Counterfactual Online Learning to Rank

Shengyao Zhuang(✉) and Guido Zuccon

The University of Queensland, St Lucia, Australia
{s.zhuang,g.zuccon}@uq.edu.au

Abstract. Exploiting users' implicit feedback, such as clicks, to learn rankers is attractive as it does not require editorial labelling effort, and adapts to users' changing preferences, among other benefits. However, directly learning a ranker from implicit data is challenging, as users' implicit feedback usually contains bias (e.g., position bias, selection bias) and noise (e.g., clicking on irrelevant but attractive snippets, adversarial clicks). Two main methods have arisen for optimizing rankers based on implicit feedback: counterfactual learning to rank (CLTR), which learns a ranker from the historical click-through data collected from a deployed, logging ranker; and online learning to rank (OLTR), where a ranker is updated by recording user interaction with a result list produced by multiple rankers (usually via interleaving).

In this paper, we propose a counterfactual online learning to rank algorithm (COLTR) that combines the key components of both CLTR and OLTR. It does so by replacing the online evaluation required by traditional OLTR methods with the counterfactual evaluation common in CLTR. Compared to traditional OLTR approaches based on interleaving, COLTR can evaluate a large number of candidate rankers in a more efficient manner. Our empirical results show that COLTR significantly outperforms traditional OLTR methods. Furthermore, COLTR can reach the same effectiveness of the current state-of-the-art, under noisy click settings, and has room for future extensions.

1 Introduction

Traditional learning to rank (LTR) requires labelled data to permit the learning of a ranker: that is, a training dataset with relevance assessments for every query-document pair is required. The acquisition of such labelled datasets presents a number of drawbacks: they are expensive to construct [5,25], there may be ethical issues in privacy-sensitive tasks like email search [37], and they cannot capture changes in user's preferences [19].

The reliance on users implicit feedbacks such as clicks is an attractive alternative to the construction of editorially labelled datasets, as this data does not present the aforementioned limitations [15]. However, this does not come without its own drawbacks and challenges. User implicit feedback cannot be directly treated as (pure) relevance labels because it presents a number of biases, and part of this implicit user signal may actually be noise. For example, in web

J. M. Jose et al. (Eds.): ECIR 2020, LNCS 12035, pp. 415–430, 2020.
https://doi.org/10.1007/978-3-030-45439-5_28

search, users often examine the search engine result page (SERP) from top to bottom. Thus, higher ranked documents have a higher probability to be examined, attracting more clicks (position bias), which in turn may infer these results as relevant even when they are not [7,18,24]. Other types of biases may affect this implicit feedback including selection and presentation bias [2,16,40]. In addition, clicks on SERP items may be due to noise, e.g., sometimes users may click for unexpected reasons (e.g., clickbaits and serendipity), and these noisy clicks may hurt the learnt ranker. Hence, in order to leverage the benefits of implicit feedback, LTR algorithms have to be robust to these biases and noises. There are two main categories of approaches to learning a ranker from implicit feedback [14]:

(1) Offline LTR: Methods in this category learn a ranker using historical click-through log data collected from a production system (logging ranker). A representative method in this category is Counterfactual Learning to Rank (CLTR) [18], where a user's observation probability (known as propensity) is adopted to construct an unbiased estimator which is used as the objective function to train the ranker.

(2) Online LTR (OLTR): Methods in this category interactively optimize a ranker given the current user's interactions. A representative method in this category is Dueling Bandit Gradient Descent (DBGD) [39], where multiple rankers are used to produce an interleaved[1] results list to display to the user and collect clicks. This signal is used to unbiasedly indicate which rankers that participated in the interleaving process are better (Online Evaluation) and to trigger an update of the ranker in production.

The aim of the counterfactual and the online evaluations is similar: they both attempt to unbiasedly evaluate the effectiveness of a ranker and thus can provide LTR algorithms with reliable updating information.

In this paper, we introduce Counterfactual Online Learning to Rank (COLTR), the first online LTR algorithm that combines the key aspects of both CLTR and OLTR approaches to obtain an effective ranker that can learn online from user feedback. COLTR uses the DBGD framework from OLTR to interactively update the ranker used in production, but it uses the counterfactual evaluation mechanism of CLTR in place of online evaluation. The main challenge we address is that counterfactual evaluation cannot be directly used in online learning settings because the propensity model is unknown. This is resolved by mirroring solutions developed for learning in the bandit feedback problem (and specifically the Self-Normalized Estimator [34]) within the considered ranking task – this provides a position-unbiased evaluation of rankers. Our empirical results show that COTLR significantly improves the traditional DBGD baseline algorithm. In addition, because COTLR does not require interleaving or multileaving, which is the most computationally expensive part in online evaluation [28], COLTR is more efficient than DBGD. We also find that COLTR performance is at par with the current state-of-the-art OLTR method

[1] Two rankers: interleaving [12,26]; more than two rankers: multileaving[28,30].

[22] under noisy click settings, while presenting a number of avenues for further improvement.

2 Related Work

The goal of counterfactual learning to rank (CLTR) is to learn a ranker from historical user interaction logs obtained with the ranker used in production. An advantage of this approach is that candidate rankers are trained and evaluated offline, i.e., before being deployed in production, thus avoiding exposing users to rankers of lesser quality compared to that currently in production. However, unlike traditional supervised LTR methods [20], users interaction data provides only partial feedback which cannot be directly treated as absolute relevance labels [14,16]. This is because clicks may have not been observed on some results because of position or selection bias, and clicks may have instead been observed because of noise or errors. As a result, much of the prior work has focused on removing these biases and noise.

