Probabilistic Multileave Gradient Descent

Harrie Oosterhuis, Anne Schuth, and Maarten de Rijke
harrie.oosterhuis@student.uva.nl,
{a.g.schuth,derijke}@uva.nl

University of Amsterdam, Amsterdam, The Netherlands

Abstract. Online learning to rank methods aim to optimize ranking models based on user interactions. The dueling bandit gradient descent (DBGD) algorithm is able to effectively optimize linear ranking models solely from user interactions. We propose an extension of DBGD, called probabilistic multileave gradient descent (P-MGD) that builds on probabilistic multileave, a recently proposed highly sensitive and unbiased online evaluation method. We demonstrate that P-MGD significantly outperforms state-of-the-art online learning to rank methods in terms of online performance, without sacrificing offline performance and at greater learning speed.

1 Introduction

Modern search engines are complex aggregates of multiple ranking signals. Such aggregates are learned using learning to rank methods. Online learning to rank methods learn from user interactions such as clicks [4, 6, 10, 12]. Dueling Bandit Gradient Descent [16, DBGD] uses interleaved comparison methods [1, 3, 6, 7, 10] to infer preferences and then learns by following a gradient that is meant to lead to an optimal ranker.

We introduce probabilistic multileave gradient descent (P-MGD), an online learning to rank method that builds on a recently proposed highly sensitive and unbiased online evaluation method, viz. probabilistic multileave. Multileave comparisons allow one to compare multiple but a still limited set of candidate rankers per user interaction [13]. The more recently introduced probabilistic multileave comparison method improves over this by allowing for comparisons of an unlimited number of rankers at a time [15]. We show experimentally that P-MGD significantly outperforms state-of-the-art online learning to rank methods in terms of online performance, without sacrificing offline performance and at greater learning speed than those methods. In particular, we include comparisons between P-MGD on the one hand and multiple types of DBGD and multileaved gradient descent methods [14, MGD] and candidate preselection [5, CPS] on the other. We answer the following research questions: (RQ1) Does P-MGD convergence on a ranker of the same quality as MGD and CPS? (RQ2) Does P-MGD require less queries to converge compared to MGD and CPS? (RQ3) Is the user experience during the learning process of P-MGD better than during that of MGD or CPS?

2 Probabilistic multileaving

Multileaving [13] is an online evaluation approach for inferring preferences between rankers from user clicks. Multileave methods take a set of rankers and when a query is submitted a ranking is computed for each of the rankers. These rankings are then combined into a single multileaved ranking. Team Draft Multileaving (TDM) assigns

each document in this resulting ranking to a ranker. The user is then presented with this multileaved ranking and his interactions are recorded. TDM keeps track of the clicks and attributes every clicked document to the ranker to which it was assigned. Two important aspects of online evaluation methods are *sensitivity* and *bias*. TDM is more sensitive than existing interleaving methods [3, 9, 11], since it requires fewer user interactions to infer preferences. Secondly, empirical evaluation also showed that TDM has no significant bias [13]. Probabilistic multileave [15, PM] extends probabilistic interleave [3, PI]. Unlike TDM, PM selects documents from a distribution where the probability of being added correlates with the perceived relevance. It marginalizes over all possible team assignments, which makes it more sensitive and allows it to infer preferences within a virtually unlimited set of rankers from a single interaction. The increased sensitivity of PM and its lack of bias were confirmed empirically [15]. Our novel contribution is that we use PM instead of TDM for inferring preferences in our online learning to rank method, allowing the learner to explore a virtually unlimited set of rankers.

3 Online learning to rank methods

Besides detailing learning to rank baselines, we introduce P-MGD, a variant of MGD.

Dueling Bandit Gradient Descent (DBGD) [16] uses an interleaving method (e.g. Team Draft Interleaving [10]) to infer a *relative* feedback signal: at each interaction with a user the algorithm uses interleaving to infer a preference between its current best ranker and a candidate ranker. If a preference for the candidate is inferred from the interaction DBGD updates the ranker accordingly. With $n = 1$, Algorithm 1 boils down to DBGD.

