UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Optimizing Ranking Models in an Online Setting

Oosterhuis, H.; de Rijke, M.

DOI
10.1007/978-3-030-15712-8_25
Publication date
2019
Document Version
Author accepted manuscript
Published in
Advances in Information Retrieval

Link to publication

Citation for published version (APA):
Oosterhuis, H., & de Rijke, M. (2019). Optimizing Ranking Models in an Online Setting. In L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, & D. Hiemstra (Eds.), *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019 : proceedings* (Vol. 1, pp. 382-396). (Lecture Notes in Computer Science; Vol. 11437). Springer. https://doi.org/10.1007/978-3-030-15712-8_25

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:13 10月 2026

arXiv:1901.10262v1 [cs.IR] 29 Jan 2019

Optimizing Ranking Models in an Online Setting

Harrie Oosterhuis and Maarten de Rijke

University of Amsterdam, Amsterdam, The Netherlands
{oosterhuis, derijke}@uva.nl

Abstract. Online Learning to Rank (OLTR) methods optimize ranking models by directly interacting with users, which allows them to be very efficient and responsive. All OLTR methods introduced during the past decade have extended on the original OLTR method: Dueling Bandit Gradient Descent (DBGD). Recently, a fundamentally different approach was introduced with the Pairwise Differentiable Gradient Descent (PDGD) algorithm. To date the only comparisons of the two approaches are limited to simulations with cascading click models and low levels of noise. The main outcome so far is that PDGD converges at higher levels of performance and learns considerably faster than DBGD-based methods. However, the PDGD algorithm assumes cascading user behavior, potentially giving it an unfair advantage. Furthermore, the robustness of both methods to high levels of noise has not been investigated. Therefore, it is unclear whether the reported advantages of PDGD over DBGD generalize to different experimental conditions. In this paper, we investigate whether the previous conclusions about the PDGD and DBGD comparison generalize from ideal to worst-case circumstances. We do so in two ways. First, we compare the theoretical properties of PDGD and DBGD, by taking a critical look at previously proven properties in the context of ranking. Second, we estimate an upper and lower bound on the performance of methods by simulating both *ideal* user behavior and extremely *difficult* behavior, i.e., almost-random non-cascading user models. Our findings show that the theoretical bounds of DBGD do not apply to any common ranking model and, furthermore, that the performance of DBGD is substantially worse than PDGD in both ideal and worst-case circumstances. These results reproduce previously published findings about the relative performance of PDGD vs. DBGD and generalize them to extremely noisy and non-cascading circumstances.

Keywords: Learning to rank · Online learning · Gradient descent

1 Introduction

Learning to Rank (LTR) plays a vital role in information retrieval. It allows us to optimize models that combine hundreds of signals to produce rankings, thereby making large collections of documents accessible to users through effective search and recommendation. Traditionally, LTR has been approached as a supervised learning problem, where annotated datasets provide human judgements indicating relevance. Over the years, many limitations of such datasets have become apparent: they are costly to produce [3,21] and actual users often disagree with the relevance annotations [23]. As an alternative, research into LTR approaches that learn from user behavior has increased.

By learning from the implicit feedback in user behavior, users' true preferences can potentially be learned. However, such methods must deal with the noise and biases that are abundant in user interactions [31]. Roughly speaking, there are two approaches to LTR from user interactions: learning from historical interactions and Online Learning to Rank (OLTR). Learning from historical data allows for optimization without gathering new data [14], though it does require good models of the biases in logged user interactions [4]. In contrast, OLTR methods learn by interacting with the user, thus they gather their own learning data. As a result, these methods can adapt instantly and are potentially much more responsive than methods that use historical data.

Dueling Bandit Gradient Descent (DBGD) [30] is the most prevalent OLTR method; it has served as the basis of the field for the past decade. DBGD samples variants of its ranking model, and compares them using interleaving to find improvements [12,22]. Subsequent work in OLTR has extended on this approach [10,25,28]. Recently, the first alternative approach to DBGD was introduced with *Pairwise Differentiable Gradient Descent* (PDGD) [19]. PDGD estimates a pairwise gradient that is reweighed to be unbiased w.r.t. users' document pair preferences. The original paper that introduced PDGD showed considerable improvements over DBGD under simulated user behavior [19]: a substantially higher point of performance at convergence and a much faster learning speed. The results in [19] are based on simulations using low-noise cascading click models. The pairwise assumption that PDGD makes, namely, that all documents preceding a clicked document were observed by the user, is always correct in these circumstances, thus potentially giving it an unfair advantage over DBGD. Furthermore, the low level of noise presents a close-to-ideal situation, and it is unclear whether the findings in [19] generalize to less perfect circumstances.

In this paper, we contrast PDGD over DBGD. Prior to an experimental comparison, we determine whether there is a theoretical advantage of DBGD over PDGD and examine the regret bounds of DBGD for ranking problems. We then investigate whether the benefits of PDGD over DBGD reported in [19] generalize to circumstances ranging from ideal to worst-case. We simulate circumstances that are perfect for both methods – behavior without noise or position-bias – and circumstances that are the worst possible scenario – almost-random, extremely-biased, non-cascading behavior. These settings provide estimates of upper and lower bounds on performance, and indicate how well previous comparisons generalize to different circumstances. Additionally, we introduce a version of DBGD that is provided with an oracle interleaving method; its performance shows us the maximum performance DBGD could reach from hypothetical extensions.

