Federated Online Learning to Rank with Evolution Strategies: A Reproducibility Study

No Author Given

No Institute Given

Abstract. Abstract goes here

Keywords: Your keywords optional

1 Introduction

Online learning to rank (OLTR) exploits users queries and interactions with search engine result pages (SERPs) to iteratively train and update a ranker in production [**?**]. In particular, OTR relies on implicit user feedback from interactions on SERPs, e.g., clicks, rather than editorial relevance labels. Several methods for OLTR exist that attempt to address the specific challenges of online learning [**?**]: from making sense of the implicit feedback, to exploring the space of feature weights, accounting for biases in the click signal, reducing the impact of the online learning process on user experience, among others.

An aspect that has not been received wide attention in OLTR is how the privacy of users could be guaranteed. Current methods in fact assume that a central server collects all queries and interactions of all users of the search system, and it is this central server that is responsible for the indexing of the collection, the training of the ranker and the production of the SERPs. A recent work by Kharitonov, however, has attempted to provide a mechanism for OLTR that preserves the privacy of users [4]. The method relies on the federated learning paradigm [**?**], in which data (collection, queries, interactions) is maintained at each client's side along with a copy of the ranker, and updates to the rankers that are learned from the interaction on the client side are shared to the central server, which is responsible for aggregating the update signal from clients and propagate the aggregated ranker update. In this specific case, all users observe and act on the same feature space; each user however retains control of their own data, which includes the collection, the queries and the interactions.

The following research questions are addressed in this paper:

RQ1. Does the Federated Online Learning to Rank algorithm also work on other widely used Learning to Rank datasets? RQ2. Does the number of clients affect the performance of Federated Online Learning to Rank? RQ3. Can federated online learning to rank achieve similar performance comparing with other state-of-the-art online learning to rank methods? RQ4. Can Federated Online Learning to Rank generalise to other typical OLTR evaluation setup, such as nDCG or MRR?

2 Related Work

This section gives a brief overview of traditional LTR, OLTR and federated learning.

2.1 Offline Learning to Rank

The core of Learning to Rank (LTR) is to train a ranking model using Machine Learning methods.

2.2 Online Learning to Rank

Main challenges in online learning to rank problem are: (a) noise and biases of user interactions, (b) non-continuous metrics for optimization.

OLTR updates the ranker through user's online feedback, more specifically, user's click data. Unlike traditional LTR methods, OLTR doesn't need human-annotated data, which is expensive and time-consuming to create. Besides, human-annotated data can not always represent true preference of users. Traditional LTR methods are more suitable in situation where one ranking list fits all users. In a continuously developing world, people's preference and answers to one search query may change or differ from other's. OLTR provides a better way to learning from users' feedback in order to improve users' experience and ranking effectiveness.

2.3 Federated Machine Learning

Federated Machine Learning aims to solve two main challenges existing in today's artificial intelligence research: (a) data existing in the form of isolated islands, (b) the growing concern for data privacy and security [15].

The concept of Federated Learning was first proposed by Google in 2016 [6,7]. The first Federated Learning algorithm is Federated Averaging [8]. In early year's research, the Federated Learning aims to train Machine Learning models using data from multiple data source while preventing data leakage.

3 Methodology

federated online learning to rank with evolution strategies:
In this work, Online Learning to Rank problem is extended to Federated Learning setup using Evolution Strategies optimization, which is widely used in Reinforcement Learning algorithm.
The FOLtR-ES algorithm contains three parts. First, it follows federated learning optimization scenario. Second, it uses Evolution Strategies to estimate gradients. Finally, it introduces a privatization procedure to protect users' privacy.

3.1 Optimization Scenario

The optimization scenario can be illustrated as several steps. First, the client downloads the most recently updated ranking model from the server. After B interactions is performed by the client, the performance metrics of the these interactions are averaged in the client side and a privatized message is sent to the centralized server. After receiving messages from N clients, the server combines them to estimate a single gradient g and performs an optimization step to update the current ranking model. Finally, the clients download the newly updated model from the server.