Multileave Gradient Descent (MGD) [14] is an extension to DBGD that infers preferences with a larger group of candidate rankers using multileaving, as described above. This allows the algorithm to learn and converge faster. MGD is outlined in Algorithm 1. The number of candidates compared at each iteration is set by the parameter n. MGD represents rankers by weight vectors. The ranker that MGD currently considers best is referred to as the current best ranker, initially $\mathbf{w}_0^0$, and is updated according to the user interactions, For each query issue, n candidate rankers are sampled from the unit sphere around the current best ranker. These candidate rankers and the current best ranker create rankings of documents that are subsequently multileaved and the resulting list is shown to the user. Clicks from the user are then interpreted by the multileaving method to infer a preference among the candidates. MGD allows multiple candidates to be preferred over the current best; we consider the Mean-Winner update approach [14] as it is the most robust; it updates the current best towards the mean of all preferred candidates. The algorithm repeats this for every incoming query, yielding an unending adaptive process.

Probabilistic Multileave Gradient Descent (P-MGD) is introduced in this paper. The novelty of this method comes from the usage of PM instead of TDM as its multileaving method. TDM needs to assign each document to a team in order to infer preferences. This limits the number of rankers that are compared at each interaction to the number of displayed documents. PM on the other hand allows for a virtually unlimited number of rankers to be compared. The advantage of P-MGD is that it can learn faster by having n, the number of candidates, in Algorithm 1 exceed the length of the result list.

Candidate Preselection (CPS) [5], unlike MGD, does not alter the number of candidates compared per impression. It speeds up learning by reusing historical data to select

Algorithm 1 Multileave Gradient Descent: MGD(n, α, δ, $\mathbf{w}_0^0$)

for $q_t, t \leftarrow 0..\infty$ **do**
  $\mathbf{l}^0 \leftarrow generate\_list(\mathbf{w}_t^0, q_t)$ *// ranking of current best*
  for $i \leftarrow 1...n$ **do**
    $\mathbf{u}^i \leftarrow sample\_unit\_vector()$
    $\mathbf{w}_t^i \leftarrow \mathbf{w}_t^0 + \delta\mathbf{u}^i$ *// create a candidate ranker*
    $\mathbf{l}^i \leftarrow generate\_list(\mathbf{w}_t^i, q_t)$ *// exploratory ranking*
  $\mathbf{m}_t, \mathbf{t}_t \leftarrow multileave(\mathbf{l})$ *// multileaving and teams*
  $\mathbf{b}_t \leftarrow infer\_winners(\mathbf{t}_t, receive\_clicks(\mathbf{m}_t))$ *// set of winning candidates*
  $\mathbf{w}_{t+1}^0 \leftarrow \mathbf{w}_t^0 + \alpha \frac{1}{|\mathbf{b}_t|} \sum_{j \in \mathbf{b}_t} \mathbf{u}^j$ *// update, note that $\mathbf{b}_t$ could be empty*

more promising candidates for DBGD. A set of candidates is generated by repeatedly sampling the unit sphere around the current best uniformly. Several rounds are simulated to eliminate all but one candidate. Each round starts by sampling two candidates between which a preference is inferred with Probabilistic Interleave [3, PI]. The least preferred of the two candidates is discarded; if no preference is found, one is discarded at random. The remaining candidate is then used by DBGD.

4 Experimental setup

We describe our experiments, designed to answer the research questions posed in §1.[1] An experiment is based on a stream of independent queries submitted by users interacting with the system that is being trained. A result list of ten documents is displayed in response to each query. Users interact with the list by clicking on zero or more documents. The queries are sampled from several static datasets, clicks are simulated using click models. Four learning to rank datasets [8] were selected to cover a diverse set of tasks: named page finding (*NP2003*, 150 queries), topic distillation (*TD2003*, 50 queries), medical search (*OHSUMED*, 106 queries), and general web search (*MSLR-WEB10K*, 10K queries). Each dataset consists of a set of feature vectors of length 45–136, encoding query document relations, and manual relevance assessments for each document with respect to queries.

Table 1. Overview of instantiations of CCM [2].

	$P(click=1\|R)$					$P(stop=1\|R)$				
R	0	1	2	3	4	0	1	2	3	4
per	0.0	0.2	0.4	0.8	1.0	0.0	0.0	0.0	0.0	0.0
nav	0.05	0.3	0.5	0.7	0.95	0.2	0.3	0.5	0.7	0.9
inf	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5

To simulate user interactions, we follow [14]. Clicks are produced by a *cascade click model* (CCM) [2]. Users are assumed to examine documents from top to bottom and click with probability $P(click = 1|R)$, conditioned on relevance grade R. The user then stops with probability $P(stop = 1|R)$. Table 1 lists the instantiations of CCM: a *perfect* (per) user with very reliable feedback, a *navigational* (nav) user looking for a singly highly relevant document, and an *informational* (inf) user whose interactions are noisier. Runs consist of 10,000 queries, exceeding the 1,000 queries used in previous work. Performance is evaluated offline and online. Offline NDCG is measured on held out data and represents the quality of the trained ranker. Online performance reflects the user experience during training and is measured by the discounted cumulative NDCG of the result lists shown to the user. A discount factor $\gamma = 0.9995$ was chosen so

[1] Our experimental code is available at https://bitbucket.org/ilps/lerot.