In summary, the following research questions are addressed in this paper:

RQ1 Do the regret bounds of DBGD provide a benefit over PDGD?
RQ2 Do the advantages of PDGD over DBGD observed in prior work generalize to extreme levels of noise and bias?
RQ3 Is the performance of PDGD reproducible under non-cascading user behavior?

2 Related Work

This section provides a brief overview of traditional LTR (Section 2.1), of LTR from historical interactions (Section 2.2), and OLTR (Section 2.3).

2.1 Learning to rank from annotated datasets

Traditionally, LTR has been approached as a supervised problem; in the context of OLTR this approach is often referred to as *offline* LTR. It requires a dataset containing relevance annotations of query-document pairs, after which a variety of methods can be applied [16]. The limitations of offline LTR mainly come from obtaining such annotations. The costs of gathering annotations are high as it is both time-consuming and expensive [3,21]. Furthermore, annotators cannot judge for very specific users, i.e., gathering data for personalization problems is infeasible. Moreover, for certain applications it would be unethical to annotate items, e.g., for search in personal emails or documents [29]. Additionally, annotations are stationary and cannot account for (perceived) relevance changes [6,15,27]. Most importantly, though, annotations are not necessarily aligned with user preferences; judges often interpret queries differently from actual users [23]. As a result, there has been a shift of interest towards LTR approaches that do not require annotated data.

2.2 Learning to rank from historical interactions

The idea of LTR from user interactions is long-established; one of the earliest examples is the original pairwise LTR approach [13]. This approach uses historical click-through interactions from a search engine and considers clicks as indications of relevance. Though very influential and quite effective, this approach ignores the *noise* and *biases* inherent in user interactions. Noise, i.e., any user interaction that does not reflect the user's true preference, occurs frequently, since many clicks happen for unexpected reasons [23]. Biases are systematic forms of noise that occur due to factors other than relevance. For instance, interactions will only involve displayed documents resulting in selection bias [29]. Another important form of bias in LTR is position bias, which occurs because users are less likely to consider documents that are ranked lower [31]. Thus, to learn true preferences from user interactions effectively, a LTR method should be robust to noise and handle biases correctly.

In recent years counter-factual LTR methods have been introduced that correct for some of the bias in user interactions. Such methods uses inverse propensity scoring to account for the probability that a user observed a ranking position [14]. Thus, clicks on positions that are observed less often due to position bias will have greater weight to account for that difference. However, the position bias must be learned and estimated somewhat accurately [1]. On the other side of the spectrum are click models, which attempt to model user behavior completely [4]. By predicting behavior accurately, the effect of relevance on user behavior can also be estimated [2,29].

An advantage of these approaches over OLTR is that they only require historical data and thus no new data has to be gathered. However, unlike OLTR, they do require a fairly accurate user model, and thus they cannot be applied in cold-start situations.

2.3 Online learning to rank

OLTR differs from the approaches listed above because its methods intervene in the search experience. They have control over what results are displayed, and can learn

from their interactions instantly. Thus, the online approach performs LTR by interacting with users directly [30]. Similar to LTR methods that learn from historical interaction data, OLTR methods have the potential to learn the true user preferences. However, they also have to deal with the noise and biases that come with user interactions. Another advantage of OLTR is that the methods are very responsive, as they can apply their learned behavior instantly. Conversely, this also brings a danger as an online method that learns incorrect preferences can also worsen the experience immediately. Thus, it is important that OLTR methods are able to learn reliably in spite of noise and biases. Thus, OLTR methods have a two-fold task: they have to simultaneously present rankings that provide a good user experience *and* learn from user interactions with the presented rankings.

The original OLTR method is Dueling Bandit Gradient Descent (DBGD); it approaches optimization as a dueling bandit problem [30]. This approach requires an online comparison method that can compare two rankers w.r.t. user preferences; traditionally, DBGD methods use interleaving. Interleaving methods take the rankings produced by two rankers and combine them in a single result list, which is then displayed to users. From a large number of clicks on the presented list the interleaving methods can reliably infer a preference between the two rankers [12,22]. At each timestep, DBGD samples a candidate model, i.e., a slight variation of its current model, and compares the current and candidate models using interleaving. If a preference for the candidate is inferred, the current model is updated towards the candidate slightly. By doing so, DBGD will update its model continuously and should oscillate towards an inferred optimum. Section 3 provides a complete description of the DBGD algorithm.

Virtually all work in OLTR in the decade since the introduction of DBGD has used DBGD as a basis. A straightforward extension comes in the form of Multileave Gradient Descent [25] which compares a large number of candidates per interaction [18,24,26]. This leads to a much faster learning process, though in the long term this method does not seem to improve the point of convergence.

One of the earliest extensions of DBGD proposed a method for reusing historical interactions to guide exploration for faster learning [10]. While the initial results showed great improvements [10], later work showed performance drastically decreasing in the long term due to bias introduced by the historical data [20]. Unfortunately, OLTR work that continued this historical approach [28] also only considered short term results; moreover, the results of some work [32] are not based on held-out data. As a result, we do not know whether these extensions provide decent long-term performance and it is unclear whether the findings of these studies generalize to more realistic settings.

