An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance

Sebastian Bruch, Xuanhui Wang, Michael Bendersky, Marc Najork
Google Research
{bruch,xuanhui,bemike,najork}@google.com

ABSTRACT

One of the challenges of learning-to-rank for information retrieval is that ranking metrics are not smooth and as such cannot be optimized directly with gradient descent optimization methods. This gap has given rise to a large body of research that reformulates the problem to fit into existing machine learning frameworks or defines a surrogate, ranking-appropriate loss function. One such loss is ListNet's [4] which measures the cross entropy between a distribution over documents obtained from scores and another from ground-truth labels. This loss was designed to capture permutation probabilities and as such is considered to be only loosely related to ranking metrics. In this work, however, we show that the above statement is not entirely accurate. In fact, we establish an analytical connection between ListNet's loss and two popular ranking metrics in a learning-to-rank setup with binary relevance labels. In particular, we show that the loss bounds Mean Reciprocal Rank and Normalized Discounted Cumulative Gain. Our analysis sheds light on ListNet's behavior and explains its superior performance on binary labeled data over data with graded relevance.

CCS CONCEPTS

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

KEYWORDS

Learning to Rank with Binary Relevance; Learning to Rank Loss Functions; Softmax Cross Entropy Ranking Loss

ACM Reference Format:
Sebastian Bruch, Xuanhui Wang, Michael Bendersky, Marc Najork. 2019. An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance. In *The 2019 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '19), October 2–5, 2019, Santa Clara, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3341981.3344221

1 INTRODUCTION AND RELATED WORK

Ranking is a central problem in information retrieval with applications in recommender systems, question answering, and *ad hoc* document retrieval or search, where the goal is to order a set of documents in decreasing order of relevance given a query. A machine learning formulation of this problem is known as learning-to-rank.

Learning-to-rank algorithms [13] are supervised methods that learn a parameterized function from the joint space of queries and sets of documents into the space of permutations. Due to the factorial growth of permutations, the learning task quickly becomes intractable. This is generally addressed by breaking the problem into two parts, known as the *score-and-sort* approach: The first step is to learn a *scoring* function that takes a query and a set of n documents into a vector in $\mathbb{R}^n$, where the i^{th} element is a measure indicating the relevance of the i^{th} document with respect to the given query, and in a second step sorting documents by relevance scores. With a few exceptions [7, 17], most learning-to-rank methods [1–4, 10, 11, 18, 21, 22] further reduce the problem to that of learning a *univariate* scoring function that produces a score in $\mathbb{R}$ for a single query-document pair.

Ideally, the parameters of a learning-to-rank scoring function are learned by directly maximizing a ranking utility. While this is plausible when the scoring function is affine [14], in its more general form this proves challenging. This is because ranking metrics such as Normalized Discounted Cumulative Gain (NDCG) [9] are functions of permutations to the real line. The discrete nature of permutations entails that ranking metrics are non-smooth with respect to the scoring function: small perturbations of scores may not necessarily lead to a change in the order of documents, leading to discontinuity at a measure-zero set and flat regions elsewhere.

The non-differentiability and otherwise uninformative gradients of ranking utilities pose a challenge that the learning-to-rank community has sought to study. Broadly speaking, the existing literature offers a range of methods where either the problem is reformulated so it fits into existing machine learning frameworks or a differentiable surrogate, ranking-appropriate loss is minimized in an indirect attempt to maximize metrics. The first class describes algorithms such as RankSVM [10] and RankNet [1] which focus on the correct classification of ordered pairs, and AdaRank [23] which optimizes an exponential upper-bound of metrics using boosted weak learners. The second category includes most other methods [3, 4, 16, 18, 20–22]. These constitute the so-called "listwise" algorithms—the objective is defined over the list of documents.

