Scalable Exploration for Neural Online Learning to Rank with Perturbed Feedback

Yiling Jia
yj9xs@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

Hongning Wang
hw5x@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

ABSTRACT

Deep neural networks (DNNs) demonstrates significant advantages in improving ranking performance in retrieval tasks. Driven by the recent developments in optimization and generalization of DNNs, learning a neural ranking model online from its interactions with users becomes possible. However, the required exploration for model learning has to be performed in the entire neural network parameter space, which is prohibitively expensive and limits the application of such online solutions in practice.

In this work, we propose an efficient exploration strategy for online interactive neural ranker learning based on bootstrapping. Our solution is based on an ensemble of ranking models trained with perturbed user click feedback. The proposed method eliminates explicit confidence set construction and the associated computational overhead, which enables the online neural rankers training to be efficiently executed in practice with theoretical guarantees. Extensive comparisons with an array of state-of-the-art OL2R algorithms on two public learning to rank benchmark datasets demonstrate the effectiveness and computational efficiency of our proposed neural OL2R solution.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; • **Theory of computation** → **Online learning algorithms**; **Pseudorandomness and derandomization**; **Regret bounds**.

KEYWORDS

Online learning to rank; perturbed exploration; regret analysis

ACM Reference Format:
Yiling Jia and Hongning Wang. 2022. Scalable Exploration for Neural Online Learning to Rank with Perturbed Feedback. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3477495.3532057

1 INTRODUCTION

Online learning to rank (OL2R) has recently attracted great research interest because of its unique advantages in capturing users’ ranking preferences without requiring expensive relevance labeling as in classical offline learning to rank solutions [24, 28, 33, 36, 43, 48, 54]. Because users’ implicit feedback is noisy and biased [2, 10, 26, 27], the key in OL2R is to effectively explore the unknowns for improved relevance estimation, while serving the users with high-quality ranked results, which is known as the explore-exploit trade-off.

Most existing work in OL2R assumes a linear scoring function [43, 48, 54]. Dueling bandit gradient descent (DBGD) and its different variants are the most popularly used OL2R solutions [54], where new model variants are sampled via random perturbations in the parameter space to estimate the direction for model update. As its non-linear extension, pairwise differential gradient descent (PDGD) [40] samples the next ranked document from a Plackett-Luce model and estimates an unbiased gradient from the inferred pairwise preference. PairRank [24] learns a logistic ranker online in a pairwise manner and explores the ranking space based on the model’s estimation uncertainty about the pairwise comparisons of document rankings. Though practically effective, such a linear or generalized linear model assumption is incompetent to capture the possible complex non-linear relations between a document’s ranking features and its relevance quality. This is already proved to be crucial in the past offline learning to rank practices [6, 38]

To unleash the power of representation learning, deep neural networks (DNN) have been introduced to learn the underlying scoring function for document ranking. In [40], PDGD is also experimented on a neural ranker. Though the authors reported promising empirical results, its theoretical property (e.g., convergence) is unknown. On the other hand, enabled by the substantial progress in optimization and generalization of DNNs, quantifying a neural model’s uncertainty on new data points become possible [3, 7, 8, 11, 12]. A recent work named olRankNet [23] extended PairRank with a neural network ranker, which performs exploration in the pairwise document ranking space with topological sort by using the neural tangent kernel technique [22]. Compared with PairRank, olRankNet provided encouraging performance improvement, which was reported to be the best among all state-of-the-art OL2R solutions. More importantly, olRankNet is proved to achieve a sublinear gap-dependent regret upper bound, which is defined on the total number of mis-ordered pairs over the course of interactions with users. To our best knowledge, olRankNet is the first known OL2R solution for neural rankers with theoretical guarantees.