According to position bias, users are more likely to click on top-ranked search results than those at the bottom of the SERP [2,16,18]: in CLTR this probability is referred to as *propensity*. Joachims et al. [18] developed an unbiased (with respect to position) LTR that relies on clicks using a modified SVMRank approach that optimizes the empirical risk computed using the Inverse Propensity Scoring (IPS) estimator. The IPS is an unbiased estimator which can indicate the effectiveness of a ranker given propensity (the probability that the user will examine a document) and click data [18]. However, this approach requires a propensity model to compute the IPS score. To estimate this, randomization experiments are usually required when collecting the interaction data and the propensity model is estimated under offline setting [37,38].

Aside from position bias, selection bias is also important, and it dominates problems in other ranking tasks such as recommendation and ad placement. Selection bias refers to the fact that users can only interact with items presented to them. Typically, in ad placement systems, the assumption is made that users examine the displayed ads with certainty if only one item is shown: thus no position bias. However, users are given the chance to click on the displayed item only, so clicks are heavily biased due to selection. User interactions with this kind of systems are referred to as bandit feedback [17,33,34]. The Counterfactual Risk Minimization (CRM) learning principle [33] is used to remove the bias from bandit feedback. Instead of a deterministic ranker, this group of methods assume the system relies on the hypothesis that a probability distribution is available over the candidate items, which is used to sample items to show to users. Importance sampling [3] is commonly used to remove selection bias.

Online Learning to Rank aims to optimize the production ranker interactively by exploiting user clicks [10,22,23,29]. Unlike CLTR, OLTR algorithms do not require a propensity model to handle position or selection bias. Instead, they assume that relevant documents are more likely to receive more clicks than non-relevant documents and exploits clicks to identify the gradient's direction.

Dueling Bandit Gradient Descent (DBGD) based algorithms [39] are commonly used in OLTR. The traditional DBGD uses online evaluation to unbiasedly compare two or more rankers given a user interaction [12,29]. Subsequent methods developed more reliable or more efficient online evaluation methods, including Probabilistic Interleaving (PIGD) which has been proven to be unbiased [12]. The Probabilistic Multileaving extension (PMGD) [28], compares multiple rankers at each interaction, resulting in the best DBGD-based algorithm, which reaches a better convergence given less training impressions [23]. However, this method suffers from a high computational cost because it requires sampling ranking assignments to infer outcomes. Further variations that reuse historical interaction data to accelerate the learning in DBDG have also been investigated [10].

The current state-of-the-art OLTR algorithm is Pairwise Differentiable Gradient Descent (PDGD) [22], which does not require sampling candidate rankers to create interleaved results lists for online evaluation. Instead, PDGD creates a probability distribution over the document set and constructs the result list by sampling documents from this distribution. Then the gradients are estimated from pairwise documents preferences based on user clicks. This algorithm provides much better performance than traditional DBGD-based methods in terms of final convergence and user online experience.

3 Counterfactual Online Learning to Rank

3.1 Counterfactual Evaluation for Online Learning to Rank

The proposed COLTR method uses counterfactual evaluation to estimate the effectiveness of candidate rankers based on the click data collected by the logging ranker. This is unlike DBGD and other OLTR methods that use interleaving. In the counterfactual learning to rank setting, the IPS estimator is used to eliminate position bias [18], providing an unbiased estimation. However, the IPS estimator requires that the propensities of result documents are known. The propensity of a document is the probability that the user will examine the document. In offline LTR settings, propensities are estimated using offline click-through data, via a randomization experiment [38]. Offline click-through data is not available in the online setting we consider, and thus the use of IPS in such an online setting becomes a challenge. To overcome this, we adapt the counterfactual estimator used in batch learning from logged bandit feedback [32, 34]. This type of counterfactual learning treats rankers as policies and samples documents from a probability distribution to create the result list. This allows us to use importance sampling to fix the distribution mismatch between policies and to use Monte Carlo approximation to estimate the risk function $\mathcal{R}(f_{\theta'})$:

$$\mathcal{R}(f_{\theta'}) = \frac{1}{k} \sum_{i=1}^{k} \delta_i \frac{p(d_i|f_{\theta'}, D)}{p(d_i|f_{\theta}, D)} \tag{1}$$

Where k is the number of documents in the result list, θ is the feature weights of the logging ranker, $\theta^{'}$ is the new ranker's feature weights which need to be estimated, and δ is the reward function. Following the Counterfactual Risk Minimization (CRM) learning principle [33], we set:

$$\delta_i = \begin{cases} 0, & \text{if the user clicked or did not examine } d_i \\ 1, & \text{if the user examined but did not click } d_i \end{cases} \tag{2}$$

In counterfactual learning to rank, the user examination is modelled as propensity. In learning from logged bandit feedback, only the examined documents are considered. In the online setting, however, it is unclear how to determine which documents the user has examined (e.g. a user may have considered a snippet, but did not click on it). We make the assumption that users always examine documents from top to bottom, and thus consider the documents ranked above the one that was clicked last as having been examined. With this in place, the reward function described in Eq. 2 can be used to assign rewards to documents in the result list.

