_i > f_r} = 1 + \sum_{i \neq r} \mathbb{1}_{f_i - f_r > 0}$$

$$\leq 1 + \sum_{i \neq r} e^{(f_i - f_r)} = \sum_{i} e^{(f_i - f_r)} = \frac{\sum_i e^{f_i}}{e^{f_r}},$$

where $\mathbb{1}_p$ is the indicator function taking the value 1 when the predicate p is true and 0 otherwise. The above leads to:

$$\frac{1}{\boldsymbol{\pi}[r]} \geq \frac{e^{f_r}}{\sum_i e^{f_i}} = \rho(f_r). \tag{9}$$

PROOF. Consider $\text{DCG}(\boldsymbol{\pi}_\boldsymbol{y}, \boldsymbol{y})$. Using $\log_2(1 + z) \geq 1$, $\forall z \geq 1$:

$$\text{DCG}(\boldsymbol{\pi}_\boldsymbol{y}, \boldsymbol{y}) = \sum_{i=1}^{m} \frac{2^{y_i} - 1}{\log_2(1 + \boldsymbol{\pi}_\boldsymbol{y}[i])} \leq \sum_{i=1}^{m} 2^{y_i} - \gamma_i, \tag{10}$$

for $0 \leq \gamma_i \leq 1$.

Turning to $\text{DCG}(\boldsymbol{\pi}_f, \boldsymbol{y})$ and using $1 + z \leq 2^z$ for a positive integer z or equivalently $\log_2(1 + z) \leq z$, we have the following:

$$\text{DCG}(\boldsymbol{\pi}_f, \boldsymbol{y}) = \sum_{r} \frac{2^{y_r} - 1}{\log_2(1 + \boldsymbol{\pi}_f[r])} \geq \sum_{r} \frac{2^{y_r} - 1}{\boldsymbol{\pi}_f[r]}$$

$$\geq \sum_{r}(2^{y_r} - 1)\rho(f_r) = \Big[\sum_{r} 2^{y_r}\rho(f_r)\Big] - 1$$

$$\geq \Big[\sum_{r}(2^{y_r} - \gamma_r)\rho(f_r)\Big] - 1, \tag{11}$$

where the second inequality holds by Equation (9).

Finally, consider a translation (by a constant) and log-transformation of mean NDCG, $\overline{\text{NDCG}}$, as follows:

$$\widetilde{\text{NDCG}} \triangleq \log\Big(\overline{\text{NDCG}} + \frac{1}{|\Psi|} \sum_{(\boldsymbol{x}, \boldsymbol{y})} \frac{1}{\text{DCG}(\boldsymbol{\pi}_\boldsymbol{y}, \boldsymbol{y})}\Big).$$

Given the monotonicity of $\log(\cdot)$, the maximizer of $\widetilde{\text{NDCG}}$ also maximizes $\overline{\text{NDCG}}$. We now proceed as follows:

$$\widetilde{\text{NDCG}} = \log \frac{1}{|\Psi|} \sum_{(\boldsymbol{x}, \boldsymbol{y})} \frac{1}{\text{DCG}(\boldsymbol{\pi}_\boldsymbol{y}, \boldsymbol{y})} \big[\text{DCG}(\boldsymbol{\pi}_f, \boldsymbol{y}) + 1\big]$$

$$\geq \log \frac{1}{|\Psi|} \sum_{(\boldsymbol{x}, \boldsymbol{y})} \frac{1}{\sum_j 2^{y_j} - \gamma_j} \big[\text{DCG}(\boldsymbol{\pi}_f, \boldsymbol{y}) + 1\big]$$

$$\geq \log \frac{1}{|\Psi|} \sum_{(\boldsymbol{x}, \boldsymbol{y})} \sum_{r} \phi(y_r)\rho(f_r) \tag{12}$$

$$\geq \frac{1}{|\Psi|} \sum_{(\boldsymbol{x}, \boldsymbol{y})} \sum_{r} \phi(y_r) \log \rho(f_r), \tag{13}$$

where the first inequality holds by Equation (10), the second by Equation (11) and Definition 1, and the last by repeated applications of Equation (8). Negating both sides completes the proof. □

While establishing that $\mathcal{L}$ bounds the NDCG loss is a necessary property in a surrogate, it is not sufficient. As an example, the constant function $\mathcal{L}(f) = 2$ bounds the NDCG loss, but optimizing it does not lead to an optimal f^*. This is where the notion of Fisher consistency becomes critical: In summary, a loss function is consistent with an evaluation measure, if the optimal solution to the loss function is also an optimal solution of the evaluation measure.

The cross entropy function that is at the heart of ListNet and our proposed method was shown to be consistent with the $0 - 1$ ranking loss in [33]. In general, however, the loss is not consistent with NDCG [25]. But under special conditions that are common in information retrieval, cross entropy (and as a result XE_{NDCG}) become NDCG-consistent.