3.2 Gradient Estimation

Assume that the ranking model comes from a parametric family indexed by vector $\theta \in R^n$. Each time a user u has an interaction a, the ranking quality is denoted as f. The goal of optimization is to find model vector θ^* that can maximize the mean of metric f across all interactions a from all users u:

$$\theta^* = \arg\max_{\theta} F(\theta) = \arg\max_{\theta} \mathbb{E}_u \mathbb{E}_{a|u,\theta} f(a;\theta,u) \tag{1}$$

By using Evolution Strategies (ES) [12], FOLtR-ES algorithm considers a population of parameter vectors which follow the distribution with a density function $p_\phi(\theta)$. The objective turns to finding the distribution parameter ϕ that can maximize the expectation of metrics across the population:

$$\mathbb{E}_{\theta\sim p_\phi(\theta)}\ [F(\theta)] \tag{2}$$

The gradient g of Equation (2) is obtained in a fashion similar to REINFORCE [14]:

$$\begin{aligned} g = \nabla_\phi \mathbb{E}_\theta[F(\theta)] &= \nabla_\phi \int_\theta p_\phi(\theta)F(\theta)d\theta = \int_\theta F(\theta)\nabla_\phi p_\phi(\theta)d\theta = \\ &= \int_\theta F(\theta)p_\phi(\theta)\left(\nabla_\phi \log p_\phi(\theta)\right)d\theta = \mathbb{E}_\theta\left[F(\theta)\cdot\nabla_\phi \log p_\phi(\theta)\right] \end{aligned} \tag{3}$$

Same as Evolution Strategies, FOLtR-ES instantiates population distribution $p_\phi(\theta)$ as an isotropic multivariate Gaussian distribution with mean ϕ and fixed diagonal covariance matrix $\sigma^2 I$. Thus a simple form of gradient estimation is denoted as:

$$g = \mathbb{E}_{\theta\sim p_\phi(\theta)}\left[F(\theta)\cdot\frac{1}{\sigma^2}(\theta-\phi)\right] \tag{4}$$

Based on federated optimization scenario, θ is sampled independently on client side. Combined with the definition of $F(\theta)$ in Equation (1), the gradient can be obtained as:

$$g = \mathbb{E}_u \mathbb{E}_{\theta\sim p_\phi(\theta)}\left[\left(\mathbb{E}_{a|u,\theta} f(a;\theta,u)\right)\cdot\frac{1}{\sigma^2}(\theta-\phi)\right] \tag{5}$$

To get the estimate $\hat{g}$ of g from Equation (5), $\hat{g} \approx g$, following steps are adopted: (a) each client u randomly generates a pseudo-random seed s and uses the seed to sample a perturbed model $\theta_s \sim \mathbb{N}\left(\phi, \sigma^2 I\right)$, (b) uses the average of metric f over B interactions to estimate the expected loss $\hat{f} \approx \mathbb{E}_{a|u,\theta_s} f(a;\theta_s,u)$ from Equation (5), (c) communicates the message tuple $(s,\hat{f})$ to the server, (d) the centralized server can get the estimate $\hat{g}$ of Equation (5) according to all message sent from N clients.

To reduce the variance of gradient estimates, means of antithetic variates are used in FOLtR-ES, which is also a ES trick [12]. The algorithm in gradient estimation follows standard ES except that the random seeds are sampled in client side.

3.3 Privatization Procedure

To ensure that the clients' privacy is fully protected, FOLtR-ES proposes a privatization procedure that introduces privatization noise in the communication procedure between clients and the server.

Assume that the metric used on client side is discrete or can be discretized if continuous. Thus the metric takes a finite number (n) of values, $f_0, f_1, ..., f_{n-1}$. For each time the client experiences interaction, the true value of the metric is denoted as f_0 and the rest of $n-1$ values are different from f_0. In privatization procedure, true metric value f_0 is sent with probability p. Otherwise, with probability $1-p$, send the randomly selected value $\hat{f}$ out of the remaining $n-1$ values. To ensure the same optimization goal described in section 3.3, FOLtR-ES assumes that the probability $p > 1/n$.

Unlike other Federated Learning work, FOLtR-ES adopts a strict notion of ϵ-local differential privacy [], in which the privacy is considered at the level of client side instead of the server. Through the privatization procedure, ϵ-local differential privacy is achieved. And the upper bound of ϵ is:

$$\epsilon \leq log\frac{p(n-1)}{1-p} \tag{6}$$

This means through the privatization scheme, at least $\log[p(m-1)/(1-p)]$-local differential privacy can be achieved. At the same time, any ϵ-local differential private mechanism also can obtain ϵ-differential privacy [].