Some listwise methods are derived from ranking metrics [16, 18, 20] or are partially based on metrics (e.g., LambdaRank [1] or its gradient boosted regression tree [8] variant, LambdaMART [3]). Others such as ListMLE [22] or ListNet [4] have been known to be disconnected from ranking metrics. For example, ListNet's loss function takes a probabilistic view of permutations and defines a "top one" probability distribution over documents where the mass allocated to a document indicates its likelihood of appearing at the top of the ranked list. It forms two such distributions by way of projecting a list of labels and a list of scores onto the probability simplex (using the softmax operator to normalize scores). Finally,

ICTIR '19, October 2–5, 2019, Santa Clara, CA, USA

ACM ISBN 978-1-4503-6881-0/19/10.
https://doi.org/10.1145/3341981.3344221

the distance between these two distributions as measured by cross entropy forms the loss. We refer to this as the softmax cross entropy loss function.

While the softmax cross entropy loss is seemingly disconnected from ranking metrics, in this work we prove that there indeed exists a link between the two concepts under certain conditions. In particular, we show that softmax cross entropy is a bound on Mean Reciprocal Rank (MRR) as well as NDCG when working with binary ground-truth labels. We hope the analysis presented in this work furthers our collective understanding of this loss function and explains its behavior in the presence of binary relevance labels.

The remainder of this paper is structured as follows. In Section 2, we formulate the problem of learning-to-rank and present a formal overview of the softmax cross entropy loss as used in ListNet. Section 3 presents our analysis. We discuss the findings and conclude the paper in Sections 4 and 5.

2 BACKGROUND AND NOTATION

In this section, we formulate the problem of learning-to-rank and present an overview of ranking metrics as well as the softmax cross entropy loss function used in ListNet [4].

2.1 Overview of Learning-to-Rank

In this work, we focus exclusively on the classic case of offline, supervised learning-to-rank methods where we have access to a set of training examples. Let us denote a training set with $\Psi = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^n \times \mathbb{R}_+^n\}$, where $\boldsymbol{x}$ is a vector of n items x_i, $1 \leq i \leq n$, $\boldsymbol{y}$ is a real vector of n nonnegative relevance labels y_i, $1 \leq i \leq n$, and $\mathcal{X}$ is the space of all documents. Each document x_i could generally take any form but throughout this paper we define it to be a vector of features representing a query-document pair.

As noted earlier, the objective is to learn a scoring function that produces relevance scores for documents in any $\boldsymbol{x}$ in such a way that the utility of the ordered list is maximized. Formally, the scoring function $f(\cdot; \Theta) : \mathcal{X}^n \rightarrow \mathbb{R}^n$, where Θ is a set of parameters, is learned by minimizing the empirical loss:

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

where $\ell(\cdot)$ is a local loss function. The function f is often univariate and can be rewritten as $f(\boldsymbol{x})|_i = u(x_i)$, $1 \leq i \leq n$, where $f(\cdot)|_i$ denotes the i^{th} dimension of f, and $u : \mathcal{X} \rightarrow \mathbb{R}$ computes a relevance score for each item independently of other items.

LTR algorithms differ primarily in how they parameterize f and how they define ℓ. Tried and tested parameterization methods include linear functions [10], boosted weak learners [23], gradient-boosted trees [2, 8], support vector machines [10], and neural networks [1]. For an extensive list of loss functions ℓ, we refer the reader to Section 1.

2.2 Ranking Metrics

Ranking utilities such as NDCG or MRR are designed to emphasize the top of the ranked list in a way that mimics user behavior; the contribution to the utility from a document in the ranked list fades as its rank increases, following the behavior of a typical user who is less likely to examine documents at larger rank positions. Take NDCG as an example. It is defined as follows:

$$NDCG(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) = \frac{DCG(\pi_{f(\boldsymbol{x})}, \boldsymbol{y})}{DCG(\pi^*_{\boldsymbol{x}}, \boldsymbol{y})}, \tag{2}$$

where $\pi_{f(\boldsymbol{x})}$ is a ranked list over $\boldsymbol{x}$ induced by f, $\pi^*_{\boldsymbol{x}}$ is the ideal ranked list (where items are sorted by $\boldsymbol{y}$), and DCG is defined as follows:

$$DCG(\pi, \boldsymbol{y}) = \sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log_2(1 + \pi(i))}, \tag{3}$$

where $\pi(i)$ is the rank of $x_i \in \boldsymbol{x}$. Note that, NDCG computes a score in the closed real interval $[0, 1]$.