CLAIM 1. *XE_{NDCG} is NDCG-consistent on datasets with graded relevance judgments or with a single click per query.*

PROOF. We omit a complete proof due to space constraints, but note that the above is a trivial consequence of the conditions. Briefly: [25] shows that $\mathcal{L}$ is NDCG-consistent so long as its terms,

$\ell(\boldsymbol{y}, f)$, are each normalized by the best DCG, $DCG(\boldsymbol{\pi_y}, \boldsymbol{y})$. In settings where queries receive a single click, the best DCG is simply 1, and so a $0-1$ loss-consistent surrogate is naturally NDCG-consistent too. Furthermore, when queries do not repeat as in datasets with graded relevance labels, every query has a unique relevance vector. That degenerate relevance probability distribution renders the expectation and thus normalization terms irrelevant, thereby equipping a $0-1$ loss-consistent surrogate with NDCG-consistency. □

4.2 Comparison with λMART

In this section, we compare XE_{NDCG} with λMART in terms of model complexity and generalization error. In what follows, we proceed under the strong assumption that the loss optimized by λMART in fact exists. That is, we assume that there exists a differentiable function that satisfies Equation (4).

We begin with an examination of the Lipschitz constant of the two algorithms—an upper-bound on the variation a function can exhibit. Intuitively, functions with a smaller Lipschitz constant are simpler and thus generalize better.

CLAIM 2. *The λMART loss is σm^2-Lipschitz with respect to $\|\cdot\|_\infty$.*

PROOF. Recall the definition of the Lipschitz constant for a differentiable function $h(\cdot)$:

$$Lip_h = \sup \frac{|h(f) - h(f')|}{\|f - f'\|}$$
$$= \sup \frac{|\nabla_f h(f'')(f - f')|}{\|f - f'\|} = \|\nabla_f h\|_*,$$

where the second equality holds by the Mean Value Theorem and the last by the definition of the dual norm, $\|\cdot\|_*$. Therefore, to derive the Lipschitz constant of a function with respect to the infinity norm, it is sufficient to calculate the L_1 norm of its gradient. Given that λMART's loss function is unknown, we resort to this strategy to derive its Lipschitz constant.

Observe that the terms in Equation (5) are bounded by σ and Equation (4) has at most m such terms. As such, we have that,

$$|\frac{\partial \ell}{\partial f_i}| \le \sigma m.$$

Then,

$$\|\nabla_f \ell\|_1 = \sum_{i=1}^m |\frac{\partial \ell}{\partial f_i}| \le \sum_{i=1}^m \sigma m = \sigma m^2$$

which completes the proof. □

PROPOSITION 1. *XE_{NDCG} is 2-Lipschitz with respect to $\|\cdot\|_\infty$.*

PROOF. Recall that the cost function $\ell(\cdot)$ for XE_{NDCG} is defined as follows:

$$\ell(\boldsymbol{y}, f(\boldsymbol{x})) \triangleq - \sum \phi(y_i) \log \rho(f_i),$$

where ϕ and ρ form probability distributions over labels $\boldsymbol{y}$ and scores $f(\boldsymbol{x})$ respectively, and $f_i = f(x_i)$.

Observe that the derivative of the cost function ℓ with respect to a score f_r is:

$$\frac{\partial \ell}{\partial f_r} = \frac{\partial}{\partial f_r}[- \sum_i \phi(y_i)(f_i - \log \sum_j e^{f_j})]$$
$$= \frac{\partial}{\partial f_r}[(\sum_i -\phi(y_i)f_i) + \log \sum_j e^{f_j}]$$
$$= -\phi(y_r) + \frac{e^{f_r}}{\sum_j e^{f_j}} = -\phi(y_r) + \rho(f_r).$$

We then have that,

$$|\frac{\partial \ell}{\partial f_r}| \le \phi(y_r) + \rho(f_r),$$

resulting in,

$$\|\nabla_f \ell\|_1 = \sum |\frac{\partial \ell}{\partial f_i}| \le \sum (\phi(y_r) + \rho(f_r)) = 2$$

as required. □

In order to put this difference into perspective, we use the results above to derive bounds on the generalization error of the two algorithms. But first we need the following result.