4 Experimental Settings

In this section, we introduce the experimental settings to answer the research questions addressed in Section. 1.

4.1 Datasets

The datasets we use are MQ2007 and MQ2008 [11], MLSR-WEB10K [11] and Yahoo! Webscope [1] datasets, which are commonly-used offline learning to rank datasets. Each dataset contains a centain amount of queries and corresponding candidate documents. The query-document pairs are represented by feature vectors and relevance labels. For MQ2007 and MQ2008 datasets, the relevance labels value as 0 (not relevant), 1 (partially relevant) and 2 (perfecly relevant). The relevance annotations of MLSR-WEB10K range from 0 (not relevant) to 4 (perfectly relevant). Each dataset contains five splits, except for Yahoo! Webscope. More detailed information for each dataset can be found in Table 1.

We use the original dataset in training and validating procedure without any preprocessing, such as feature selection or feature normalization.

Table 1: Information for each dataset

	number of		
	queries	relevance labels	features
MQ2007	1692	69623	46
MQ2008	784	15211	46
MLSR-WEB10K	XXX	XXX	136

4.2 Simulation

Because no public datasets with search log is available for online learning to rank, we adopt the similar setup with [3, 13] - simulating users and their reaction to the search results using labelled offline learning to rank datasets.

We follow the same method with FOLtR-ES for simulation. We sample B queries for each client randomly and use the local perturbed model to rank the documents. The length for each ranking list is limited to 10 documents. After simulating user's clicks data, we record the quality metric for each interaction and perform the privatization procedure with probability p. Next, we send the averaged metric and pseudo-random seed to optimize the centralized ranking model. Finally, each client receive the updated ranking model.

For users' click simulation, we use Cascade Click Model (CCM) [2]. We run instances of CCM using the same click probability and stop probability with [4] for MQ2007 and [10] for MLSR-WEB10K. Under CCM, users are assumed to go through the ranking list from top to bottom. Each document is examined and clicked with click probability $P(click = 1|r)$, conditioned on relevance label r. After a click, the user stops with stop probability $P(stop = 1|r)$ or continues otherwise. Usually three instantiations of CCM are considered in OLTR: a $perfect$ user with very reliable feedback, a *navigational* user searching for reasonably relevant documents, and an *informational* user with noisiest feedback among three instantiations. table 2 shows the parameters of three click models.

Table 2: Three click model instantiations for MQ2007 dataset

	$P(click = 1\|r)$			$P(stop = 1\|r)$		
	r = 0	r = 1	r = 2	r = 0	r = 1	r = 2
perfect	0.0	0.5	1.0	0.0	0.0	0.0
navigational	0.05	0.5	0.95	0.2	0.5	0.9
informational	0.4	0.7	0.9	0.1	0.3	0.5

Table 3: Three click model instantiations for MLSR-WEB10K dataset

	$P(click = 1\|r)$					$P(stop = 1\|r)$				
	r = 0	r = 1	r = 2	r = 3	r = 4	r = 0	r = 1	r = 2	r = 3	r = 4
perfect	0.0	0.2	0.4	0.8	1.0	0.0	0.0	0.0	0.0	0.0
navigational	0.05	0.3	0.5	0.7	0.95	0.2	0.3	0.5	0.7	0.9
informational	0.4	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5

4.3 Evaluation metric

As FOLtR-ES is designed to allow optimization based on any absolute measures of ranking quality, we can extend the original methods to other widely used evaluation metrics for OLTR. For better reproducing and comparison with the original algorithm, we choose the reciprocal rank of the highest clicked result in each interaction, which is the same quality metric used in the original methods. We also use Discounted Cumulative Gain (DCG) to evaluate the users' experience in each interaction and NDCG@10 to evaluate the central ranker as we want to explore the performance of FOLtR-ES on other metrics.

4.4 Baselines

To further study how far the FOLtR-ES is left behind with common OLTR methods on ranking quality, we train Pairwise Differentiable Gradient Descent (PDGD) [9] models as the baselines.

4.5 Models and Optimization

As we do not want to study the influence of ranking models to the original algorithm, we adopt the same models and optimization steps as [4]. The two ranking models are a linear ranker and a neural network with a single hidden layer of size 10. For optimization, we use Adam [5] with default parameters.

5 Results and Analysis

5.1 Effects of number of clients on FOLtR-ES

To study the influence of number of clients, we perform experiments on MQ2007 and MSLR-WEB10K datasets.