Recently, an inherently different approach to OLTR was introduced with PDGD [19]. PDGD interprets its ranking model as a distribution over documents; it estimates a pairwise gradient from user interactions with sampled rankings. This gradient is differentiable, allowing for non-linear models like neural networks to be optimized, something DBGD is ineffective at [17,19]. Section 4 provides a detailed description of PDGD. In the paper in which we introduced PDGD, claim that it provides substantial improvements over DBGD. However, those claims are based on cascading click models with low levels of noise. This is problematic because PDGD assumes a cascading user, and could thus have an unfair advantage in this setting. Furthermore, it is unclear

Algorithm 1 Dueling Bandit Gradient Descent (DBGD).

1: **Input**: initial weights: θ_1; unit: u; learning rate η.
2: **for** $t \leftarrow 1 \ldots \infty$ **do**
3: $q_t \leftarrow receive_query(t)$ *obtain a query from a user*
4: $\theta_t^c \leftarrow \theta_t + sample_from_unit_sphere(u)$ *create candidate ranker*
5: $R_t \leftarrow get_ranking(\theta_t, D_{q_t})$ *get current ranker ranking*
6: $R_t^c \leftarrow get_ranking(\theta_t^c, D_{q_t})$ *get candidate ranker ranking*
7: $I_t \leftarrow interleave(R_t, R_t^c)$ *interleave both rankings*
8: $\mathbf{c}_t \leftarrow display_to_user(I_t)$ *displayed interleaved list, record clicks*
9: **if** $preference_for_candidate(I_t, \mathbf{c}_t, R_t, R_t^c)$ **then**
10: $\theta_{t+1} \leftarrow \theta_t + \eta(\theta_t^c - \theta_t)$ *update model towards candidate*
11: **else**
12: $\theta_{t+1} \leftarrow \theta_t$ *no update*

whether DBGD with a perfect interleaving method could still improve over PDGD. Lastly, DBGD has proven regret bounds while PDGD has no such guarantees.

In this study, we clear up these questions about the relative strengths of DBGD and PDGD by comparing the two methods under non-cascading, high-noise click models. Additionally, by providing DBGD with an oracle comparison method, its hypothetical maximum performance can be measured; thus, we can study whether an improvement over PDGD is hypothetically possible. Finally, a brief analysis of the theoretical regret bounds of DBGD shows that they do not apply to any common ranking model, therefore hardly providing a guaranteed advantage over PDGD.

3 Dueling Bandit Gradient Descent

This section describes the DBGD algorithm in detail, before discussing the regret bounds of the algorithm.

3.1 The Dueling Bandit Gradient Descent method

The DBGD algorithm [30] describes an indefinite loop that aims to improve a ranking model at each step; Algorithm 1 provides a formal description. The algorithm starts a given model with weights θ_1 (Line 1); then it waits for a user-submitted query (Line 3). At this point a candidate ranker is sampled from the unit sphere around the current model (Line 4), and the current and candidate model both produce a ranking for the current query (Line 5 and 6). These rankings are interleaved (Line 7) and displayed to the user (Line 8). If the interleaving method infers a preference for the candidate ranker from subsequent user interactions the current model is updated towards the candidate (Line 10), otherwise no update is performed (Line 12). Thus, the model optimized by DBGD should converge and oscillate towards an optimum.

3.2 Regret bounds of Dueling Bandit Gradient Descent

Unlike PDGD, DBGD has proven regret bounds [30], potentially providing an advantage in the form of theoretical guarantees. In this section we answer **RQ1** by critically looking at the assumptions which form the basis of DBGD's proven regret bounds.

The original DBGD paper [30] proved a sublinear regret under several assumptions. DBGD works with the parameterized space of ranking functions $\mathcal{W}$, that is, every $\theta \in \mathcal{W}$ is a different set of parameters for a ranking function. For this study we will only consider linear models because all existing OLTR work has dealt with them [10,11,19,20,25,28,30,32]. But we note that the proof is easily extendable to neural networks where the output is a monotonic function applied to a linear combination of the last layer. Then there is assumed to be a concave utility function $u : \mathcal{W} \rightarrow \mathbb{R}$; since this function is concave, there should only be a single instance of weights that are optimal θ^*. Furthermore, this utility function is assumed to be L-Lipschitz smooth:

$$\exists L \in \mathbb{R}, \quad \forall (\theta_a, \theta_b) \in \mathcal{W}, \quad |u(\theta_a) - u(\theta_b)| < L\|\theta_a - \theta_b\|. \tag{1}$$

We will show that these assumptions are *incorrect*: there is an infinite number of optimal weights, and the utility function u cannot be L-Lipschitz smooth. Our proof relies on two assumptions that avoid cases where the ranking problem is trivial. First, the zero ranker is not the optimal model:

$$\theta^* \neq \mathbf{0}. \tag{2}$$

Second, there should be at least two models with different utility values:

$$\exists (\theta, \theta') \in \mathcal{W}, \quad u(\theta) \neq u(\theta'). \tag{3}$$