Unlike traditional DBGD-based OLTR which ranks documents according to the scores assigned by the ranking function (i.e., deterministically), COLTR creates the result list to be provided to the user for gathering feedback by sampling documents from a known probability distribution. That is, document d_i is drawn from a distribution $p(d_i|f_\theta, D)$ computed by the logging ranker θ. We use softmax to convert document scores into a probability distribution:

$$p(d_i|f_\theta, D) = \frac{e^{\frac{f_\theta(d_i)}{\tau}}}{\sum_{d \in D} e^{\frac{f_\theta(d)}{\tau}}} \tag{3}$$

where τ is the temperature parameter, which is commonly used in the field of reinforcement learning to control the sharpness of the probability distribution [31]. For high values of τ ($\tau \to \infty$), the distribution becomes uniform. For low values ($\tau \to 0$), the probability of the document with the highest score tends to 1. After a document has been picked, the probability distribution will be renormalized to avoid sampling duplicates. This kind of probabilistic ranker has been used in previous works [4,14,22].

While it has been proved that the risk estimator in Eq. 1 is an unbiased estimator, it does suffer from the propensity overfitting problem [34], i.e., the learning algorithm may learn a ranker that assigns small probability values over all the documents d_i in the result list, as this can minimize the risk function. To address this problem, we use the self-normalized risk estimator $\mathcal{R}^{SN}(f_{\theta'})$ (similar to [34]):

$$\mathcal{R}^{SN}(f_{\theta'}) = \frac{\mathcal{R}(f_{\theta'})}{\mathcal{S}(f_{\theta'})} \tag{4}$$

where:

$$\mathcal{S}(f_{\theta'}) = \frac{1}{k} \sum_{i=1}^{k} \frac{p(d_i|f_{\theta'}, D)}{p(d_i|f_\theta, D)} \tag{5}$$

Algorithm 1. Counterfactual Online Learning to Rank (COLTR).

Input: Initial weights θ_1, ranking function f, reward function δ, number of candidate ranker n, learning rate α, step size η, variance control λ;
for $t \leftarrow 1....\infty$ **do**
  $q_t \leftarrow recive\_query(t)$
  $D_t \leftarrow get\_canditate\_set(q_t)$
  $L_t \leftarrow sample\_list(f_{\theta_t}, D_t)$ // Eq.3
  $\delta_t \leftarrow recive\_clicks(L_t)$ // Eq.2
  $C \leftarrow [\ ]$ // create an empty candidate ranker pool
  for $i \leftarrow 1....n$ **do**
    $u_i \leftarrow sample\_unit\_vector()$
    $\theta_i \leftarrow \theta_t + \eta u_i$ // create a candidate ranker
    $append(C, \theta_i)$ // add the new ranker to the candidate pool
  end for
  $W \leftarrow infer\_winners(\delta_t, \theta_t, C, L_t, D_t, \lambda)$ //counterfactual evaluation, see Alg.2
  $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{|W|} \sum_{j \in W} u_j$ //update θ_t to the mean of winners' unit vector
end for

Intuitively, if propensity overfitting does occur, $\mathcal{S}(f_{\theta'})$ will be small, giving a penalty to $\mathcal{R}^{SN}(f_{\theta'})$.

Following the CRM principle, the aim of the learning algorithm is to find a ranker with feature weights θ that can optimize the self-normalized risk estimator, as well as its empirical standard deviation; formally:

$$\theta^{CRM} = argmin \left(\mathcal{R}^{SN}(f_\theta^{'}) + \lambda \sqrt{\frac{Var(\mathcal{R}^{SN}(f_\theta^{'}))}{k}} \right) \tag{6}$$

The $Var(\mathcal{R}^{SN}(f_\theta^{'}))$ is the empirical variance of $\mathcal{R}^{SN}(f_{\theta'})$, to compute which we use an approximate variance estimation [27], where $\lambda = 1$ controls the impact of empirical variance:

$$Var(\mathcal{R}^{SN}(f_\theta^{'})) = \frac{\sum_{i=1}^{k} \left(\delta_i - \mathcal{R}^{SN}(f_{\theta'})\right)^2 \left(\frac{p(d_i|f_{\theta'}, D)}{p(d_i|f_\theta, D}\right)^2}{\left(\sum_{i=1}^{k} \frac{p(d_i|f_{\theta'}, D)}{p(d_i|f_\theta, D)}\right)^2} \tag{7}$$

3.2 Learning a Ranker with COLTR

The previous section described the counterfactual evaluation that can be used in an online learning to rank setting. Next, we introduce the COLTR algorithm that can leverage the counterfactual evaluation to update the current ranker weights θ_t. COLTR uses the DBGD framework to optimize the current production ranker, but it does not rely on interleaving or multileaving comparisons.

Algorithm 2. Counterfactual Evaluation ($infer\_winners(\delta_t, \theta_t, C, L_t, D_t, \lambda)$).