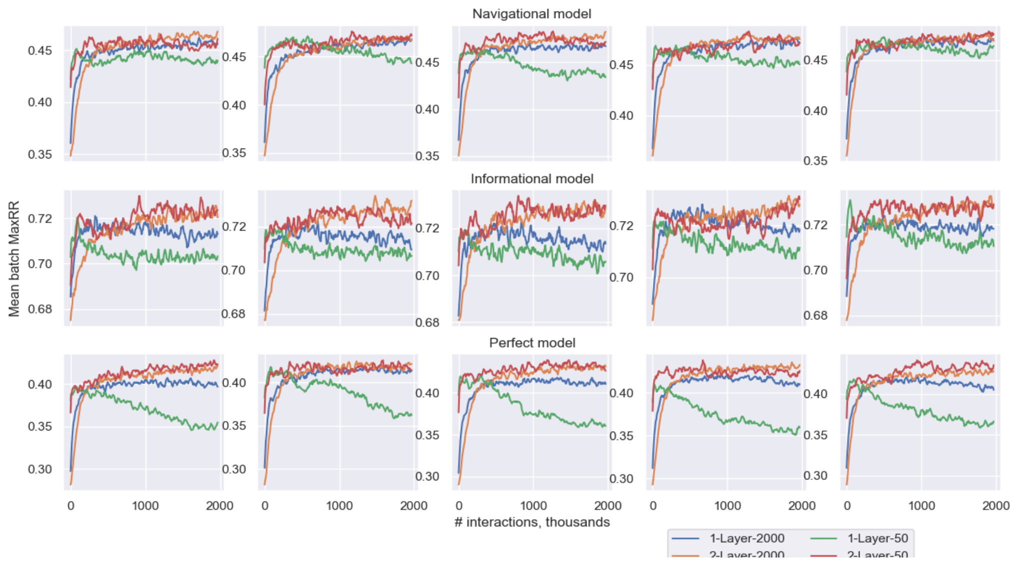

Fig. 1: Mean batch MaxRR for MQ2007 with different client number

5.2 Comparing FOLtR-ES with OLTR baselines

For a fair comparation, we set up the privacy probability $p = 1$ (lowest privacy) in FOLtR-ES. We perform experiments on MQ2007 and MLSR-WEB10K datasets with simulating 2000 clients.