We will start by defining the set of rankings a model $f(\cdot, \theta)$ will produce as:

$$\mathcal{R}_D(f(\cdot, \theta)) = \{R \mid \forall (d, d') \in D, [f(d, \theta) > f(d', \theta) \rightarrow d \succ_R d']\}. \tag{4}$$

It is easy to see that multiplying a model with a positive scalar $\alpha > 0$ will not affect this set:

$$\forall \alpha \in \mathbb{R}_{>0}, \quad \mathcal{R}_D(f(\cdot, \theta)) = \mathcal{R}_D(\alpha f(\cdot, \theta)). \tag{5}$$

Consequently, the utility of both functions will be equal:

$$\forall \alpha \in \mathbb{R}_{>0}, \quad u(f(\cdot, \theta)) = u(\alpha f(\cdot, \theta)). \tag{6}$$

For linear models scaling weights has the same effect: $\alpha f(\cdot, \theta) = f(\cdot, \alpha\theta)$. Thus, the first assumption cannot be true since for any optimal model $f(\cdot, \theta^*)$ there is an infinite set of equally optimal models: $\{f(\cdot, \alpha\theta^*) \mid \alpha \in \mathbb{R}_{>0}\}$.

Then, regarding L-Lipschitz smoothness, using any positive scaling factor:

$$\forall \alpha \in \mathbb{R}_{>0}, \quad |u(\theta_a) - u(\theta_b)| = |u(\alpha\theta_a) - u(\alpha\theta_b)|, \tag{7}$$

$$\forall \alpha \in \mathbb{R}_{>0}, \quad \|\alpha\theta_a - \alpha\theta_b\| = \alpha\|\theta_a - \theta_b\|. \tag{8}$$

Thus the smoothness assumption can be rewritten as:

$$\exists L \in \mathbb{R}, \quad \forall \alpha \in \mathbb{R}_{>0}, \quad \forall (\theta_a, \theta_b) \in \mathcal{W}, \quad |u(\theta_a) - u(\theta_b)| < \alpha L\|\theta_a - \theta_b\|. \tag{9}$$

However, there is always an infinite number of values for α small enough to break the assumption. Therefore, we conclude that a concave L-Lipschitz smooth utility function

can never exist for a linear ranking model, thus the proof for the regret bounds is not applicable when using linear models.

Consequently, the regret bounds of DBGD do not apply to the ranking problems in previous work. One may consider other models (e.g., spherical coordinate based models), however this still means that for the simplest and most common ranking problems there are no proven regret bounds. As a result, we answer **RQ1** negatively, the regret bounds of DBGD do not provide a benefit over PDGD for the ranking problems in LTR.

4 Pairwise Differentiable Gradient Descent

The Pairwise Differentiable Gradient Descent (PDGD) [19] algorithm is formally described in Algorithm 2. PDGD interprets a ranking function $f(\cdot, \theta)$ as a probability distribution over documents by applying a Plackett-Luce model:

$$P(d|D, \theta) = \frac{e^{f(d,\theta)}}{\sum_{d' \in D} e^{f(d',\theta)}}. \tag{10}$$

First, the algorithm waits for a user query (Line 3), then a ranking R is created by sampling documents without replacement (Line 4). Then PDGD observes clicks from the user and infers pairwise document preferences from them. All documents preceding a clicked document and the first succeeding one are assumed to be observed by the user. Preferences between clicked and unclicked observed documents are inferred by PDGD; this is a long-standing assumption in pairwise LTR [13]. We denote an *inferred* preference between documents as $d_i \succ_{\mathbf{c}} d_j$, and the probability of the model placing d_i earlier than d_j is denoted and calculated by:

$$P(d_i \succ d_j \mid \theta) = \frac{e^{f(d_i,\theta)}}{e^{f(d_i,\theta)} + e^{f(d_j,\theta)}}. \tag{11}$$

The gradient is estimated as a sum over inferred preferences with a weight ρ per pair:

$$\begin{aligned} &\Delta f(\cdot, \theta) \\ &\quad \approx \sum_{d_i \succ_{\mathbf{c}} d_j} \rho(d_i, d_j, R, D)[\Delta P(d_i \succ d_j \mid \theta)] \\ &\quad = \sum_{d_i \succ_{\mathbf{c}} d_j} \rho(d_i, d_j, R, D) P(d_i \succ d_j \mid \theta) P(d_j \succ d_i \mid \theta)(f'(d_i, \theta) - f'(d_j, \theta)). \end{aligned} \tag{12}$$

After computing the gradient (Line 10), the model is updated accordingly (Line 11). This will change the distribution (Equation 10) towards the inferred preferences. This distribution models the confidence over which documents should be placed first; the exploration of PDGD is naturally guided by this confidence and can vary per query.

The weighting function ρ is used to make the gradient of PDGD unbiased w.r.t. document pair preferences. It uses the reverse pair ranking: $R^*(d_i, d_j, R)$, which is the same ranking as R but with the document positions of d_i and d_j swapped. Then ρ is the ratio between the probability of R and R^*:

$$\rho(d_i, d_j, R, D) = \frac{P(R^*(d_i, d_j, R) \mid D)}{P(R \mid D) + P(R^*(d_i, d_j, R) \mid D)}. \tag{13}$$

Algorithm 2 Pairwise Differentiable Gradient Descent (PDGD).