THEOREM 2. *Let $\mathcal{F}$ be a compact space of bounded functions from $\mathcal{X}$ to $[0, 1]$, $n = |\Psi|$ be the number of training examples, Lip_ℓ the Lipschitz constant of loss function ℓ, and $\mathfrak{N}(\frac{\epsilon}{4Lip_\ell}, \mathcal{F}, \|\cdot\|_\infty)$ the covering number of $\mathcal{F}$ by L_∞ balls of radius $\frac{\epsilon}{4Lip_\ell}$. The following generalization error bound holds:*

$$\mathcal{P}\{\mathcal{E}(f) \le \epsilon\} \ge 1 - 2\mathfrak{N}(\frac{\epsilon}{4Lip_\ell}, \mathcal{F}, \|\cdot\|_\infty) exp(\frac{-2n\epsilon^2}{Lip_\ell^2}),$$

where the generalization error $\mathcal{E}$ is defined as follows:

$$\mathcal{E}(f) \triangleq \underset{\mathcal{X}^m \times \mathcal{Y}^m}{\mathbb{E}}[\ell(\boldsymbol{y}, f(\boldsymbol{x}))] - \frac{1}{n} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \Psi} \ell(\boldsymbol{y}, f(\boldsymbol{x})).$$

PROOF. Based on the proofs in [12, 26]. □

The dependence of the generalization error bound on the Lipschitz constant suggests that unlike λMART, XE_{NDCG}'s generalization error does not degrade as the number of documents per training example increases. Furthermore, given its larger Lipschitz constant and potentially higher complexity, we hypothesize that λMART is less robust to noise and generalizes poorly in settings where the number of documents per training example is large.

We note that, the independence of the ListNet generalization error bound from m was also reported in [29] for linear models, but we present the (structure of the) bounds here to allow a direct comparison between λMART and XE_{NDCG}.

We conclude this section with the following note: It is true that the Lipschitz constant is only a loose measure of the complexity of a function. We naturally do not expect the bounds to hold exactly in practice, but we expect the bounds to hint at an algorithm's behavior in extremes. As our experiments show later, an empirical comparison of the two functions is in alignment with the analysis above: As the experimental setup approaches more extreme levels of noise, a likely scenario in click data, the two algorithms behave very differently.

4.3 Approximating the Inverse Hessian

In this work, we fix the hypothesis space, $\mathcal{F}$, to Gradient Boosted Regression Trees. This is, in part, because we are interested in a fair comparison of ListNet, XE_{NDCG}, and λMART in isolation of other factors, as explained in Section 5. As most GBRT learning algorithms use second-order optimization methods (e.g., Newton's), however, we must approximate the inverse Hessian for ListNet and XE_{NDCG}.

Unfortunately, XE_{NDCG} as defined in Definition 1 results in a Hessian that is singular, making the loss incompatible with a straightforward implementation of Newton's second-order method. We resolve this technical difficulty by making a small adjustment to the formulation of the loss function.

Let us re-define the score distribution, ρ, from Definition 1 as follows for a negligible $\epsilon > 0$:

$$\rho(f_i) = \frac{e^{f_i}}{\sum_{j=1}^{m} e^{f_j} + \epsilon}. \tag{14}$$

In effect, we take away a small probability mass, $\rho(f_{m+1}) = \epsilon/(\sum e^{f_j} + \epsilon)$, from the score distribution for a nonexistent, $m + 1^{\text{th}}$ document with label probability $\phi(f_{m+1}) = 0$. The gradients of the loss will take the following form:

$$\frac{\partial \ell}{\partial f_r} = \frac{\partial}{\partial f_r} \left[\sum_i (-\phi(y_i)f_i) + \log(\sum_j e^{f_j} + \epsilon) \right]$$

$$= -\phi_r + \rho_r,$$

where $\phi_r = \phi(y_r)$ and $\rho_r = \rho(f_r)$. The Hessian looks as follows:

$$H_{ij} = \begin{cases} \rho_i(1 - \rho_i), & i = j \\ -\rho_i\rho_j, & i \neq j \end{cases}$$

CLAIM 3. *The Hessian, as defined above, is positive definite.*

PROOF. A complete proof may be found in the appendix. Observe that H is strictly diagonally dominant:

$$|H_{kk}| = \rho_k(1 - \rho_k) = \rho_k(1 - \frac{e^{f_k}}{\sum e^{f_j} + \epsilon})$$

$$= \rho_k \frac{\sum_{j \neq k} e^{f_j} + \epsilon}{\sum e^{f_j} + \epsilon} > \rho_k \sum_{j \neq k} \rho_j = \sum_{j \neq k} |H_{kj}|.$$

By the properties of strictly diagonally dominant matrices and the fact that the diagonal elements of H are positive, we have that $H \succ 0$ and therefore invertible. □

We now turn to approximating the inverse of H as required. Write $H = D(I - S)$ where I is the identity matrix, D is a diagonal matrix where $D_{ii} = \rho_i(1 - \rho_i)$ and S is a square matrix where,

$$S_{ij} = \begin{cases} 0, & i = j \\ \rho_j/(1 - \rho_i), & i \neq j \end{cases}.$$

CLAIM 4. *The spectral radius of S is strictly less than 1.*

PROOF. A complete proof is presented in the appendix. S is a square matrix with nonnegative entries. By the Perron-Frobenious theorem, its spectral radius is bounded above by the maximum row-wise sum of entries, which, in S, is strictly less than 1. □