Input: rewards δ_t, logging ranker θ_t candidate ranker set C, result list L_t, candidate document set D_t, variance control λ;
$\mathcal{R} \leftarrow [\theta_t]$, $k \leftarrow length(L_t)$
for θ_i in C **do**
  $r_i \leftarrow 0$, $s_i \leftarrow 0$
  for d_j in L_t **do**
    $p \leftarrow p(d_j|f_{\theta_t}, D_t)$ // compute the logging probability using Eq.3
    $p^{'} \leftarrow p(d_j|f_{\theta_i}, D_t)$ // compute the new ranker probability using Eq.3
    $r_i \leftarrow r_i + \delta_i \frac{p^{'}}{p}$ //compute the $R(\theta_i)$ using Eq.1
    $s_i \leftarrow s_i + \frac{p^{'}}{p}$ //compute the $S(\theta_i)$ using Eq.5
  end for
  $r_i^{SN} \leftarrow \frac{r_i}{s_i}$, $v_i \leftarrow Var(r_i^{SN})$ //Eq.4 and Eq.7
  $append(\mathcal{R}, r_i^{SN} + \lambda\sqrt{\frac{v_i}{k}})$ //Eq.6
end for
return $where(\mathcal{R} < \mathcal{R}[0]) - 1$ // indexes of candidate ranker that has lower risk

Algorithm 1 describes the COLTR updating process: similar to DBGD, it requires the initial ranker weights θ_1, the learning rate α which is used to control the update speed, and the step size η which controls the gradient size. At each timestamp t, i.e., at each round of user interactions (line 2), the search engine receives a query q_t issued by a user (line 3). Then the candidate document set D_t is generated given q_t (line 4), and the results list L_t is created by sampling documents d_i without replacement from the probability distribution computed by Eq. 3 (line 5). The results list is then presented to the user and clicks observed. Then the reward label vector δ_t is generated according to Eq. 2 (line 6)[2]. Next, an empty candidate ranker pool C is created (line 7) and candidate rankers are generated and added to the pool (lines 8–12). Counterfactual evaluation is used to compute the risk associated to each ranker, as described in Algorithm 2. The rankers with a risk lower than the logging ranker are said to win and are placed in the set W (line 13). Finally, the current ranker weights are updated by adding the mean of the winners' unit vector (line 14) modulated by the learning rate α.

The method COLTR uses for computing gradients is similar to that of DBGD with Multileaving (PMGD) [29]. However, COLTR is more efficient. In fact, it does not need to generate an interleaved or multileaved result list for exploring user preferences. When the length of the result list is large, the computational cost for multileaving becomes considerable. In addition, using online evaluation to infer outcomes is very expensive, especially for probabilistic multileaving evaluation [28]: this type of evaluation requires sampling a large number of ranking assignments to decide which ones are the winner rankers – a computationally expensive operation. In contrast, the time complexity for counterfactual evaluation increases linearly with the number of candidate rankers (the for loop in

[2] Note that the length of δ_t is equal to the length of L_t.

Algorithm 2, line 3[3]). To compute the probabilities of sampling documents for the logging and new rankers (Algorithm 2, line 6 and 7), the document scores in Eq. 3 need to be renormalized after each rank: this attracts additional computational cost. For efficiency reasons, we approximate these probabilities by assuming independence, so that we can compute the probabilities only once[4]. As a result, COLTR can efficiently compare a large number of candidate rankers at each interaction.

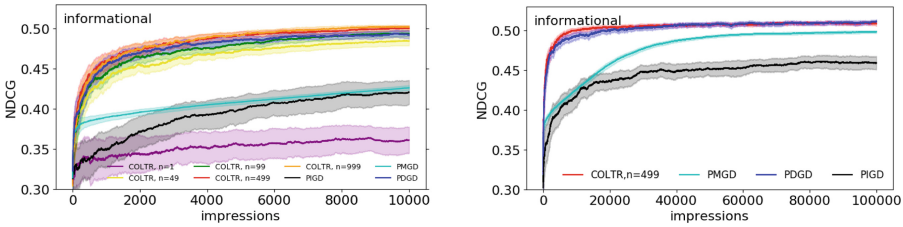
4 Empirical Evaluation

Datasets. We used four publicly available web search LTR datasets to evaluate COLTR. Each dataset contains query-document pair features and (graded) relevance labels. All feature values are normalised using MinMax at the query level. The datasets are split into training, validation and test sets using the splits according to the datasets. The smallest datasets in our experiments are MQ2007 (1,700 queries) and MQ2008 (800 queries) [25], which are a subset of LETOR 4.0. They rely on the Gov2 collection and the query set from the TREC Million Query Track [1]. Query-document pairs are represented with respect to 46 features and 3-graded relevance (from 0, not relevant, to 2, very relevant). In addition to these datasets, we use the larger MLSR-WEB10K [25] and Yahoo! Learning to Rank Challenge datasets [5]. Data for these datasets comes from commercial search engines (Bing and Yahoo, respectively), and relevance labels are assigned on a five-point scale (0 to 4). MLSR-WEB10K contains 10,000 queries and 125 retrieved documents on average, which are represented with respect to 136 features; while, Yahoo! is the largest dataset we consider, with 29,921 queries and 709,877 documents, represented using 700 features.

Simulating User Behaviour. Following previous OLTR work [9,11,22,23,29,41], we use the cascade click model (CCM) [6,8] to generate user clicks. This click model assumes users examine documents in the result list from top to bottom and decide to click with a probability $p(click = 1|R)$, where R is the relevance grade of the examined document. After a document is clicked, the user may stop examining the remainder of the list with probability $p(stop = 1|R)$. In line with previous work, we study three different user behaviours and the corresponding click models. The *perfect* model simulates the user who clicks on every relevant document in the result list and never clicks on non-relevant documents. The *navigational* model simulates the user looking for a single highly relevant document and thus is unlikely to continue after finding the first relevant one. The *informational* model represents the user that searches for topical information and that exhibits a much nosier click behaviour. We use the settings used by previous work for instantiating these click behaviours, e.g., see Table 1 in [23]. In our experiments, the issuing of queries is simulated by uniformly sampling

[3] The length of the results list L_t is fixed.