Input: initial weights: θ_1; scoring function: f; learning rate η.
for $t \leftarrow 1 \dots \infty$ **do**
  $q_t \leftarrow receive_query(t)$ *// obtain a query from a user*
  $\mathbf{R}_t \leftarrow sample_list(f_{\theta_t}, D_{q_t})$ *// sample list according to Eq. 10*
  $\mathbf{c}_t \leftarrow receive_clicks(\mathbf{R}_t)$ *// show result list to the user*
  $\nabla f(\cdot, \theta_t) \leftarrow \mathbf{0}$ *// initialize gradient*
  for $d_i \succ_\mathbf{c} d_j \in \mathbf{c}_t$ **do**
    $w \leftarrow \rho(d_i, d_j, R, D)$ *// initialize pair weight (Eq. 13)*
    $w \leftarrow w \times P(d_i \succ d_j \mid \theta_t) P(d_j \succ d_i \mid \theta_t)$ *// pair gradient (Eq. 12)*
    $\nabla f(\cdot, \theta_t) \leftarrow \nabla f_{\theta_t} + w \times (f'(d_i, \theta_t) - f'(d_j, \theta_t))$ *// model gradient (Eq. 12)*
  $\theta_{t+1} \leftarrow \theta_t + \eta \nabla f(\cdot, \theta_t)$ *// update the ranking model*

In the original PDGD paper [19], the weighted gradient is proven to be unbiased w.r.t. document pair preferences under certain assumptions about the user. Here, this unbiasedness is defined by being able to rewrite the gradient as:

$$E[\Delta f(\cdot, \theta)] = \sum_{(d_i, d_j) \in D} \alpha_{ij} (f'(\mathbf{d}_i, \theta) - f'(\mathbf{d}_j, \theta)), \tag{14}$$

and the sign of α_{ij} agreeing with the preference of the user:

$$sign(\alpha_{ij}) = sign(relevance(d_i) - relevance(d_j)). \tag{15}$$

The proof in [19] only relies on the difference in the probabilities of inferring a preference: $d_i \succ_\mathbf{c} d_j$ in R and the opposite preference $d_j \succ_\mathbf{c} d_i$ in $R^*(d_i, d_j, R)$. The proof relies on the sign of this difference to match the user's preference:

$$\begin{aligned} sign(P(d_i \succ_\mathbf{c} d_j \mid R) - P(d_j \succ_\mathbf{c} d_i \mid R^*)) = \\ sign(relevance(d_i) - relevance(d_j)). \end{aligned} \tag{16}$$

As long as Equation 16 is true, Equation 14 and 15 hold as well. Interestingly, this means that other assumptions about the user can be made than in [19], and other variations of PDGD are possible, e.g., the algorithm could assume that all documents are observed and the proof still holds.

The original paper on PDGD reports large improvements over DBGD, however these improvements were observed under simulated cascading user models. This means that the assumption that PDGD makes about which documents are observed are always true. As a result, it is currently unclear whether the method is really better in cases where the assumption does not hold.

5 Experiments

In this section we detail the experiments that were performed to answer the research questions in Section 1.[1]

[1] The resources for reproducing the experiments in this paper are available at `https://github.com/HarrieO/OnlineLearningToRank`

Table 1. Click probabilities for simulated *perfect* or *almost random* behavior.

	$P(click(d) \mid relevance(d), observed(d))$				
$relevance(d)$	0	1	2	3	4
perfect	0.00	0.20	0.40	0.80	1.00
almost random	0.40	0.45	0.50	0.55	0.60

5.1 Datasets

Our experiments are performed over three large labelled datasets from commercial search engines, the largest publicly available LTR datasets. These datasets are the *MLSR-WEB10K* [21], *Yahoo! Webscope* [3], and *Istella* [5] datasets. Each contains a set of queries with corresponding preselected document sets. Query-document pairs are represented by feature vectors and five-grade relevance annotations ranging from *not relevant* (0) to *perfectly relevant* (4). Together, the datasets contain over 29,900 queries and between 136 and 700 features per representation.

5.2 Simulating user behavior

In order to simulate user behavior we partly follow the standard setup for OLTR [8,11,20,25,33]. At each step a user issued query is simulated by uniformly sampling from the datasets. The algorithm then decides what result list to display to the user, the result list is limited to $k = 10$ documents. Then user interactions are simulated using click models [4]. Past OLTR work has only considered *cascading click models* [7]; in contrast, we also use *non-cascading click models*. The probability of a click is conditioned on relevance and observance:

$$P(click(d) \mid relevance(d), observed(d)). \tag{17}$$

We use two levels of noise to simulate *perfect* user behavior and *almost random* behavior [9], Table 1 lists the probabilities of both. The *perfect* user observes all documents, never clicks on anything non-relevant, and always clicks on the most relevant documents. Two variants of *almost random* behavior are used. The first is based on cascading behavior, here the user first observes the top document, then decides to click according to Table 1. If a click occurs, then, with probability $P(stop \mid click) = 0.5$ the user stops looking at more documents, otherwise the process continues on the next document. The second *almost random* behavior is simulated in a non-cascading way; here we follow [14] and model the observing probabilities as:

$$P(observed(d) \mid rank(d)) = \frac{1}{rank(d)}. \tag{18}$$

The important distinction is that it is safe to assume that the cascading user has observed all documents ranked before a click, while this is not necessarily true for the non-cascading user. Since PDGD makes this assumption, testing under both models can show us how much of its performance relies on this assumption. Furthermore, the *almost random* model has an extreme level of noise and position bias compared to the click models used in previous OLTR work [11,20,25], and we argue it simulates an (almost) worst-case scenario.