Claim 4 allows us to apply Neumann's result to approximate $(I - S)^{-1}$ as follows:

$$(I - S)^{-1} = \sum_{k=0}^{\infty} S^k \approx I + S + S^2.$$

Using this result, we may approximate H^{-1} as follows:

$$H^{-1} = (I - S)^{-1}D^{-1} \approx (I + S + S^2)D^{-1}$$

With that, we can finally calculate the update rule in Newton's method which requires the quantity $H^{-1}\nabla$:

$$(H^{-1}\nabla)_k = \sum_i H_{ki}^{-1}\nabla_i$$

$$\approx \sum_i (I + S + S^2)_{ki}(D^{-1}\nabla)_i$$

$$= \sum_i (I + S + S^2)_{ki} \frac{-\phi_i + \rho_i}{\rho_i(1 - \rho_i)}$$

$$= \underbrace{\frac{-\phi_k + \rho_k}{\rho_k(1 - \rho_k)}}_{(ID^{-1}\nabla)_k} + \underbrace{\frac{1}{1 - \rho_k}\sum_{i \neq k}\frac{-\phi_i + \rho_i}{1 - \rho_i}}_{(SD^{-1}\nabla)_k} + \sum_{i \neq k}\frac{\rho_i(SD^{-1}\nabla)_i}{1 - \rho_k}$$

$$= \frac{-\phi_k + \rho_k + \rho_k \sum_{i \neq k}\frac{-\phi_i + \rho_i}{1 - \rho_i} + \rho_k \sum_{i \neq k}\rho_i(SD^{-1}\nabla)_i}{\rho_k(1 - \rho_k)}.$$

5 EXPERIMENTS

We are largely interested in a comparison of (a) the overall performance of ListNet, λMART, and XE_{NDCG} on benchmark learning-to-rank datasets, and (b) their robustness to various types and degrees of noise as a proxy to complexity. In this section, we describe our experimental setup and report our empirical findings.

It is important to note that there is an extensive list of published work [1, 3, 22, 36] that compare learning-to-rank algorithms on benchmark datasets we use in this work. We rely on prior research and do not include methods that have been shown repeatedly to be weaker than λMART, including BoltzRank [30], ListMLE [33], Position-Aware ListMLE [19], SoftRank [28], ApproxNDCG [3, 24], or other direct optimization methods [21, 35].

5.1 Datasets

We conduct experiments on two publicly available benchmark datasets: MSLR Web30K [23] and Yahoo! Learning to Rank Challenge Set 1 [7]. Web30K contains roughly 30,000 examples, with an average of 120 documents per example. Documents are represented by 136 numeric features. Yahoo! also has about the same number of examples but with an average of 24 documents per example, each represented by 519 features. Documents in both datasets are labeled with graded relevance from 0 to 4 with larger labels indicating a higher relevance.

From each dataset, we sample training (60%), validation (20%), and test (20%) examples, and train and compare models on the resulting splits. We repeat this procedure 100 times and obtain mean NDCG at different rank cutoffs for each trial. We subsequently compare the ranking quality between pairs of models and determine statistical significance of differences using a paired t-test.

Table 1: NDCG (percentage) on test sets at rank cutoffs 5 and 10, averaged over 100 randomized trials. In each trial, training, validation, and test sets are sampled from the datasets. The differences at all rank cutoffs between all models are statistically significant according to a paired *t*-test ($\alpha = 0.01$).

	Web30K		Yahoo!	
Model	@5	@10	@5	@10
ListNet	47.68	49.76	71.76	76.52
λMART	48.08	49.94	73.00	77.49
xE$_{\text{NDCG}}$MART	48.23	50.27	73.37	77.84

During evaluation, we discard examples with no relevant documents. There are 982 and 1,135 such examples in the Web30K and Yahoo! datasets. The reason for ignoring these examples during evaluation is that their ranking quality can be arbitrarily 0 or 1, which only skews the average.

5.2 Models

We train λMART models using LightGBM [17]. The hyperparameters are guided by previous work [2, 17, 31]. For Web30K, max_bin is 255, learning_rate is 0.02, num_leaves is 400, min_data_in_leaf is 50, min_sum_hessian_in_leaf is set to 0, σ is 1, and lambdamart_norm is set to false. For Yahoo!, num_leaves is 200 and min_data_in_leaf is 100. We use NDCG@5 to select the best models on validation sets and fix early stopping round to 50 up to 500 trees.

We also implemented ListNet and xE$_{\text{NDCG}}$ in LightGBM. [1] As noted earlier, by fixing the hypothesis space to gradient boosted trees, we aim to strictly compare the performance of the loss functions and shield our analysis from any effect the hypothesis space may have on convergence and generalization. An additional reason for choosing gradient boosted trees is that, recent evidence [3, 22, 36] confirm their superior performance against other hypothesis spaces such as deep neural networks, at least on the benchmark datasets we use in this work. We refer to the tree-based model trained by optimizing xE$_{\text{NDCG}}$ as xE$_{\text{NDCG}}$MART. For training purposes, we use the same hyperparameters above.