Despite being theoretically sound and empirically effective, olRankNet’s limitation is also remarkably serious: its computational cost for performing the required exploration is prohibitively high (almost *cubic* to the number of neural network’s parameters). More specifically, to quantify the uncertainty of its estimated pairwise preferences among candidate documents, it has to maintain a high-probability confidence set for the current ranker’s parameter estimation over time. However, the construction of the confidence set depends on the dimensionality of the neural network’s parameters:

as required by the neural tangent kernel, an inverse of the covariance matrix computed based on the gradient of the entire neural network is needed whenever the network is updated. For example, for a simple two layer feed-forward neural network with input dimension d and m neurons in each layer, the size of the covariance matrix is $(md+m^2+m)^2$. The best known time complexity for computing the inverse of this covariance matrix is $O\big((md+m^2+m)^{2.373}\big)$, by the optimized Coppersmith–Winograd algorithm [51]. This computational complexity quickly outpaces the limit of any modern computational machinery, given m or d are usually very large in practice (e.g., m is often in the hundreds and d in tens of thousands) and such matrix inverse operation is needed in every round when the neural network is updated. Due to this limitation, olRankNet has to employ the diagonal approximation of the covariance matrix in its actual implementations [23]. But such an approximation loses its all theoretical guarantees, which unfortunately leads to a gap between the theoretical and empirical performance of olRankNet. And even how this gap would depend on the dimensionality of the network and affect olRankNet's performance is completely unknown. This inevitably limits the application of the neural OL2R solutions, especially the olRankNet-type algorithms, in practice.

In this work, we develop an efficient and scalable exploration strategy for olRankNet by eliminating its explicit confidence set construction. The basic idea is to use bootstrapping technique to measure the uncertainty of a neural ranker's output via a set of sample estimates. In particular, we maintain N rankers in parallel. And in each round, after receiving the user's click feedback, each of the rankers is updated with the observed clicks and independently generated pseudo noise from a zero-mean Gaussian distribution. The overall model's estimation uncertainty on a pair of documents is then determined by an ensemble of the estimates from all N rankers. For example, for a document pair (i, j), if all N rankers predict $i \succ j$, (i, j) is considered as in a certain rank order, otherwise it is considered as in an uncertain rank order, where exploration is needed. Besides regular neural network updates, no additional computation is needed, which greatly reduces the computational overhead as required in olRankNet. We name our new solution as Perturbed Pairwise Neural Rank (or P^{2}NeurRank in short). We rigorously prove that with a high probability P^{2}NeurRank obtains the same regret as olRankNet, but the computational complexity is way much lower. In addition, as no approximation is needed in P^{2}NeurRank, its theoretical analysis directly suggests its empirical performance. Our extensive empirical evaluations demonstrate the strong advantage in both efficiency and effectiveness of P^{2}NeurRank against olRankNet and a rich set of state-of-the-art solutions over two OL2R benchmark datasets on standard retrieval metrics.

2 RELATED WORK

Online learning to rank with neural rankers. Most of the existing parametric OL2R solutions are limited to linear ranking models [36, 54]. In particular, DBGD and its extensions [44, 48, 49, 54], as the most popularly referred OL2R solutions, are inherently designed for linear models as they rely on random perturbations in linear model weights for parameter estimation. But such a linear assumption is incompetent to capture any non-linear relations about documents' relevance quality under given queries, which shields such OL2R algorithms away from the successful practices in offline learning to rank models that are empowered by DNNs [6, 41].

Such a limitation motivates some preliminary attempts in OL2R. In [40], pairwise differentiable gradient descent (PDGD) is proposed to sample the next ranked document from a Plackett-Luce model and estimate an unbiased gradient from the inferred pairwise ranking preference. Although improved empirical performance is reported for PDGD with a neural ranker, there is no theoretical guarantee on its online performance. Taking a completely different perspective, PairRank [24] directly learns a pairwise logistic regression ranker online and explores the pairwise ranking space via a divide-and-conquer strategy based on the model's uncertainty about the documents' rankings. The authors claimed logistic regression can be treated as a one-layer feed-forward neural network, but it is unclear how PairRank can be extended to more general neural ranking architectures.