[4] We empirically observed that this assumption does not deteriorate the effectiveness of the method.



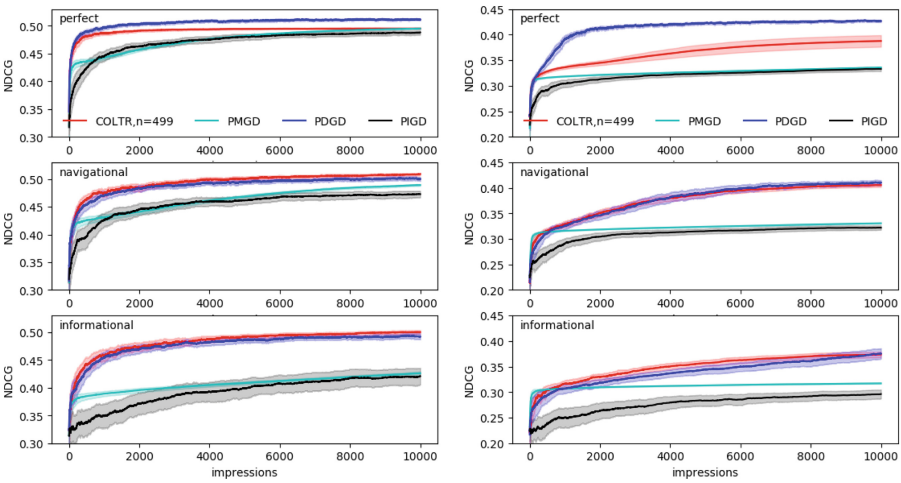
(a) Offline performance for 10,000 impressions, for COLTR under different settings of n and the baselines.

(b) Long term offline performance (100,000 impressions). For clarity, only $n = 499$ is reported for COLTR.

Fig. 1. Offline performance on the MQ2007 with the informational click model.

from the training dataset (line 3 in Algorithm 1). Then a result list is generated in answer to the query and the list is then displayed to the user. Finally, user's clicks on displayed results are simulated using CCM.

Baselines. Three baselines are considered for comparison with COLTR. The traditional DBGD with probabilistic interleaving (PIGD) [12] is used as a representative OLTR method – note that COLTR also uses DBGD, but with counterfactual evaluation in place of the interleaving method. For PIGD, only one candidate ranker is sampled at each interaction; sampling occurs by randomly varying feature weights on the unit sphere with step size $\eta = 1$, and updating the current ranker with learning rate $\alpha = 0.01$. The Probabilistic Multileaving Gradient Descent method (PMGD) [23] is also used in our experiments, as it is the DBGD-based method that has been reported to achieve the highest performance so far for this class of approaches [21]. For this baseline, we use the same parameters settings reported in previous work [22], where the number of candidates was set to $n = 49$, step size to $\eta = 1$ and learning rate to $\alpha = 0.01$. The third baseline we consider is the Pairwise Differentiable Gradient Descent (PDGD) [22], which is the current state-of-the-art OLTR method. We set PDGD's parameters according to Oosterhuis et al. [22], and specify learning rate $\alpha = 0.1$ and use zero initialization. For COLTR, we use $\eta = 1$. We use a learning rate decay for α: in this case, α starts at 0.1 and decreases according to $\alpha = \alpha * 0.99966$ after each update. We set the temperature parameter $\tau = 0.1$ when sampling documents and test different numbers of candidate rankers from $n = 1$ to $n = 999$. For all experiments, we only display $k = 10$ documents to the user, and all methods are used to optimize a linear ranker. Note, we do not directly compared with Counterfactual LTR approaches like that of Joachims et al. [18] because we consider an online setup (while counterfactual LTR requires a large datasets of previous interactions, and the estimation of propensity, which is unfeasible to be performed in an online setting).



(a) MQ2007 dataset (b) MSLR-WEB10K

Fig. 2. Offline performance under three different click models

Evaluation Measures. The Effectiveness of the considered OLTR methods is measured with respect to both *offline* and *online* performance. For offline performance, we average the nDCG@10 scores of the production ranker over the queries in the held-out test set. This measure indicates the effectiveness of the learned ranker. The offline performance of each method is measured for 10,000 impressions, and the final offline performance is also recorded. Online performance is computed as the nDCG@10 score produced by the result list displayed to the user during the training phase [13]. This measure indicates the quality of the user experience during training. A discount factor γ is used to ensure that long-term impressions have less impact, i.e. $\sum_{t=1} nDCG(L_t) \cdot \gamma^{t-1}$. Following previous work [21–23], we choose $\gamma = 0.9995$ so that impressions after the horizon of 10,000 have less than a 1% impact. We repeated each experiment 125 times, spread over different training folds. The evaluation results are averaged and statistically significant differences between system pairs are computed using a two-tailed t-test.