Table 2. Offline score (NDCG) after 10,000 query impressions of each of the algorithms for the 3 instantiations of the CCM (see Table 1). Bold values indicate maximum performance per dataset and click model. Statistically significant improvements (losses) over the DBGD and TD-MGD baseline are indicated by $^{\triangle}(p < 0.05)$ and $^{\blacktriangle}(p < 0.01)$ ($^{\triangledown}$and $^{\blacktriangledown}$). Standard deviation in brackets.

		PI-DBGD	TD-MGD-9c	CPS	P-MGD-9c	P-MGD-99c
perfect	**TD2003**	0.330 (0.07)	0.327 (0.07)	0.288 (0.10) $^{\blacktriangledown}$ $^{\blacktriangledown}$	0.330 (0.07)	**0.334 (0.07)**
	OHSUMED	0.452 (0.06)	0.443 (0.06)	0.395 (0.06) $^{\blacktriangledown}$ $^{\blacktriangledown}$	0.459 (0.07)	**0.459 (0.07)**
	NP2003	0.718 (0.08)	0.727 (0.08)	0.702 (0.09) $^{\triangledown}$	0.730 (0.08)	**0.732 (0.08)**
	MSLR-WEB10K	0.318 (0.03)	**0.351 (0.03)**	0.246 (0.03) $^{\blacktriangledown}$ $^{\blacktriangledown}$	0.325 (0.03) $^{\blacktriangledown}$	0.320 (0.03) $^{\blacktriangledown}$
navigational	**TD2003**	0.319 (0.08)	0.322 (0.07)	0.271 (0.10) $^{\blacktriangledown}$ $^{\blacktriangledown}$	0.324 (0.08)	**0.332 (0.07)**
	OHSUMED	0.436 (0.06)	**0.446 (0.06)**	0.389 (0.07) $^{\blacktriangledown}$ $^{\blacktriangledown}$	0.435 (0.06)	0.431 (0.06) $^{\triangledown}$
	NP2003	0.706 (0.08)	0.721 (0.08)	0.702 (0.08)	0.719 (0.08)	**0.723 (0.08)**
	MSLR-WEB10K	0.305 (0.03)	**0.320 (0.03)**	0.238 (0.03) $^{\blacktriangledown}$ $^{\blacktriangledown}$	0.309 (0.03) $^{\blacktriangledown}$	0.308 (0.03) $^{\blacktriangledown}$
informational	**TD2003**	0.288 (0.09)	**0.323 (0.08)**	0.233 (0.10) $^{\blacktriangledown}$ $^{\blacktriangledown}$	0.310 (0.09)	0.313 (0.10) $^{\triangle}$
	OHSUMED	0.421 (0.06)	0.440 (0.06)	0.389 (0.07) $^{\blacktriangledown}$ $^{\blacktriangledown}$	**0.441 (0.07)** $^{\triangle}$	0.434 (0.07)
	NP2003	0.679 (0.08)	0.706 (0.08)	0.665 (0.09) $^{\blacktriangledown}$	0.704 (0.08) $^{\triangle}$	**0.707 (0.08)** $^{\blacktriangle}$
	MSLR-WEB10K	0.283 (0.03)	**0.311 (0.03)**	0.213 (0.03) $^{\blacktriangledown}$ $^{\blacktriangledown}$	0.303 (0.03) $^{\blacktriangle}$ $^{\triangledown}$	0.306 (0.03) $^{\blacktriangle}$

that queries beyond a horizon of 10,000 impressions have $< 1\%$ impact. Two tailed Student's t-tests are used for significance testing. All experiments ran 125 times (5 folds, 25 repetitions), results are averaged. Parameters come from previous work: $\mathbf{w}_0 = \mathbf{0}$, $\alpha = 0.01$, $\delta = 1$. We use two baselines: (PI-DBGD) – DBGD with PI, PI was chosen for a fair comparison with PM methods; and (TD-MGD-9c) – MGD with TDM, the number of candidate rankers is $n = 9$ so that each ranker is represented exactly once in the result list. Furthermore, two additional algorithms are compared to the baselines: (P-MGD-n) – MGD with PM, this algorithm is run with $n = 9$ and $n = 99$ candidates; the former matches the number of candidates with our TD-MGD baseline, the latter exploits the large number of candidates that PM enables; and CPS is run with the settings reported as best in [3]: $\eta = 6$ candidates, $\zeta = 10$ rounds, history length $\lambda = 10$; we use the unbiased version and discard historic interactions without clicks.