Recently, substantial progress in optimization and generalization of DNNs enables theoretical analysis about the neural models [18, 37, 39, 46, 47, 52, 53, 57, 58]. For example, with the neural tangent kernel technique [22], the uncertainty of a neural model's estimation on new data points can be quantified [25, 55, 56]. Most recently, olRankNet [23] is proposed to extend PairRank with a multi-layer neural ranker. The authors proved that the good theoretical properties of PairRank (i.e., sublinear regret) are inherited in olRankNet, and empirically improved performance over PairRank was also reported in olRankNet. However, one serious issue of olRankNet is its cumbersome computational complexity: to quantify the confidence interval of the neural ranker's output for the exploration purpose, one has to compute the inverse of a covariance matrix which is derived by the gradient of entire neural network. The complexity is almost cubic to the number of parameters in the neural network, which is prohibitively expensive for OL2R, as this matrix inverse is needed every time the ranker is updated. The authors in [23] suggested using diagonal approximation for the covariance matrix, but no guarantee is provided about the impact of such an approximation.

Randomized exploration in online learning. Efficient exploration is critical for online algorithms, as the model learns by actively acquiring feedback from the environment [34]. Distinct from the deterministic exploration strategies, such as upper confidence bound [1, 5], randomization-based exploration enjoys advantages in its light computational overhead and thus has received increasing attention in online learning community. The most straightforward randomization-based exploration strategy is ϵ-greedy [5], which takes the currently estimated best action with probability $1 - \epsilon$, otherwise randomly take an action. It has been applied in OL2R in [19]. Almost no additional computation is needed in ϵ-greedy for exploration, but the exploration is also independent from the current model estimation and therefore can hardly be optimal in practice. More advanced randomization-based exploration strategies are built on the bootstrapping technique in statistics. Giro [30] explores by updating a model with a bootstrapped sample of its history with pseudo reward. In [29, 32], random noise is added to the observed feedback for the model training to achieve the purpose of exploration in model's output. Such a strategy is proved to be effective in both linear and generalized linear model training. Most recently, Jia et al. [25] proved randomization can also be used for online neural network learning. The most closely related work to our study is [4, 21], where an ensemble of models are trained to approximate the confidence interval for the purpose of exploration in online model update.

3 METHOD

In this section, we provide a brief introduction of the general problem setting in OL2R, and then present our proposed solution for scalable exploration in neural OL2R.

3.1 Problem Formulation

In OL2R, a ranker directly learns from the interactions with users for T rounds. At each round $t = 1, ..., T$, the ranker receives a query q_t and its associated V_t candidate documents represented as a set of d-dimensional query-document feature vectors, $\mathcal{X}_t = \{\mathbf{x}_1^t, \mathbf{x}_2^t, \dots \mathbf{x}_{V_t}^t\}$ with $\mathbf{x}_i^t \in \mathbb{R}^d$, and we assume that $\|\mathbf{x}_i^t\| \leq u$. Once the query is received, the ranker determines the ranking of the V_t documents based on its knowledge about the documents' relevance so far. We denote $\pi_t = (\pi_t(1), ..., \pi_t(V_t)) \in \Pi([V_t])$ as the ranking of the L_t documents, where $\pi_t(i)$ represents the rank position of document i given query q_t, and $\Pi([V_t])$ is the set of all permutations of the V_t documents. After the ranked list is returned to the user, the user examines the list and provides her click feedback $C_t = \{c_1^t, c_2^t, ..., c_{V_t}^t\}$, where $c_i^t = 1$ if the user clicks on document i at round t; otherwise $c_i^t = 0$. The ranker updates itself according to the feedback and proceeds to the next query.

Existing studies have repeatedly demonstrated that C_t is biased and noisy [2, 10, 26, 27]. Users tend to click more on the top-ranked documents, which is known as the *position bias*; and as users can only interact with the documents shown to them, the ranker only has partial observations about relevance feedback from user clicks, which is known as the *presentation bias*. Therefore, a good OL2R solution needs to carefully deal with the biased implicit feedback and effectively explore the unknowns for improved relevance estimation on the one hand, and serve users with the currently best estimated ranked result on the other hand.

In this work, we follow the standard practice and treat clicks as relative preference feedback [26]. More specifically, the clicked documents are assumed to be preferred over those examined but unclicked documents. Besides, we consider every document that precedes a clicked document and the first l subsequent unclicked document as examined. Such an assumption is widely adopted and proved to be effective in both offline and online learning to rank [2, 24, 40, 48]. In particular, we denote o_t as the index of the last examined position in the ranked list π_t at round t.