Finally, we must address the choice for γ in xE$_{\text{NDCG}}$MART. In this work, we simplify the choice by sampling γ uniformly from $[0, 1]^m$ for every training example (with m documents) and at every iteration of boosting. We leave a detailed examination of the effect of this parameter to a future study.

5.3 Ranking Quality

We compare the ranking quality of the three models under consideration. We report model quality by measuring average NDCG at rank cutoffs 5 and 10. As noted earlier, we also measure statistical significance in the difference between model qualities using a paired *t*-test with significance level set to $\alpha = 0.01$. Our results are summarized in Table 1.

From Table 1, we observe that ListNet consistently performs poorly across both datasets. The quality gap between ListNet and
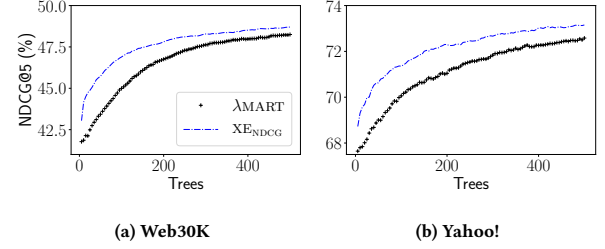
[1]Available at github.com/microsoft/LightGBM.

(a) Web30K **(b) Yahoo!**

Figure 1: NDCG@5 on validation sets during training of λMART and xE$_{\text{NDCG}}$MART in a representative trial.

λMART is statistically significant at all rank cutoffs. This observation is in agreement with past studies [2].

On the other hand, our proposed xE$_{\text{NDCG}}$MART yields a significant improvement over ListNet. This observation holds consistently across both datasets and rank cutoffs and lends support to our theoretical findings in previous sections.

Not only does xE$_{\text{NDCG}}$MART outperform ListNet, its performance surpasses that of λMART's. While xE$_{\text{NDCG}}$MART's gain over λMART is smaller than its gap with ListNet, the differences are statistically significant. This is an encouraging result: xE$_{\text{NDCG}}$MART is not only theoretically sound and is equipped with better properties, it also performs well empirically compared to the state-of-the-art algorithm.

A notable difference between λMART and xE$_{\text{NDCG}}$MART is in their convergence rate. Figure 1 plots NDCG@5 on validation sets as more trees are added to the ensemble. To avoid clutter, the figure illustrates just one trial (out of 100) but we note that we observe a similar trend across trials. From Figure 1, it is clear that xE$_{\text{NDCG}}$MART outperforms λMART by a wider margin when the number of trees in the ensemble is small. This property is important in latency-sensitive applications where a smaller ensemble is preferred.

5.4 Robustness to Noise in Graded Labels

We now turn to model robustness where we perform a comparative analysis of the effect of noise on λMART and xE$_{\text{NDCG}}$MART. The robustness of a ranking model to noise is important in practice due to uncertainty in relevance labels, whether judged by human experts or deduced from user feedback such as clicks. We expect λMART to overfit to noise and be less robust due to its higher model complexity—see findings in Section 4.2. As such, we expect the performance of λMART to degrade at a higher pace than xE$_{\text{NDCG}}$MART as we inject more noise into the dataset. We put this hypothesis to the test through two types of experiments.

In the first series of experiments, we focus on the effect of enlarging the document list per training example by the addition of noise. In particular, we augment document lists for training examples with negative documents using the following protocol. For every training example (x, y), we sample from the collection of all documents in the training set excluding x to form $x' = \{\overline{x} \mid (\overline{x}, \overline{y}) \sim (\overline{x}, \overline{y}), (\overline{x}, \overline{y}) \sim \Psi \setminus (x, y)\})$. Subsequently, we augment x by adding x' as non-relevant documents: $(x \oplus x', y \oplus 0)$,
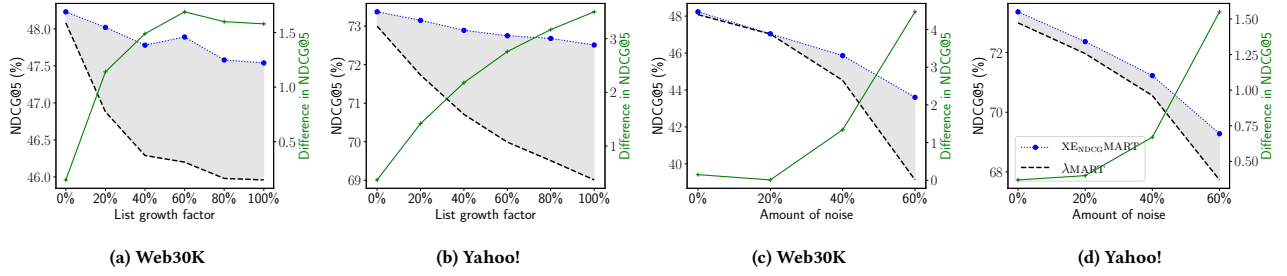
Figure 2: Mean NDCG@5 on test sets, averaged over 100 trials. To reduce clutter, legends appear only in the rightmost figure. In each trial, training, validation, and test sets are sampled from the dataset. In (a) and (b), training examples are augmented by additional (randomly sampled) negative documents. For example, the data point at "40%" indicates a 40% increase in the number of documents for every example. In (c) and (d), a percentage of relevance labels are set to a random value. The solid (green) lines show the difference in NDCG@5 between the two models in absolute terms.