5.3 Experimental runs

In our experiments we simulate runs consisting of 1,000,000 impressions; each run was repeated 125 times under each of the three click models. PDGD was run with $\eta = 0.1$ and zero initialization, DBGD was run using Probabilistic Interleaving [20] with zero initialization, $\eta = 0.001$, and the unit sphere with $\delta = 1$. Other variants like Multi-leave Gradient Descent [25] are not included; previous work has shown that their performance matches that of regular DBGD after around 30,000 impressions [19,20,25]. The initial boost in performance comes at a large computational cost, though, as the fastest approaches keep track of at least 50 ranking models [20], which makes running long experiments extremely impractical. Instead, we introduce a novel oracle version of DBGD, where, instead of interleaving, the NDCG values on the current query are calculated and the highest scoring model is selected. This simulates a hypothetical perfect interleaving method, and we argue that the performance of this oracle run indicates what the upper bound on DBGD performance is.

Performance is measured by NDCG@10 on a held-out test set, a two-sided t-test is performed for significance testing. We do not consider the user experience during training, because past work has already investigated this aspect thoroughly [19].

6 Experimental Results and Analysis

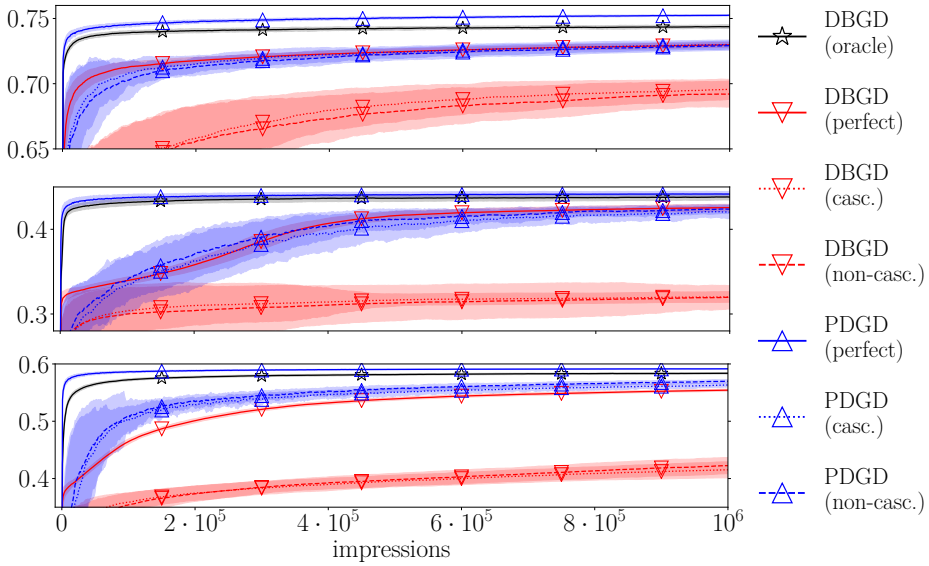
Recall that in Section 3.2 we have already provided a negative answer to **RQ1**: the regret bounds of DBGD do not provide a benefit over PDGD for the ranking problems in LTR. In this section we present our experimental results and answer **RQ2** (whether the advantages of PDGD over DBGD of previous work generalize to extreme levels of noise and bias) and **RQ3** (whether the performance of PDGD is reproducible under non-cascading user behavior).

Our main results are presented in Table 2. Additionally, Figure 1 displays the average performance over 1,000,000 impressions. First, we consider the performance of DBGD; there is a substantial difference between its performance under the *perfect* and *almost random* user models on all datasets. Thus, it seems that DBGD is strongly affected by noise and bias in interactions; interestingly, there is little difference between performance under the cascading and non-cascading behavior. On all datasets the *oracle* version of DBGD performs significantly better than DBGD under *perfect* user behavior. This means there is still room for improvement and hypothetical improvements in, e.g., interleaving could lead to significant increases in long-term DBGD performance.

Next, we look at the performance of PDGD; here, there is also a significant difference between performance under the *perfect* and *almost random* user models on all datasets. However, the effect of noise and bias is very limited compared to DBGD, and this difference at 1,000,000 impressions is always less than 0.03 NDCG on any dataset.

To answer **RQ2**, we compare the performance of DBGD and PDGD. Across all datasets, when comparing DBGD and PDGD under the same levels of interaction noise and bias, the performance of PDGD is significantly better in every case. Furthermore, PDGD under the *perfect* user model significantly outperforms the *oracle* run of DBGD, despite the latter being able to directly observe the NDCG of rankers on the current query. Moreover, when comparing PDGD performance under the *almost random* user

Fig. 1. Performance (NDCG@10) on held-out data from Yahoo (top), MSLR (center), Istella (bottom) datasets, under the *perfect*, and *almost random* user models: cascading (casc.) and non-cascading (non-casc.). The shaded areas display the standard deviation.



model with DBGD under the *perfect* user model, we see the differences are limited and in both directions. Thus, even under ideal circumstances DBGD does not consistently outperform PDGD under extremely difficult circumstances. As a result, we answer **RQ2** positively: our results strongly indicate that the performance of PDGD is considerably better than DBGD and that these findings generalize from ideal circumstances to settings with extreme levels of noise and bias.