5 Results and analysis

To investigate where P-MGD converges compared to TD-MGD and CPS (RQ1), we consider offline NDCG (Table 2). P-MGD significantly outperforms PI-DBGD on all runs. Compared to TD-MGD, P-MGD performs significantly worse on some datasets; no significant difference can be found for the majority of runs. The number of candidates in P-MGD does not seem to affect the offline performance strongly. Surprisingly, the offline performance of CPS is significantly worse than TD-MGD and DBGD on all runs except for four instances. Fig. 1 shows that the offline performance of CPS drops after an initial peak. CPS seems to overfit because of the effect of historical data on candidate sampling. The other methods sample candidates uniformly, thus noisy false preferences are expected in all directions evenly; therefore, over time they will oscillate in the right direction. Conversely, CPS samples more candidates in the directions that historical data expects the best candidates to be, causing the method not to oscillate but drift due to noise. The increased sensitivity of CPS does not compensate for its bias in the long run.

To answer how the learning speed of P-MGD compares to our baselines (RQ2) we consider Fig. 1, which shows offline performance on the *NP-2003* dataset with the

Table 3. Online score (NDCG) after 10,000 query impressions of each of the algorithms for the 3 instantiations of the CCM (see Table 1). Notation is the same as that of Table 2.

		PI-DBGD	TD-MGD-9c	CPS	P-MGD-9c	P-MGD-99c
perfect	**TD2003**	499.1 (34.9)	557.0 (32.4)	503.7 (42.6) ▼	541.7 (32.3) ▲▼	**563.8** (**30.8**) ▲
	OHSUMED	780.3 (28.0)	781.4 (21.5)	764.7 (31.5) ▼▼	799.4 (25.6) ▲▲	**819.5** (**23.8**) ▲▲
	NP2003	1128.8 (27.9)	**1244.9** (**31.6**)	1222.8 (33.7) ▲▼	1169.3 (25.6) ▲▼	1191.9 (24.9) ▲▼
	MSLR-WEB10K	532.4 (18.5)	548.6 (9.4)	480.8 (41.5) ▼▼	548.3 (9.5) ▲	**560.5** (**6.9**) ▲▲
navigational	**TD2003**	453.5 (55.1)	**514.9** (**33.9**)	469.7 (61.0) △▼	480.6 (52.4) ▲▼	499.9 (54.7) ▲▽
	OHSUMED	747.4 (73.6)	**790.2** (**27.3**)	739.0 (72.4) ▼	776.1 (75.5) ▲	785.3 (75.6) ▲
	NP2003	1055.5 (103.4)	1173.0 (32.3)	**1206.7** (**110.9**) ▲▲	1085.0 (101.4) △▼	1114.0 (102.9) ▲▼
	MSLR-WEB10K	505.6 (22.9)	523.2 (12.3)	472.5 (54.9) ▼▼	523.2 (47.5) ▲	**538.8** (**7.9**) ▲▲
informational	**TD2003**	340.2 (76.2)	432.5 (53.2)	443.2 (42.8) ▲	430.1 (37.1) ▲	**490.4** (**36.0**) ▲▲
	OHSUMED	703.7 (49.8)	749.4 (73.4)	741.1 (31.6) ▲	764.5 (29.3) ▲△	**789.6** (**24.7**) ▲▲
	NP2003	849.0 (150.0)	1023.7 (101.6)	**1221.6** (**27.5**) ▲▲	996.0 (47.4) ▲▼	1119.5 (28.5) ▲▲
	MSLR-WEB10K	464.8 (56.0)	495.0 (46.8)	446.1 (46.6) ▼▼	514.3 (16.9) ▲▲	**536.7** (**48.6**) ▲▲

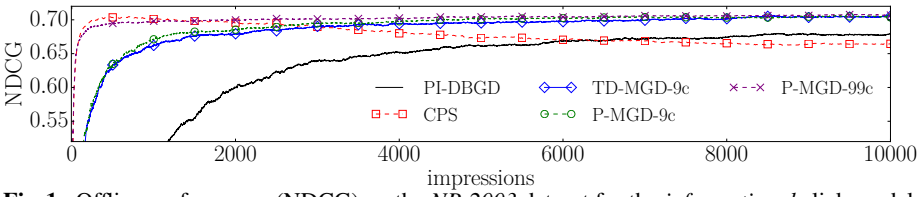

Fig. 1. Offline performance (NDCG) on the *NP-2003* dataset for the *informational* click model.

informational click model. P-MGD with 99 candidates and CPS perform substantially better than TD-MGD and DBGD during the first 1,000 queries and it takes around 2,000 queries before TD-MGD to reach a similar level of performance. From the substantial difference between P-MGD with 9 and 99 candidates, also present in the other runs, we conclude that P-MGD with a large number of candidates has a greater learning speed.