Different from offline learning to rank, OL2R needs to serve the users while learning from its presented rankings. Therefore cumulative regret is an important metric for evaluating OL2R. In this work, we follow the regret defined as the number of mis-ordered pairs from the presented ranking to the ideal one [23, 24, 33],

$$R_T = \mathbb{E}\big[\sum\nolimits_{t=1}^T r_t\big] = \mathbb{E}\big[\sum\nolimits_{t=1}^T K(\pi_t, \pi_t^*)\big]$$

where π_t is the ranked list generated by the current ranker, π_t^* is the optimal ranking for the current query, and $K(\pi_t, \pi_t^*) = \Big|\big\{(i,j) : i < j, \big(\pi_t(i) < \pi_t(j) \wedge \pi_t^*(i) > \pi_t^*(j)\big) \vee \big(\pi_t(i) > \tau_t(j) \wedge \pi_t^*(i) < \pi_t^*(j)\big)\big\}\Big|$. Such a pairwise regret definition directly connects an OL2R algorithm's online performance with classical ranking evaluations as most ranking metrics, such as ARP and NDCG can be decomposed into pairwise comparisons [50].

3.2 Exploration in olRankNet

In the recent decade, DNNs have demonstrated powerful representation learning capacity and significantly boosted the performance for a wide variety of machine learning tasks [14, 35], including learning to rank [13, 17, 20, 45]. OL2R, on the other hand, has received limited benefit from the advances in DNNs. While DNNs are generally more accurate at predicting a document's relevance under a given query (i.e., exploitation), creating practical strategies to balance exploration and exploitation for neural ranking models in sophisticated online learning scenarios is challenging.

Built on the advances in renewed understandings about the generalization of DNNs, olRankNet extends PairRank with a neural scoring function and demonstrates the best empirical performance with theoretical guarantees [23]. olRankNet directly learns a neural ranker from users' implicit feedback with a fully connected neural network $f(\mathbf{x};\boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_L\phi(\mathbf{W}_{L-1}\phi(\dots\phi(\mathbf{W}_1\mathbf{x})))$, where depth $L \geq 2$, $\phi(\mathbf{x}) = \max\{\mathbf{x}, 0\}$, and $\mathbf{W}_1 \in \mathbb{R}^{m\times d}$, $\mathbf{W}_i \in \mathbb{R}^{m\times m}$, $2 \leq i \leq L-1$, $\mathbf{W}_L \in \mathbb{R}^{m\times 1}$, and $\boldsymbol{\theta} = [\text{vec}(\mathbf{W}_1)^\top, \dots, \text{vec}(\mathbf{W}_L)^\top]^\top \in \mathbb{R}^p$ with $p = m+md+m^2(L-2)$. At each round, the model $\boldsymbol{\theta}_t$ is updated by optimizing the cross-entropy loss between the predicted pairwise relevance distribution on all documents and those inferred from user feedback till round t with a ℓ_2-regularization term centered at the randomly initialized parameter $\boldsymbol{\theta}_0$:

$$\mathcal{L}_t(\boldsymbol{\theta}) = \sum\nolimits_{s=1}^t \sum\nolimits_{(i,j)\in\Omega_s} -(1-\mathbf{y}_{ij}^s)\log\big(1-\sigma(f_{ij})\big) - \mathbf{y}_{ij}^s \log\big(\sigma(f_{ij})\big) + m\lambda/2\|\boldsymbol{\theta}-\boldsymbol{\theta}_0\|^2, \quad (3.1)$$

where $f_{ij}^t = f(\mathbf{x}_i;\boldsymbol{\theta}_{t-1}) - f(\mathbf{x}_j;\boldsymbol{\theta}_{t-1})$ is the difference between the estimated ranking scores of document i and j, λ is the ℓ_2-regularization coefficient, Ω_s denotes the set of document pairs that received different click feedback at round s, i.e. $\Omega_s = \{(i,j) : c_i^s \neq c_j^s, \forall\tau_s(i) \leq \tau_s(j) \leq o_t\}$, $\mathbf{y}_{ij}^s$ indicates whether document i is preferred over document j based on the click feedback, i.e., $\mathbf{y}_{ij}^s = (c_i^s - c_j^s + 1)/2$ [6].