Finally, to answer **RQ3**, we look at the performance under the two *almost random* user models. Surprisingly, there is no clear difference between the performance of PDGD under *cascading* and *non-cascading* user behavior. The differences are small and per dataset it differs which circumstances are slightly preferred. Therefore, we answer **RQ3** positively: the performance of PDGD is reproducible under *non-cascading* user behavior.

7 Conclusion

In this study, we have reproduced and generalized findings about the relative performance of Dueling Bandit Gradient Descent (DBGD) and Pairwise Differentiable Gradient Descent (PDGD). Our results show that the performance of PDGD is reproducible under non-cascading user behavior. Furthermore, PDGD outperforms DBGD in both *ideal* and extremely *difficult* circumstances with high levels of noise and bias. Moreover, the performance of PDGD in extremely *difficult* circumstances is comparable to that of DBGD in *ideal* circumstances. Additionally, we have shown that the regret bounds of DBGD are not applicable to the ranking problem in LTR. In summary, our results

Table 2. Performance (NDCG@10) after 1,000,000 impressions for DBGD and PDGD under a *perfect* click model and two almost-random click models: *cascading* and *non-cascading*, and DBGD with an *oracle* comparator. Significant improvements and losses ($p < 0.01$) between DBGD and PDGD are indicated by ▲, ▼, and ○ (no significant difference). Indications are in order of: *oracle*, *perfect*, *cascading*, and *non-cascading*.

	Yahoo	**MSLR**	**Istella**
	Dueling Bandit Gradient Descent		
oracle	0.744 (0.001) ▼ ▲ ▲	0.438 (0.004) ▼ ▲ ▲	0.584 (0.001) ▼ ▲ ▲
perfect	0.730 (0.002) ▼ ○ ○	0.426 (0.004) ▼ ▲ ▲	0.554 (0.002) ▼ ▼ ▼
cascading	0.696 (0.008) ▼ ▼ ▼	0.320 (0.006) ▼ ▼ ▼	0.415 (0.014) ▼ ▼ ▼
non-cascading	0.692 (0.010) ▼ ▼ ▼	0.320 (0.014) ▼ ▼ ▼	0.422 (0.014) ▼ ▼ ▼
	Pairwise Differentiable Gradient Descent		
perfect	0.752 (0.001) ▲ ▲ ▲ ▲	0.442 (0.003) ▲ ▲ ▲ ▲	0.592 (0.000) ▲ ▲ ▲ ▲
cascading	0.730 (0.003) ▼ ○ ▲ ▲	0.420 (0.007) ▼ ▼ ▲ ▲	0.563 (0.003) ▼ ▲ ▲ ▲
non-cascading	0.729 (0.003) ▼ ○ ▲ ▲	0.424 (0.005) ▼ ▼ ▲ ▲	0.570 (0.003) ▼ ▲ ▲ ▲

strongly confirm the previous finding that PDGD consistently outperforms DBGD, and generalizes this conclusion to circumstances with extreme levels of noise and bias.

Consequently, there appears to be no advantage to using DBGD over PDGD in either theoretical or empirical terms. In addition, a decade of OLTR work has attempted to extend DBGD in numerous ways without leading to any measurable long-term improvements. Together, this suggests that the general approach of DBGD based methods, i.e., sampling models and comparing with online evaluation, is not an optimally effective way of optimizing ranking models. Although the PDGD method considerably outperforms the DBGD approach, we currently do not have a theoretical explanation for this difference. Thus it seems plausible that a more effective OLTR method could be derived, if the theory behind the effectiveness of OLTR methods is better understood. Due to this potential and the current lack of regret bounds applicable to OLTR, we argue that a theoretical analysis of OLTR could make a very valuable future contribution to the field.

Finally, we consider the limitations of the comparison in this study. As is standard in OLTR our results are based on simulated user behavior. These simulations provide valuable insights: they enable direct control over biases and noise, and evaluation can be performed at each time step. In this paper, the generalizability of this setup was pushed the furthest by varying the conditions to the extremely difficult. It appears unlikely that more reliable conclusions can be reached from simulated behavior. Thus we argue that the most valuable future comparisons would be in experimental settings with real users. Furthermore, with the performance improvements of PDGD the time seems right for evaluating the effectiveness of OLTR in real-world applications.