5 Results Analysis

5.1 Offline Performance: Final Ranker Convergence

We first investigate how the number of candidate rankers impacts offline performance. Figure 1(a) displays the offline nDCG of COLTR and the baselines under the informational click setting when a different number of candidate rankers is

used by COLTR (recall that PIGD uses two rankers and PMGD uses 49 rankers). Consider COLTR with one candidate ranker in addition to the production ranker ($n = 1$) and PIGD: both are considering a single alternative ranker to that in production. From the figure, it is clear that PIGD achieves a better offline performance than COLTR. However, when more candidate rankers are considered, e.g., n is increased to 49, the offline performance of COLTR becomes significantly higher than that of PIGD. Furthermore, COLTR is also better than PMGD when the same number of candidate rankers are considered. Moreover, COLTR allows to efficiently compare a large number of candidate rankers at each interaction (impression), and thus can test with a larger set of candidate rankers. We find that increasing the number of candidate rankers can help boosting the offline performance of COLTR and achieve a higher final converge. When $n = 499$, COLTR can reach significantly better ($p < 0.01$) offline performance than PDGD, the current state-of-the-art OLTR method. However, beyond $n = 499$ there are only minor improvements in offline performance, achieved at a higher computational cost – thus, in the remaining experiments, we consider only $n = 499$.

We also consider long-term convergence. Figure 1(b) displays the results for COLTR (with $n = 499$) and the baselines after 100,000 impressions. Because a learning rate decay is used in COLTR, the learning rate becomes insignificant after 30,000 impressions. In order to prevent this to happen, we stop the learning rate decay when $\alpha < 0.01$, and we leave $\alpha = 0.01$ constant for the remaining impressions. The figure shows that, contrary to the results in Fig. 1(a), PMGD can reach much higher performance than PIGD when enough impressions are considered – this finding is consistent with previously reported observations [22]. Nevertheless, both COLTR and PDGD are still significantly better than PIGD and PMGD, and have similar convergence: their offline performance is less affected by the long term impressions.

Figure 2 displays the offline performance across datasets of varying dimensions (small: MQ2007, and large: MSLR-WEB10K) under three different click models and for 10,000 impressions. The results show that PDGD and COLTR outperform PIGD and PMGD for all click models. We also find that, overall, COLTR and the current state-of-the-art online LTR approach, PDGD have very similar learning curves across all click models and datasets, apart for the perfect click model on the MSLR-WEB10K dataset, for which COLTR is severely outperformed by PDGD. Note, the trends observed in Fig. 2 found also for the majority of the remaining datasets. For space reasons, we omit these results from the paper, but we make them available as an online appendix at http://ielab.io/COLTR. Table 1 reports the final convergence performance for all datasets and click models (including statistical significance analysis), displaying similar trends across the considered datasets.

Table 1. Offline nDCG performance obtained under different click models. Significant gains and losses of COLTR over PIGD, PMGD and PDGD are marked by $^{\triangle}$, $^{\triangledown}$ ($p < 0.05$) and $^{\blacktriangle}$, $^{\blacktriangledown}$ ($p < 0.01$) respectively.

		MQ2007	MQ2008	MSLR10K	Yahoo!
Perfect	PIGD	0.488	0.684	0.333	0.677
	PMGD	0.495	0.689	0.336	0.716
	PDGD	0.511	0.699	0.427	0.734
	COLTR, n = 499	$0.495^{\blacktriangle}$ $^{\blacktriangledown}$	0.682 $^{\triangledown}$ $^{\blacktriangledown}$	$0.388^{\blacktriangle}$ $^{\blacktriangle}$ $^{\blacktriangledown}$	0.718 $^{\blacktriangle}$ $^{\blacktriangle}$ $^{\blacktriangledown}$
Navig.	PIGD	0.473	0.670	0.322	0.642
	PMGD	0.489	0.681	0.330	0.709
	PDGD	0.500	0.696	0.410	0.718
	COLTR, n = 499	0.508 $^{\blacktriangle}$ $^{\blacktriangle}$ $^{\blacktriangle}$	$0.689^{\blacktriangle}$ $^{\triangle}$ $^{\triangledown}$	$0.405^{\blacktriangle}$ $^{\blacktriangle}$ $^{\triangledown}$	0.718 $^{\blacktriangle}$ $^{\blacktriangle}$
Inform.	PIGD	0.421	0.641	0.296	0.605
	PMGD	0.426	0.687	0.317	0.677
	PDGD	0.492	0.693	0.375	0.709
	COLTR, n = 499	0.500 $^{\blacktriangle}$ $^{\blacktriangle}$ $^{\blacktriangle}$	$0.686^{\blacktriangle}$ $^{\blacktriangledown}$	$0.374^{\blacktriangle}$ $^{\blacktriangle}$	0.706 $^{\blacktriangle}$ $^{\blacktriangle}$ $^{\triangledown}$

5.2 Online Performance: User Experience

Along with the performance obtained by rankers once training is over, the user experience obtained during training should also be considered. Table 2 reports the online performance of all methods, for all datasets and click models. The state-of-the-art PDGD has the best online performance across all conditions. COLTR outperforms PIGD and PMGD when considering the perfect click model. For other click models, COLTR is better than PIGD but it does provide less cumulative online performance than PMGD, even if it achieves a better offline performance. We posit that this is because PMGD uses a deterministic ranking function to create the result list the user observes, and via multileaving it guarantees that the interleaved result list is not worse than that of the worst candidate ranker. COLTR instead uses a probabilistic ranking function, and if the document sampling distribution is too similar to a uniform distribution, the result list may incorrectly contain many non-relevant documents: this results in a bad online performance. A uniform sampling distribution is obtained because noisy clicks result in some candidate rankers randomly winning the counterfactual evaluation and thus slowing down the gradient convergence and achieving an "elastic effect", where the weight vectors go forward in one interaction, and backwards in the next. This will cause the margins between the documents' scores assigned by the ranking function to become too small and thus the softmax function will not generate a "deterministic" distribution. This also explains why the online performance is much better when clicks are perfect: the gradient directions corresponding to the winning candidates are likely similar, leading the current ranker moving fast through large gradient updates (no elastic effect).