where $\oplus$ denotes concatenation. Finally, we train models on the resulting training set and evaluate on the (unmodified) test set. As before, we repeat this experiment 100 times.

We illustrate NDCG@5 on the test sets averaged over 100 trials and for various degrees of augmentation in Figures 2a and 2b. The trend confirms our hypothesis: On both datasets, the performance of λMART degrades more severely as more noise is added to the training set, increasing the number of documents per example (m). This effect is more pronounced on the Yahoo! dataset where m is on average small. We note that the increase in NDCG@5 of XE_{NDCG}MART from the 40% mark to 60% on Web30K is not statistically significant.

In another series of experiments we perturb relevance labels in the training set. To that end, for each training example (x, y), we randomly choose a subset of its documents and set their labels (independently) to 0 through 4 with decreasing probabilities: $p(0) = .5$, $p(1) = .2$, $p(2) = .15$, $p(3) = .1$, $p(4) = .05$. We train models on the perturbed training set and evaluate on the (unmodified) test set. As before, we repeat this experiment 100 times.

The results are shown in Figures 2c and 2d. We observe once more that λMART's performance degrades more rapidly with more noise. This behavior is more pronounced on Web30K.

5.5 Robustness to Noise in Simulated Clicks

In Section 5.4, we examined the behavior of λMART and XE_{NDCG}MART in the presence of noise on datasets with explicit relevance judgments. In this section, we provide an analysis of the robustness of the two algorithms on a simulated click dataset where noise occurs more naturally (e.g., where a user clicks a non-relevant document by accident).

We follow the procedure proposed in [16] to simulate a user in the cascade click model [11]. In the cascade click model, when presented with a ranked list of documents, a user scans the list sequentially from the top and clicks a document according to a *click probability* distribution—the probability that a document is clicked given its relevance label. We assume the user is persistent in that they continue to examine the list until either a document is clicked or they reach the end of the list.

We construct click datasets as follows. We first create training and validation splits using the procedure of Section 5.1. Given a training (or validation) example (x, y) consisting of m documents and relevance labels, we shuffle its elements and sequentially scan the resulting list to produce clicks using the cascade click model. We stop at the first occurrence of a click and return the list up to the first click as an "impression." We create 10 impressions per training example to form our click dataset. Finally, we train ranking models on the click dataset and evaluate on the original (non-click) test set. We repeat this experiment 20 times and measure mean NDCG.

In our experiments, we adjust the click probability of non-relevant documents to simulate noise in the training set. We begin with click probabilities set to $(.05, .3, .5, .7, .95)$ for relevance labels 0 through 4, respectively. That is, in this setting, a non-relevant document is clicked 5% of the time. In subsequent experiments, we increase the click probability of non-relevant documents by .05.

The results of our experiments on Web30K and Yahoo! are plotted in Figure 3. Clearly, the performance of XE_{NDCG}MART on the test sets is consistently better than λMART for all levels of noise in the training set. We note that all differences are statistically significant according to a paired t-test ($\alpha = 0.01$). Additionally, as with previous experiments, the performance of XE_{NDCG}MART is more robust to noise: its performance degrades more slowly than λMART.

6 CONCLUSION

In this work, we presented a novel "listwise" learning-to-rank loss function, XE_{NDCG}, that, unlike existing methods bounds NDCG—a popular ranking metric—in a general setting. We contrasted our proposed loss function with λMART and showed its superior theoretical properties. In particular, we showed that the loss function optimized by λMART (if it exists), has a higher complexity with a Lipschitz constant that is a function of the number of documents, m. In contrast, the complexity of XE_{NDCG} is invariant to m.