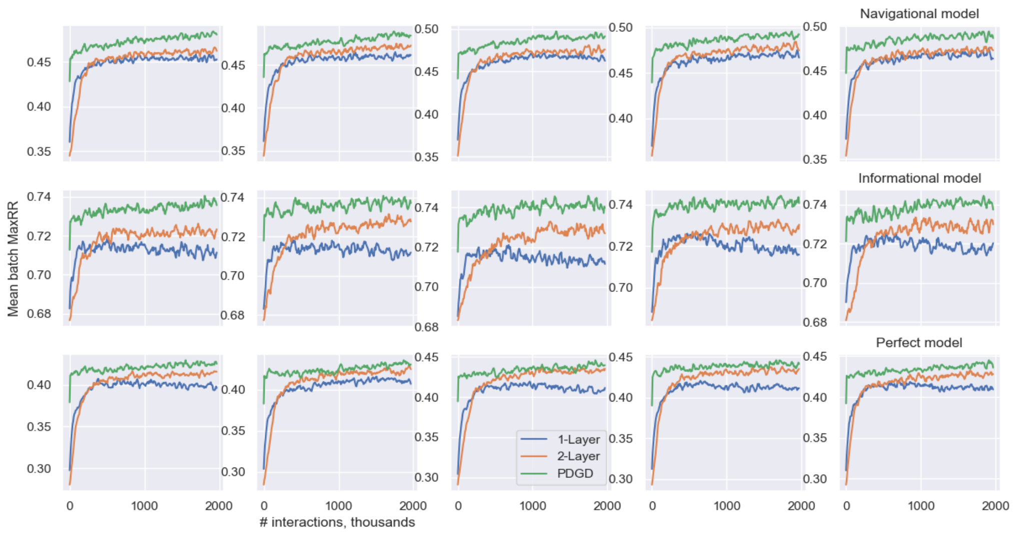

Fig. 2: Mean batch MaxRR for MQ2007 with 2000 clients and $p = 1$

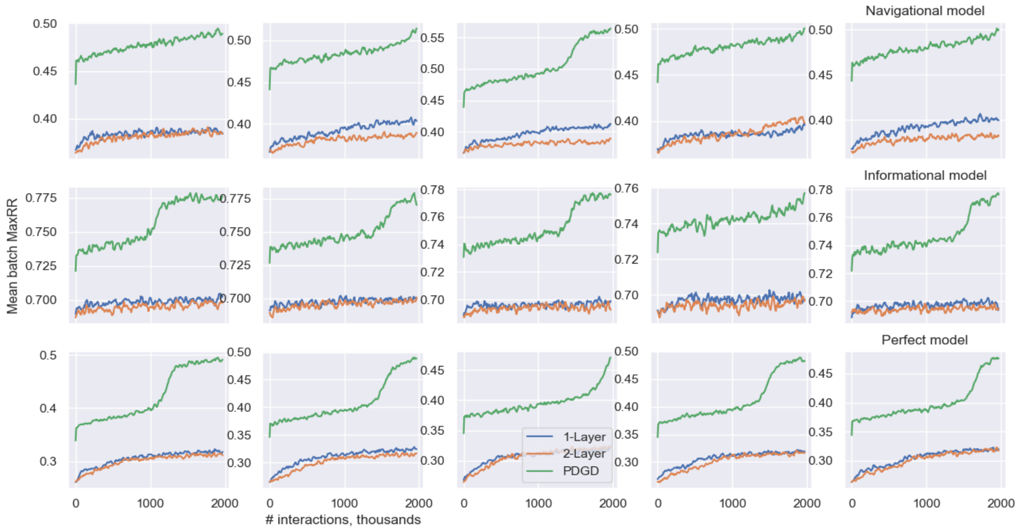

Fig. 3: Mean batch MaxRR for MSLR-WEB10K with 2000 clients and $p = 1$

6 Conclusions

Acknowledgements.

References

1. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Chapelle, O., Chang, Y., Liu, T. (eds.) Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010. JMLR Proceedings, vol. 14, pp. 1–24. JMLR.org (2011), `http://proceedings.mlr.press/v14/chapelle11a.html`
2. Guo, F., Liu, C., Wang, Y.M.: Efficient multiple-click models in web search. In: Baeza-Yates, R., Boldi, P., Ribeiro-Neto, B.A., Cambazoglu, B.B. (eds.) Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009. pp. 124–131. ACM (2009). https://doi.org/10.1145/1498759.1498818, `https://doi.org/10.1145/1498759.1498818`
3. Hofmann, K., Schuth, A., Whiteson, S., de Rijke, M.: Reusing historical interaction data for faster online learning to rank for IR. In: Leonardi, S., Panconesi, A., Ferragina, P., Gionis, A. (eds.) Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013. pp. 183–192. ACM (2013). https://doi.org/10.1145/2433396.2433419, `https://doi.org/10.1145/2433396.2433419`
4. Kharitonov, E.: Federated online learning to rank with evolution strategies. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. pp. 249–257 (2019)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

6. Konecný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. CoRR **abs/1610.02527** (2016), `http://arxiv.org/abs/1610.02527`
7. Konecný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. CoRR **abs/1610.05492** (2016), `http://arxiv.org/abs/1610.05492`
8. McMahan, H.B., Moore, E., Ramage, D., y Arcas, B.A.: Federated learning of deep networks using model averaging. corr abs/1602.05629 (2016). arXiv preprint arXiv:1602.05629 (2016)
9. 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 (2018)
10. Oosterhuis, H., Schuth, A., de Rijke, M.: Probabilistic multileave gradient descent. In: European Conference on Information Retrieval. pp. 661–668. Springer (2016)
11. Qin, T., Liu, T.: Introducing LETOR 4.0 datasets. CoRR **abs/1306.2597** (2013), `http://arxiv.org/abs/1306.2597`
12. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864 (2017)
13. Schuth, A., Oosterhuis, H., Whiteson, S., de Rijke, M.: Multileave gradient descent for fast online learning to rank. In: Bennett, P.N., Josifovski, V., Neville, J., Radlinski, F. (eds.) Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016. pp. 457–466. ACM (2016). https://doi.org/10.1145/2835776.2835804, `https://doi.org/10.1145/2835776.2835804`
14. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3-4), 229–256 (1992)
15. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)