To answer RQ3 we evaluate the user experience during learning. Table 3 displays the results of our online experiments. In all runs the online performance of P-MGD significantly improves over DBGD, again showing the positive effect of increasing the number of candidates. Compared to TD-MGD, P-MGD performs significantly better under the *informational* click model. We conclude that P-MGD is a definite improvement over TD-MGD when clicks contain a large amount of noise. We attribute this difference to the greater learning speed of P-MGD: fewer queries are required to find rankers of the same performance as TD-MGD. Consequently, the rankings shown to users are better during the learning process. When comparing CPS to TD-MGD we see no significant improvements except on the *informational* and *navigational* runs on the *NP-2003* dataset. This is surprising as CPS was introduced as an alternative to DBGD that improves the user experience. Thus, P-MGD is a better alternative of TD-MGD especially when clicks are noisy; CPS does not offer reliable benefits when compared to TD-MGD.

6 Conclusions

We have introduced an extension of multileave gradient descent (MGD) that uses a recently introduced multileaving method, probabilistic multileaving. Our extension, *probabilistic multileave gradient descent* (P-MGD) marginalizes over document assignments in multileaved rankings. P-MGD has an increased sensitivity as it can infer

preferences over a large number of assignments. P-MGD can be run with a virtually unlimited number of candidates. We have compared P-MGD with dueling bandit gradient descent (DBGD), team-draft multileave gradient descent (TD-MGD), and candidate preselection (CPS), both offline and online. CPS overfits in terms of offline performance, due to bias introduced by the reuse of historical data. Online results for CPS did not show a convincing benefit over TD-MGD either. In contrast, P-MGD significantly improves over DBGD and TD-MGD in terms of online performance under noisy click models, without a significant decrease in offline performance. Moreover, P-MGD shows a greater learning speed than TD-MGD and DBGD, which becomes more evident as click model noise increases. Thus, P-MGD is a robust alternative for TD-MGD that is better able to deal with interaction noise.

Acknowledgements. This research was supported by Amsterdam Data Science, the Dutch national program COMMIT, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the ESF Research Network Program ELIAS, the Royal Dutch Academy of Sciences (KNAW) under the Elite Network Shifts project, the Microsoft Research Ph.D. program, the Netherlands eScience Center under project number 027.012.105, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, 612.066.930, CI-14-25, SH-322-15, 652.002.001, the Yahoo Faculty Research and Engagement Program, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

7 References

[1] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Trans. Inf. Syst.*, 30(1), 2012.
[2] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *WSDM '09*. ACM, 2009.
[3] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *CIKM '11*. ACM, 2011.
[4] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Inf. Retr.*, 16(1), 2012.
[5] K. Hofmann, A. Schuth, S. Whiteson, and M. de Rijke. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM '13*. ACM, 2013.
[6] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*. ACM, 2002.
[7] T. Joachims. Evaluating retrieval performance using clickthrough data. In *Text Mining*. Physica/Springer, 2003.
[8] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR '07*, 2007.
[9] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *WSDM '13*. ACM, 2013.
[10] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*. ACM, 2008.
[11] F. Radlinski, P. N. Bennett, and E. Yilmaz. Detecting duplicate web documents using clickthrough data. In *WSDM '11*. ACM, 2011.
[12] M. Sanderson. Test collection based evaluation of information retrieval systems. *Found. & Tr. Inform. Retr.*, 4(4):247–375, 2010.
[13] A. Schuth, F. Sietsma, S. Whiteson, D. Lefortier, and M. de Rijke. Multileaved comparisons for fast online evaluation. In *CIKM '14*, pages 71–80. ACM, Nov. 2014.
[14] A. Schuth, H. Oosterhuis, S. Whiteson, and M. de Rijke. Multileave gradient descent for fast online learning to rank. In *WSDM '16*. ACM, February 2016.
[15] A. Schuth et al. Probabilistic multileave for online retrieval evaluation. In *SIGIR '15*, 2015.
[16] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML '09*, 2009.