Reciprocal Rank is only considers the position of the first relevant document and is generally appropriate for evaluations on click logs. The metric is computed as follows:

$$\text{RR}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) = \max_{\{i \mid y_i > 0\}} \frac{1}{\pi_{f(\boldsymbol{x})}(i)}. \tag{4}$$

In practice, given an evaluation set $Q = \{(\boldsymbol{x}, \boldsymbol{y})\}$ and a metric $\mathcal{M}$, we are interested in the mean metric:

$$\overline{\mathcal{M}}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) = \frac{1}{|Q|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in Q} \mathcal{M}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}). \tag{5}$$

This results in mean NDCG and mean RR (or MRR) which we denote by $\overline{\text{NDCG}}$ and $\overline{\text{RR}}$ respectively.

2.3 Softmax Cross Entropy

Finally, we detail the construction of the ListNet [4] loss function. To compute the loss, $\boldsymbol{y}$ and $f(\boldsymbol{x})$ are projected onto the probability simplex to form the ground-truth distribution $P_{\boldsymbol{y}}$ and the score distribution P_f as follows:

$$P_{\boldsymbol{y}}(x_i) = \frac{y_i}{\sum_{j=1}^{n} y_j}, \quad P_f(x_i) = \frac{e^{f(\boldsymbol{x})|_i}}{\sum_{j=1}^{n} e^{f(\boldsymbol{x})|_j}}. \tag{6}$$

These probabilities 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.

Given the two distributions, the loss is their distance as measured by cross entropy:

$$\ell(\boldsymbol{y}, f(\boldsymbol{x})) = H(P_{\boldsymbol{y}}, P_f) \triangleq -\sum_{i=1}^{n} P_{\boldsymbol{y}}(x_i) \log P_f(x_i). \tag{7}$$

Finally, inserting this loss into Equation (1) gives the softmax cross entropy empirical loss.

3 ANALYSIS

In this section, we begin by showing a connection between the softmax cross entropy empirical loss and MRR when only a single document is relevant. We then extend the proof to MRR on queries with arbitrary number of relevant documents. Finally, we establish that the loss bounds NDCG as well.

In each proof, we make use of the application of Jensen's inequality to the log function. Given the concavity of log, the inequality takes the following form:

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

where X is a random variable and $\mathbb{E}[\cdot]$ denotes expectation.

Lastly, to simplify notation and arguments, we assume all queries have at least a single relevant document. As queries with no relevant documents do not contribute to the loss function, excluding them in our analysis does not lead to loss of generality.

Theorem 1. *Consider* $\Psi = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^n \times \{0,1\}^n \mid \boldsymbol{y}^T\mathbf{1} = 1\}$ *(i.e., a single relevant document per query). Softmax cross entropy empirical loss is a bound on mean reciprocal rank in log-scale.*

Proof. Consider first a single example $(\boldsymbol{x}, \boldsymbol{y})$ and let r be the index of its relevant document ($y_r = 1$). Let $\pi_{f(\boldsymbol{x})}$ be a permutation produced by a ranking function f for $\boldsymbol{x}$, and denote the rank of x_i with $\pi_{f(\boldsymbol{x})}(i)$. Let $\mathbb{I}_p$ be the indicator function that takes the value 1 if the predicate p is true and 0 otherwise, and write f_k for $f(\boldsymbol{x})|_k$. We have the following:

$$\pi_{f(\boldsymbol{x})}(r) = 1 + \sum_{i \neq r} \mathbb{I}_{f_i > f_r} = 1 + \sum_{i \neq r} \mathbb{I}_{f_i - f_r > 0}$$
$$\leq 1 + \sum_{i \neq r} e^{f_i - f_r} = \sum_i e^{f_i - f_r} = \frac{\sum_i e^{f_i}}{e^{f_r}},$$

which leads to the following result:

$$\mathrm{RR}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) = \frac{1}{\pi_{f(\boldsymbol{x})}(r)} \geq \frac{e^{f_r}}{\sum_i e^{f_i}}. \tag{9}$$

Consider now $\log(\overline{\mathrm{RR}})$. By Equation (9) and Jensen's inequality:

$$\log\overline{\mathrm{RR}}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) = \log \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{\pi_{f(\boldsymbol{x})}(r)} \geq \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \log \frac{e^{f_r}}{\sum_i e^{f_i}}$$

Turning the problem above from maximization to minimization, it is clear that the last term is the mean of the softmax cross entropy loss. Given that maximizing $\overline{\mathrm{RR}}$ is equivalent to minimizing negative log of the same quantity—as $\log(\cdot)$ is monotonically increasing—the above completes the proof. □

Theorem 2. *Consider* $\Psi = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^n \times \{0,1\}^n\}$ *(i.e., multiple relevant documents per query). Softmax cross entropy empirical loss is a bound on mean reciprocal rank in log-scale.*

Proof. Define $I(\boldsymbol{x}) = \{i \mid y_i = 1\}$, the set of indices of relevant documents in $\boldsymbol{x}$. By definition $\sum_i y_i = |I(\boldsymbol{x})|$. Rewrite Equation (4) as follows:

$$\mathrm{RR}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) = \max_{r \in I(\boldsymbol{x})} \frac{1}{\pi_{f(\boldsymbol{x})}(r)} \geq \frac{1}{|I(\boldsymbol{x})|} \sum_{r \in I(\boldsymbol{x})} \frac{1}{\pi_{f(\boldsymbol{x})}(r)}. \tag{10}$$

As a result:

$$\overline{\mathrm{RR}}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) = \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \mathrm{RR}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y})$$
$$\geq \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{|I(\boldsymbol{x})|} \sum_{r \in I(\boldsymbol{x})} \frac{1}{\pi_{f(\boldsymbol{x})}(r)} \quad \text{(by Eq. (10))}$$
$$\geq \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{|I(\boldsymbol{x})|} \sum_{r \in I(\boldsymbol{x})} \frac{e^{f_r}}{\sum_i e^{f_i}}. \quad \text{(by Eq. (9))}$$

Taking the log of the above:

$$\log\overline{\mathrm{RR}}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) \geq \log \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{|I(\boldsymbol{x})|} \sum_{r \in I(\boldsymbol{x})} \frac{e^{f_r}}{\sum_i e^{f_i}}$$
$$\geq \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{|I(\boldsymbol{x})|} \sum_{r \in I(\boldsymbol{x})} \log \frac{e^{f_r}}{\sum_i e^{f_i}}. \quad \text{(Jensen's applied twice)}$$

Turning the maximization problem into a minimization by negating the objective function as in the proof of Theorem 1, it is clear that the last term is the mean of softmax cross entropy loss, thereby completing the proof. □

Theorem 3. *Consider* $\Psi = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^n \times \{0,1\}^n\}$ *(i.e., multiple relevant documents per query). Softmax cross entropy empirical loss is a bound on mean Normalized Discounted Cumulative Gain in log-scale.*

Proof. Similar to the proof of Theorem 2, define $I(\boldsymbol{x}) = \{i \mid y_i = 1\}$. Consider first $\mathrm{DCG}(\pi^*_{\boldsymbol{x}}, \boldsymbol{y})$ for a single example $(\boldsymbol{x}, \boldsymbol{y})$:

$$\mathrm{DCG}(\pi^*_{\boldsymbol{x}}, \boldsymbol{y}) = \sum_{i=1}^{|I(\boldsymbol{x})|} \frac{1}{\log_2(1+i)} \leq \sum_{i=1}^{|I(\boldsymbol{x})|} 1 \leq |I(\boldsymbol{x})|.$$