Table 2. Online cumulative nDCG performance under different click models. Significant gains and losses of COLTR over PIGD, PMGD and PDGD are marked by $^{\triangle}$, $^{\triangledown}$ ($p < 0.05$) and $^{\blacktriangle}$, $^{\blacktriangledown}$ ($p < 0.01$) respectively.

		MQ2007	MQ2008	MSLR10K	Yahoo!
Perfect	PIGD	795.6	1184.8	549.8	1202.0
	PMGD	824.8	1225.6	587.6	1284.7
	PDGD	936.1	1345.5	718.5	1407.8
	COLTR, n = 499	933.0$^{\blacktriangle}$ $^{\blacktriangle}$	1344.2 $^{\blacktriangle}$ $^{\blacktriangle}$	641.7$^{\blacktriangle}$ $^{\blacktriangle}$ $^{\blacktriangledown}$	1370.0 $^{\blacktriangle}$ $^{\blacktriangle}$ $^{\blacktriangledown}$
Navig.	PIGD	766.3	1152.1	533.6	1174.1
	PMGD	796.4	1195.9	581.3	1258.4
	PDGD	883.0	1309.0	642.8	1358.9
	COLTR, n = 499	790.7 $^{\blacktriangle}$ $^{\triangledown}$ $^{\blacktriangledown}$	1112.0 $^{\blacktriangledown}$ $^{\blacktriangledown}$ $^{\blacktriangledown}$	542.9$^{\blacktriangle}$ $^{\blacktriangledown}$ $^{\blacktriangledown}$	1194.8$^{\blacktriangle}$ $^{\blacktriangledown}$ $^{\blacktriangledown}$
Inform.	PIGD	681.8	1068.3	483.8	1149.6
	PMGD	745.7	1188.3	575.8	1237.9
	PDGD	859.5	1297.5	600.6	1325.4
	COLTR, n = 499	780.9 $^{\blacktriangle}$ $^{\blacktriangle}$ $^{\blacktriangledown}$	1138.7 $^{\blacktriangle}$ $^{\blacktriangledown}$ $^{\blacktriangledown}$	522.1 $^{\blacktriangle}$ $^{\blacktriangledown}$ $^{\blacktriangledown}$	1186.5 $^{\blacktriangle}$ $^{\blacktriangledown}$ $^{\blacktriangledown}$

6 Conclusion

In this paper, we have presented a novel online learning to rank algorithm that combines the key aspects of counterfactual learning and OLTR. Our method, *counterfactual online learning to rank* (COLTR), replaces online evaluation, which is the most computational expensive step in the traditional DBGD-style OLTR methods, with counterfactual evaluation. COLTR does not derive a gradient function and use it to optimise an objective, but still samples different rankers, akin to the online evaluation practice. As a result, COLTR can evaluate a large number of candidate rankers at a much lower computational expense.

Our empirical results, based on publicly available web search LTR datasets, also show that the COLTR can significantly outperform DBGD-style OLTR methods across different datasets and click models for offline performance. We also find that COLTR achieves the same offline performance as the state-of-the-art OLTR model, the PDGD, across all datasets under noisy click settings. This means COLTR can provide a robust and effective ranker to be deployed into production, once trained online. However, due to the uniform sampling distribution employed by COLTR to select among candidate documents, COLTR has worse online performance than PMGD and PDGD.

Future work will investigate the difference between gradients provided by PDGD and COLTR, as they both use a probabilistic ranker to create the result list. This analysis could provide further indications about the reasons why the online performance of COLTR is limited. Other improvements could be implemented for COLTR. First, instead of stochastically learning at each interaction, historical user interaction data could be used to perform batch learning, which

may provide even more reliable gradients under noisy clicks. Note that this extension is possible, and methodologically simple for COLTR, but not for PDGD. Second, the use of the exploration variance reduction method [35,36] could be investigated to reduce the gradient exploration space: this may solve the uniform sampling distribution problem.

Acknowledgements. Dr Guido Zuccon is the recipient of an Australian Research Council DECRA Research Fellowship (DE180101579) and a Google Faculty Award.