Furthermore, we proposed a model that optimizes XE_{NDCG} to learn an ensemble of gradient-boosted decision trees which we refer to as XE_{NDCG}MART. Through extensive experiments on two benchmark learning-to-rank datasets, we demonstrated that our
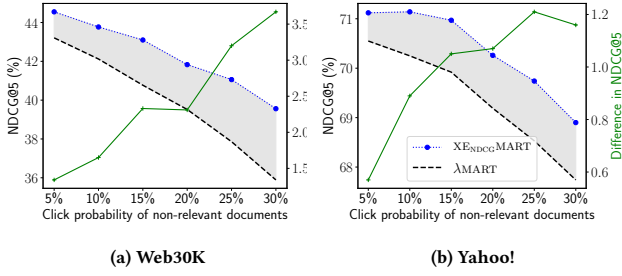
(a) Web30K **(b) Yahoo!**

Figure 3: Mean NDCG@5 on test sets, averaged over 20 trials. In each trial, training and validation sets are turned into clicks using the cascade click model and a random base ranker. The horizontal axis indicates the click probability of non-relevant documents. The solid (green) lines show the difference in NDCG@5 between the two models.

proposed method performs better than ListNet and λMART in terms of quality and robustness. We showed that, XE_{NDCG}MART is less sensitive to the number of documents and is more robust in the presence of noise. Finally, our experiments suggest that the performance gap between XE_{NDCG}MART and λMART widens if we constrain the size of the learned ensemble. Better performance with fewer trees is important for latency-sensitive applications.

As a future direction, we are interested in an examination of the bound and its effect on the convergence and consistency of XE_{NDCG}. In particular, in this work, we treated γ's as hyperparameters. However, more effective strategies for solving γ's and obtaining tighter bounds during boosting remain unexplored. Furthermore, given its robustness to label noise (implicit and explicit), we are also interested in studying XE_{NDCG} in an online learning setting.

ACKNOWLEDGMENTS

A APPENDIX

A.1 Proof of Claim 3

Using $\rho_r = \rho(f_r) = e^{f_r}/(\sum e^{f_j} + \epsilon)$ to denote the score probability of the rth document, the Hessian can be written as follows:

$$H_{ij} = \begin{cases} \rho_i(1 - \rho_i), & i = j \\ -\rho_i\rho_j, & i \neq j \end{cases}$$

CLAIM. *The Hessian, as defined above, is positive definite.*

PROOF. We first prove that H is strictly diagonally dominant. By definition, a square matrix A is strictly diagonally dominant if the following holds for all i: $|A_{ii}| > \sum_{j \neq i} |A_{ij}|$. Observe that:

$$|H_{kk}| = \rho_k(1 - \rho_k) = \rho_k(1 - \frac{e^{f_k}}{\sum e^{f_j} + \epsilon})$$

$$= \rho_k \frac{\sum_{j \neq k} e^{f_j} + \epsilon}{\sum e^{f_j} + \epsilon} > \rho_k \sum_{j \neq k} \rho_j = \sum_{j \neq k} |H_{kj}|.$$

Using this property, we now prove nonsingularity of H by contradiction. Assume there exists a vector $\boldsymbol{u} \neq \boldsymbol{0}$ such that $H\boldsymbol{u} = \boldsymbol{0}$. Let i be the index of the u_i with the largest magnitude: $i = \arg\max_i |u_i|$. Then:

$$\sum_j H_{ij}u_j = 0 \Rightarrow H_{ii}u_i = -\sum_{j \neq i} H_{ij}u_j \overset{u_i \neq 0}{\Rightarrow} H_{ii} = -\sum_{j \neq i} \frac{u_j}{u_i}H_{ij}$$

$$\Rightarrow |H_{ii}| \leq \sum_{j \neq i} |\frac{u_j}{u_i}H_{ij}| \Rightarrow |H_{ii}| \leq \sum_{j \neq i} |H_{ij}|,$$

which is a contradiction. This concludes the proof for nonsingularity of the Hessian, which is already sufficient for subsequent results. However, as a consequence of the Gershgorin circle theorem it can further be shown that, because the diagonal elements of H are strictly positive, H is positive definite. □

A.2 Proof of Claim 4

Use $\rho_r = \rho(f_r) = e^{f_r}/(\sum e^{f_j} + \epsilon)$ to denote the score probability of the rth document. The nonnegative, square matrix S in Claim 4 is defined as follows:

$$S_{ij} = \begin{cases} 0, & i = j \\ \rho_j/(1 - \rho_i), & i \neq j \end{cases}.$$

CLAIM. *The spectral radius of S is strictly less than 1.*

PROOF. Note that, for all eigenvalues λ_i, $1 \leq i \leq n$ of an $n \times n$ matrix A, their corresponding eigenvectors $\boldsymbol{u}_i$, and for any induced operator norm $\|\cdot\|$ we have that:

$$\|A\| = \sup_{\boldsymbol{x}} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \geq \frac{\|A\boldsymbol{u}_i\|}{\|\boldsymbol{u}_i\|} = \frac{\|\lambda_i\boldsymbol{u}_i\|}{\|\boldsymbol{u}_i\|} = |\lambda_i|, \forall i.$$