Clearly, then:

$$\frac{1}{\mathrm{DCG}(\pi^*, \boldsymbol{y})} \geq \frac{1}{|I(\boldsymbol{x})|}. \tag{11}$$

Turning to $\mathrm{DCG}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y})$ and using $1 + z \leq e^z$ or equivalently $\log(1+z) \leq z$, we have the following:

$$\mathrm{DCG}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) = \sum_{r \in I(\boldsymbol{x})} \frac{1}{\log_2(1 + \pi_{f(\boldsymbol{x})}(r))} \tag{12}$$
$$\geq \sum_{r \in I(\boldsymbol{x})} \frac{1}{\pi_{f(\boldsymbol{x})}(r)} \geq \sum_{r \in I(\boldsymbol{x})} \frac{e^{f_r}}{\sum_i e^{f_i}}, \tag{13}$$

where the last inequality holds by Equation (9).

Finally, consider $\log(\overline{\mathrm{NDCG}})$:

$$\log \overline{\mathrm{NDCG}} = \log \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{\mathrm{DCG}(\pi^*_{\boldsymbol{x}}, \boldsymbol{y})} \mathrm{DCG}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y})$$
$$\geq \log \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{|I(\boldsymbol{x})|} \mathrm{DCG}(\pi_{f(\boldsymbol{x})}, \boldsymbol{y}) \quad \text{(by Eq. (11))}$$
$$\geq \log \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{|I(\boldsymbol{x})|} \sum_{r \in I(\boldsymbol{x})} \frac{e^{f_r}}{\sum_i e^{f_i}} \quad \text{(by Eq. (13))}$$
$$\geq \frac{1}{|\Psi|} \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{1}{|I(\boldsymbol{x})|} \sum_{r \in I(\boldsymbol{x})} \log \frac{e^{f_r}}{\sum_i e^{f_i}}. \quad \text{(Jensen's)}$$

As before, negating the objective function to create a minimization problem completes the proof. □

4 DISCUSSION

The analysis in Section 3 suggests that optimizing the ListNet loss is an indirect attempt at optimizing MRR or NDCG when given binary relevance judgments—we note that the bound does not hold when labels are graded. We believe, in addition to the discussion in the original work [4], our analysis explains ListNet's behavior in comparison with other "pairwise" methods that focus on preference ordering. Moreover, our analysis sheds light on why ListNet was shown in [4] to outperform other methods on the TREC .GOV collection [6], which provides binary relevance judgments, by a

Table 1: Comparison of learning-to-rank models on the test set by mean NDCG (95% confidence intervals) over 5 trials. λMART's gain over ListNet is also reported.

Model	NDCG@5	NDCG@10
ListNet Web30K	46.72 (±0.08)	48.98 (±0.09)
λMART Web30K	49.20 (±0.07, ↑5.3%)	51.05 (±0.02, ↑4.2%)
ListNet Web30K-b	75.83 (±0.07)	73.89 (±0.08)
λMART Web30K-b	78.16 (±0.08, ↑3.1%)	75.60 (±0.09, ↑2.3%)
ListNet Yahoo!	72.90 (±0.05)	77.49 (±0.05)
λMART Yahoo!	74.16 (±0.14, ↑1.7%)	78.40 (±0.10, ↑1.2%)
ListNet Yahoo!-b	89.95 (±0.07)	90.80 (±0.05)
λMART Yahoo!-b	90.56 (±0.07, ↑0.7%)	91.21 (±0.07, ↑0.4%)

much wider margin than on other datasets with graded relevance labels.

