An Experimental Study for Federated Online Learning to Rank with Evolution Strategies

No Author Given

No Institute Given

Abstract. Abstract goes here

Keywords: Your keywords optional

1 Introduction

one sentence for federated learning

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

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) [3], 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:

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

The gradient g of Equation (2) is obtained in a fashion similar to REINFORCE [4]:

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

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 [3]. 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

4.1 Datasets

The datasets we use are MQ2007 and MQ2008 [2], MLSR-WEB10K[2] and Yahoo! Webscope [1] datasets. More detailed information for each dataset can be found in Table X.

MQ2007 dataset contains 1692 unique queries and 69623 relevance labels. MQ2008 dataset contains 784 unique queries and 15211 relevance labels. For MQ2007 and MQ2008 dataset, each query-document pair contains 46 features.

Each dataset is provided with five splits, except for Yahoo! Webscope.

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

4.2 evaluation metric

4.3 baselines

4.4 models

4.5 click models

we simulate uses by applying an instance of Cascade Click Model (CCM) []
(further explain about CCM)
(table for CCM we use)

5 Results and Analysis

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. Qin, T., Liu, T.: Introducing LETOR 4.0 datasets. CoRR **abs/1306.2597** (2013), http://arxiv.org/abs/1306.2597
3. 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)
4. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3-4), 229–256 (1992)