This is, in particular, true for the infinity norm:

$$\|A\|_\infty = \max_i \sum_j |A_{ij}|.$$

The inequality above holds for the spectral radius of A which is defined as the largest absolute value of A's eigenvalues: $\max_i |\lambda_i|$. Therefore, we have that the spectral radius of S is bounded above by:

$$\max_i \sum_j |S_{ij}| = \max_i \sum_j \frac{\rho_j}{1 - \rho_i} = \max_i \frac{\sum_{j \neq i} \rho_j}{1 - \rho_i}$$

$$= \max_i \frac{1 - \rho_i - \epsilon'}{1 - \rho_i} = \max_i 1 - \frac{\epsilon'}{1 - \rho_i} < 1,$$

where $\epsilon' = \epsilon/\sum e^{f_k} + \epsilon$. That completes the proof. □

REFERENCES

[1] Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. 2020. A Stochastic Treatment of Learning to Rank Scoring Functions. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 61–69.

[2] Sebastian Bruch, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2019. An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on the Theory of Information Retrieval*.

[3] Sebastian Bruch, Masrour Zoghi, Mike Bendersky, and Marc Najork. 2019. Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*. 89–96.

[5] Christopher J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report MSR-TR-2010-82. Microsoft Research.

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

[7] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. 1–24.

[8] Olivier Chapelle, Donald Metzler, Ya Zhang, and Pierre Grinspan. 2009. Expected Reciprocal Rank for Graded Relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. 621–630.

[9] Olivier Chapelle and Mingrui Wu. 2010. Gradient Descent Optimization of Smoothed Information Retrieval Metrics. *Information Retrieval* 13, 3 (June 2010), 216–235.

[10] Stéphan Clémençon and Nicolas Vayatis. 2008. Empirical Performance Maximization for Linear Rank Statistics. In *Proceedings of the 21st International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada). 305–312.

[11] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-bias Models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining* (Palo Alto, California, USA). 87–94.

[12] Felipe Cucker and Steve Smale. 2002. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.* 39 (2002), 1–49.

[13] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* 29, 5 (2001), 1189–1232.

[14] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

[15] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 217–226.

[16] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom). 781–789.

[17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*. 3146–3154.

[18] Yanyan Lan, Tie-Yan Liu, Zhiming Ma, and Hang Li. 2009. Generalization Analysis of Listwise Learning-to-rank Algorithms. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada). 577–584.

[19] Yanyan Lan, Yadong Zhu, Jiafeng Guo, Shuzi Niu, and Xueqi Cheng. 2014. Position-Aware ListMLE: A Sequential Learning Process for Ranking. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence* (Quebec City, Quebec, Canada). 449–458.

[20] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[21] Donald A Metzler, W Bruce Croft, and Andrew Mccallum. 2005. *Direct maximization of rank-based metrics for information retrieval*. CIIR report 429. University of Massachusetts.

[22] Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. Self-Attentive Document Interaction Networks for Permutation Equivariant Ranking. arXiv:1910.09676

[23] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. arXiv:1306.2597

[24] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval* 13, 4 (2010), 375–397.

[25] Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. 2011. On NDCG Consistency of Listwise Ranking Methods. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Vol. 15. PMLR, 618–626.

[26] Cynthia Rudin. 2009. The P-Norm Push: A Simple Convex Ranking Algorithm That Concentrates at the Top of the List. *Journal of Machine Learning Research* 10 (Dec. 2009), 2233–2271.

[27] Cynthia Rudin and Yining Wang. 2018. Direct Learning to Rank and Rerank. In *Proceedings of Artificial Intelligence and Statistics AISTATS*.

[28] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. SoftRank: Optimizing Non-smooth Rank Metrics. In *Proceedings of the 1st International Conference on Web Search and Data Mining*. 77–86.

[29] Ambuj Tewari and Sougata Chaudhuri. 2015. Generalization Error Bounds for Learning to Rank: Does the Length of Document Lists Matter?. In *Proceedings of the 32nd International Conference on Machine Learning* (Lille, France). 315–323.

[30] Maksims N. Volkovs and Richard S. Zemel. [n.d.]. BoltzRank: learning to maximize expected ranking gain. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada). 1089–1096.

[31] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy). 1313–1322.

[32] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13, 3 (2010), 254–270.

[33] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*. 1192–1199.

[34] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 391–398.

[35] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. 2008. Directly Optimizing Evaluation Measures in Learning to Rank. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Singapore, Singapore). 107–114.

[36] Honglei Zhuang, Xuanhui Wang, Michael Bendersky, Alexander Grushetsky, Yonghui Wu, Petr Mitrichev, Ethan Sterling, Nathan Bell, Walker Ravina, and Hai Qian. 2020. Interpretable Learning-to-Rank with Generalized Additive Models. arXiv:2005.02553