In an attempt to verify the consistency of this behavior, we have conducted experiments of our own on two benchmark datasets: MSLR Web30K [15] Fold 1 and Yahoo! Learning-to-Rank Challenge [5] Set 1. Both contain roughly 30,000 queries. Web30K (Yahoo!) has 120 (24) documents per query on average, each represented by 136 (519) numeric features. Documents are labeled with graded relevance from 0 to 4 with larger labels indicating a higher relevance. It is important to note that in both datasets, queries with no relevant documents are discarded during evaluation. Finally, by setting all non-zero labels to 1 we construct two synthetic variants of these datasets with binary labels which we refer to as Web30K-b and Yahoo!-b.

We implemented the ListNet loss in LightGBM [12] and compare its performance with LambdaMART [3] (λMART). Hyperparameters are tuned on validation sets and are configured as follows: For Web30K (Yahoo!), learning rate is 0.1, num_leaves is 255 (200), min_data_in_leaf is 50 (100), and min_sum_hessian_in_leaf is 100 (10). Hyperparameters do not change for Web30K-b but for Yahoo!-b, min_sum_hessian_in_leaf is set to 400. We use NDCG@5 on validation sets to select the best models and allow up to 500 trees with early stopping rounds set to 30.

Table 1 summarizes the results. While λMART consistently and statistically significantly outperforms ListNet as anticipated, the gap between ListNet and λMART is narrower on the datasets with binary labels. This finding is in agreement with the analysis in Section 3 as well as the results reported in [4].

5 CONCLUSION AND FUTURE WORK

In this work, we provided proof that ListNet's loss function is a bound on MRR and NDCG in the binary relevance regime. Our analysis helps explain ListNet's behavior and sheds light on its superior performance on binary labeled data as presented in the original publication as well as experiments in this work.

We conclude by noting that the form of the softmax cross entropy loss is more friendly than λMART to incorporating inverse propensity weights [19] for countering position bias. Together with the fact that it can be computed very efficiently, these factors make the loss an appropriate choice for learning from click logs. We wish to explore these directions in future work.

6 ACKNOWLEDGEMENTS

This work would not be possible without the support provided by the TF-Ranking team. We thank Donald Metzler for his input on an early draft of this work, and the anonymous reviewers for their feedback. The first author's deepest gratitude goes to Katherine for her invaluable encouragement and wholehearted support.

REFERENCES

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proc. of the 22nd International Conference on Machine Learning*. 89–96.
[2] Christopher J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report MSR-TR-2010-82. Microsoft Research.
[3] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Proc. of the 19th International Conference on Neural Information Processing Systems*. 193–200.
[4] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proc. of the 24th International Conference on Machine Learning*. 129–136.
[5] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proc. of the Learning to Rank Challenge*. 1–24.
[6] Nick Craswell, David Hawking, Ross Wilkinson, and Mingfang wu. 2003. Overview of the TREC 2003 web track. 78–92.
[7] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 65–74.
[8] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* 29, 5 (2001), 1189–1232.
[9] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.
[10] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 217–226.
[11] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proc. of the 10th ACM International Conference on Web Search and Data Mining*. 781–789.
[12] 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.
[13] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.
[14] 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.
[15] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. arXiv:1306.2597
[16] 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.
[17] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, De-Sheng Wang, and Hang Li. 2008. Global ranking using continuous conditional random fields. In *Proc. of the 21st International Conference on Neural Information Processing Systems*. 1281–1288.
[18] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. SoftRank: Optimizing Non-smooth Rank Metrics. In *Proc. of the 1st International Conference on Web Search and Data Mining*. 77–86.
[19] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *Proc. of the 11th International Conference on Web Search and Data Mining*. 610 –618.
[20] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *Proc. of the 27th ACM International Conference on Information and Knowledge Management*. 1313–1322.
[21] 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.
[22] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proc. of the 25th International Conference on Machine Learning*. 1192–1199.
[23] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 391–398.