Acknowledgements. This research was supported by Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), the Innovation Center for Artificial Intelligence (ICAI), and the Netherlands Organization for Scientific Research (NWO) under project nr 612.001.551. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

1. Ai, Q., Bi, K., Luo, C., Guo, J., Croft, W.B.: Unbiased learning to rank with unbiased propensity estimation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 385–394. ACM (2018)
2. Borisov, A., Markov, I., de Rijke, M., Serdyukov, P.: A neural click model for web search. In: WWW. pp. 531–541. International World Wide Web Conferences Steering Committee (2016)
3. Chapelle, O., Chang, Y.: Yahoo! Learning to Rank Challenge Overview. Journal of Machine Learning Research **14**, 1–24 (2011)
4. Chuklin, A., Markov, I., de Rijke, M.: Click Models for Web Search. Morgan & Claypool Publishers (2015)
5. Dato, D., Lucchese, C., Nardini, F.M., Orlando, S., Perego, R., Tonellotto, N., Venturini, R.: Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. ACM Transactions on Information Systems (TOIS) **35**(2), Article 15 (2016)
6. Dumais, S.T.: The web changes everything: Understanding and supporting people in dynamic information environments. In: ECDL. pp. 1–1. Springer (2010)
7. Guo, F., Liu, C., Wang, Y.M.: Efficient multiple-click models in web search. In: WSDM. pp. 124–131. ACM (2009)
8. He, J., Zhai, C., Li, X.: Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In: CIKM. pp. 2029–2032. ACM (2009)
9. Hofmann, K.: Fast and Reliable Online Learning to Rank for Information Retrieval. Ph.D. thesis, University of Amsterdam (2013)
10. Hofmann, K., Schuth, A., Whiteson, S., de Rijke, M.: Reusing historical interaction data for faster online learning to rank for IR. In: WSDM. pp. 183–192. ACM (2013)
11. Hofmann, K., Whiteson, S., de Rijke, M.: Balancing exploration and exploitation in learning to rank online. In: ECIR. pp. 251–263. Springer (2011)
12. Hofmann, K., Whiteson, S., de Rijke, M.: A probabilistic method for inferring preferences from clicks. In: CIKM. pp. 249–258. ACM (2011)
13. Joachims, T.: Optimizing search engines using clickthrough data. In: KDD. pp. 133–142. ACM (2002)
14. Joachims, T., Swaminathan, A., Schnabel, T.: Unbiased learning-to-rank with biased feedback. In: WSDM. pp. 781–789. ACM (2017)
15. Lefortier, D., Serdyukov, P., de Rijke, M.: Online exploration for detecting shifts in fresh intent. In: CIKM. pp. 589–598. ACM (November 2014)
16. Liu, T.Y.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval **3**(3), 225–331 (2009)
17. Oosterhuis, H., de Rijke, M.: Balancing speed and quality in online learning to rank for information retrieval. In: CIKM. pp. 277–286. ACM (2017)
18. Oosterhuis, H., de Rijke, M.: Sensitive and scalable online evaluation with theoretical guarantees. In: CIKM. pp. 77–86. ACM (2017)
19. Oosterhuis, H., de Rijke, M.: Differentiable unbiased online learning to rank. In: CIKM. pp. 1293–1302. ACM (2018)
20. Oosterhuis, H., Schuth, A., de Rijke, M.: Probabilistic multileave gradient descent. In: ECIR. pp. 661–668. Springer (2016)
21. Qin, T., Liu, T.Y.: Introducing letor 4.0 datasets. arXiv preprint arXiv:1306.2597 (2013)
22. Radlinski, F., Craswell, N.: Optimized interleaving for online retrieval evaluation. In: WSDM. pp. 245–254. ACM (2013)
23. Sanderson, M.: Test collection based evaluation of information retrieval systems. Foundations and Trends in Information Retrieval **4**(4), 247–375 (2010)

24. Schuth, A., Bruintjes, R.J., Büttner, F., van Doorn, J., Groenland, C., Oosterhuis, H., Tran, C.N., Veeling, B., van der Velde, J., Wechsler, R., Woudenberg, D., de Rijke, M.: Probabilistic multileave for online retrieval evaluation. In: SIGIR. pp. 955–958. ACM (2015)
25. Schuth, A., Oosterhuis, H., Whiteson, S., de Rijke, M.: Multileave gradient descent for fast online learning to rank. In: WSDM. pp. 457–466. ACM (2016)
26. Schuth, A., Sietsma, F., Whiteson, S., Lefortier, D., de Rijke, M.: Multileaved comparisons for fast online evaluation. In: CIKM. pp. 71–80. ACM (2014)
27. Vakkari, P., Hakala, N.: Changes in relevance criteria and problem stages in task performance. Journal of Documentation **56**, 540–562 (2000)
28. Wang, H., Langley, R., Kim, S., McCord-Snook, E., Wang, H.: Efficient exploration of gradient space for online learning to rank. In: SIGIR. pp. 145–154. ACM (2018)
29. Wang, X., Bendersky, M., Metzler, D., Najork, M.: Learning to rank with selection bias in personal search. In: SIGIR. pp. 115–124. ACM (2016)
30. Yue, Y., Joachims, T.: Interactively optimizing information retrieval systems as a dueling bandits problem. In: ICML. pp. 1201–1208. ACM (2009)
31. Yue, Y., Patel, R., Roehrig, H.: Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In: WWW. pp. 1011–1018. ACM (2010)
32. Zhao, T., King, I.: Constructing reliable gradient exploration for online learning to rank. In: CIKM. pp. 1643–1652. ACM (2016)
33. Zoghi, M., Whiteson, S., de Rijke, M., Munos, R.: Relative confidence sampling for efficient on-line ranker evaluation. In: WSDM. pp. 73–82. ACM (2014)