References

1. Allan, J., Carterette, B., Aslam, J.A., Pavlu, V., Dachev, B., Kanoulas, E.: Million query track 2007 overview. In: TREC Proceedings (2007)
2. Baeza-Yates, R.: Bias on the web. Commun. ACM **61**(6), 54–61 (2018)
3. Bottou, L., et al.: Counterfactual reasoning and learning systems: the example of computational advertising. J. Mach. Learn. Res. **14**(1), 3207–3260 (2013)
4. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th International Conference on Machine Learning, pp. 129–136. ACM (2007)
5. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Proceedings of the Learning to Rank Challenge, pp. 1–24 (2011)
6. Chuklin, A., Markov, I., Rijke, M.D.: Click models for web search. Synth. Lect. Inf. Concepts Retrieval Serv. **7**(3), 1–115 (2015)
7. Guan, Z., Cutrell, E.: An eye tracking study of the effect of target rank on web search. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2007, pp. 417–420. ACM, New York (2007)
8. Guo, F., Liu, C., Wang, Y.M.: Efficient multiple-click models in web search. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining, pp. 124–131. ACM (2009)
9. He, J., Zhai, C., Li, X.: Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 2029–2032. ACM (2009)
10. Hofmann, K., Schuth, A., Whiteson, S., de Rijke, M.: Reusing historical interaction data for faster online learning to rank for IR. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pp. 183–192. ACM (2013)
11. Hofmann, K., Whiteson, S., de Rijke, M.: Balancing exploration and exploitation in learning to rank online. In: Clough, P., et al. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 251–263. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20161-5_25
12. Hofmann, K., Whiteson, S., De Rijke, M.: A probabilistic method for inferring preferences from clicks. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 249–258. ACM (2011)
13. Hofmann, K., et al.: Fast and reliable online learning to rank for information retrieval. In: SIGIR Forum, vol. 47, p. 140 (2013)
14. Jagerman, R., Oosterhuis, H., de Rijke, M.: To model or to intervene: a comparison of counterfactual and online learning to rank from user interactions. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, pp. 15–24. Association for Computing Machinery (2019)

15. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 133–142. ACM (2002)
16. Joachims, T., Granka, L.A., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback. SIGIR **5**, 154–161 (2005)
17. Joachims, T., Swaminathan, A., de Rijke, M.: Deep learning with logged bandit feedback. In: The Sixth International Conference on Learning Representations (ICLR) (2018)
18. Joachims, T., Swaminathan, A., Schnabel, T.: Unbiased learning-to-rank with biased feedback. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 781–789. ACM (2017)
19. Lefortier, D., Serdyukov, P., De Rijke, M.: Online exploration for detecting shifts in fresh intent. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 589–598. ACM (2014)
20. Liu, T.Y., et al.: Learning to rank for information retrieval. Found. Trends Inf. Retrieval **3**(3), 225–331 (2009)
21. Oosterhuis, H., de Rijke, M.: Balancing speed and quality in online learning to rank for information retrieval. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 277–286. ACM (2017)
22. Oosterhuis, H., de Rijke, M.: Differentiable unbiased online learning to rank. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1293–1302. ACM (2018)
23. Oosterhuis, H., Schuth, A., de Rijke, M.: Probabilistic multileave gradient descent. In: Ferro, N., et al. (eds.) ECIR 2016. LNCS, vol. 9626, pp. 661–668. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30671-1_50
24. Pan, B., Hembrooke, H., Joachims, T., Lorigo, L., Gay, G., Granka, L.: In google we trust: users' decisions on rank, position, and relevance. J. Comput.-Mediat. Commun. **12**(3), 801–823 (2007)
25. Qin, T., Liu, T.Y.: Introducing LETOR 4.0 datasets. arXiv preprint arXiv:1306.2597 (2013)
26. Radlinski, F., Kurup, M., Joachims, T.: How does clickthrough data reflect retrieval quality? In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 43–52. ACM (2008)
27. Rubinstein, R.Y., Kroese, D.P.: Simulation and the Monte Carlo Method, vol. 10. Wiley, Hoboken (2016)
28. Schuth, A., et al.: Probabilistic multileave for online retrieval evaluation. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 955–958. ACM (2015)
29. Schuth, A., Oosterhuis, H., Whiteson, S., de Rijke, M.: Multileave gradient descent for fast online learning to rank. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp. 457–466. ACM (2016)
30. Schuth, A., Sietsma, F., Whiteson, S., Lefortier, D., de Rijke, M.: Multileaved comparisons for fast online evaluation. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 71–80. ACM (2014)
31. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2011)
32. Swaminathan, A., Joachims, T.: Batch learning from logged bandit feedback through counterfactual risk minimization. J. Mach. Learn. Res. **16**(1), 1731–1755 (2015)

33. Swaminathan, A., Joachims, T.: Counterfactual risk minimization: learning from logged bandit feedback. In: International Conference on Machine Learning, pp. 814–823 (2015)
34. Swaminathan, A., Joachims, T.: The self-normalized estimator for counterfactual learning. In: Advances in Neural Information Processing Systems, pp. 3231–3239 (2015)
35. Wang, H., Kim, S., McCord-Snook, E., Wu, Q., Wang, H.: Variance reduction in gradient exploration for online learning to rank. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019 (2019)
36. Wang, H., Langley, R., Kim, S., McCord-Snook, E., Wang, H.: Efficient exploration of gradient space for online learning to rank. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 145–154. ACM (2018)
37. Wang, X., Bendersky, M., Metzler, D., Najork, M.: Learning to rank with selection bias in personal search. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp. 115–124. ACM (2016)
38. Wang, X., Golbandi, N., Bendersky, M., Metzler, D., Najork, M.: Position bias estimation for unbiased learning to rank in personal search. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 610–618. ACM (2018)
39. Yue, Y., Joachims, T.: Interactively optimizing information retrieval systems as a dueling bandits problem. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1201–1208. ACM (2009)
40. Yue, Y., Patel, R., Roehrig, H.: Beyond position bias: examining result attractiveness as a source of presentation bias in clickthrough data. In: Proceedings of the 19th International Conference on World Wide Web, pp. 1011–1018. ACM (2010)
41. Zoghi, M., Whiteson, S.A., De Rijke, M., Munos, R.: Relative confidence sampling for efficient on-line ranker evaluation. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, pp. 73–82. ACM (2014)