However, there is uncertainty in the estimated model $\boldsymbol{\theta}_t$ due to the click noise, i.e., $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\| \neq 0$, where $\boldsymbol{\theta}^*$ is assumed to be the underlying ground-truth model parameter. And therefore the model's output ranking might be wrong because of this uncertainty. olRankNet decides to randomize its output document rankings where its estimation is still uncertain, which helps collect unbiased feedback for improved model estimation subsequently. Based on the neural tangent kernel technique, the uncertainty of olRankNet's estimated pairwise rank order can be analytically quantified and upper bounded with a high probability, under the assumption that pairwise click noise follows a R-sub-distribution [23]. This is described in the following lemma.

Lemma 3.1. (Confidence Interval of Pairwise Rank Order in olRankNet). There exist positive constants C_1 and C_2 such that for any $\delta_1 \in (0,1)$, with satisfied step size of gradient descent η, and the neural network width m, at round $t < T$, for any document pair

(i, j) under query q_t, with probability at least $1 - \delta_1$,

$$|\sigma(f_{ij}^t) - \sigma(f_{ij}^*)| \leq \alpha_t \|\mathbf{g}_{ij}^t/\sqrt{m}\|_{\mathbf{A}_t^{-1}} + \epsilon(m), \tag{3.2}$$

where $\epsilon(m)$ is the approximation error from gradient descent in neural network optimization, f_{ij}^* is the ground-truth difference between document pair (i, j), $\mathbf{g}_{ij}^s = \mathbf{g}(\mathbf{x}_i; \boldsymbol{\theta}_s) - \mathbf{g}(\mathbf{x}_j; \boldsymbol{\theta}_s)$ with $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta})$ as the gradient of input $\mathbf{x}$ with respect to the entire network parameters, $\alpha_t = \bar{C}_1\left(\sqrt{R^2 \log(\det(\mathbf{A}_t)/\delta_1^2 \det(\lambda \mathbf{I}))} + \sqrt{\lambda}\bar{C}_2\right)$ with $\bar{C}_1$ and $\bar{C}_2$ as positive constants, $\mathbf{A}_t = \sum_{s=1}^{t-1}\sum_{(i',j')\in\Omega_s} \frac{1}{m}\mathbf{g}_{i',j'}^s \mathbf{g}_{i',j'}^{s\ \top} + \lambda \mathbf{I}$.

With the constructed confidence interval of the estimated pairwise document rank order, olRankNet separates all the candidate document pairs into two sets, certain rank order S_t^c and uncertain rank order S_t^u. S_t^c contains the document pairs where with high probability the estimated pairwise order is correct. For example, for document i and j, if the lower confidence bound of the probability that i is better than j, i.e., $\sigma(f_{ij}^t) - \alpha_t \|\mathbf{g}_{ij}^t/\sqrt{m}\|_{\mathbf{A}_t^{-1}} - \epsilon(m)$, is larger than 0.5, then (i, j) belongs to S_t^c. Otherwise, the pair belongs to S_t^u, which indicates that the predicted rank order can still be wrong.

When constructing the ranked list, olRankNet first builds a ranking graph with all the candidate documents and the certain rank orders in S_t^c. Then topological sort is performed, where the certain rank orders will be followed (i.e., exploitation), and uncertain rank orders will be randomized (i.e., exploration). With such a ranking strategy, olRankNet is proved to have an $O(\log^2(T))$ cumulative pairwise regret upper bound.

Although olRankNet has a strong theoretical foundation, its scalability is severely limited due to the additional computation required for constructing the confidence interval. In particular, the covariance matrix $\mathbf{A}$ in Lemma 3.1 is constructed with the gradient of the scoring function with respect to the network parameters, of which the size p is very large. In order to construct the confidence interval according to Eq (3.2), the inverse of the covariance matrix $\mathbf{A}$ has to be computed whenever the model is updated, which results in an unacceptably high computational cost (around $O(p^3)$). As a consequence, it is practically impossible for olRankNet to be exactly executed. In [23], approximation is employed to make olRankNet operational in practice, e.g., only using the diagonal of $\mathbf{A}$. However, there is no theoretical guarantee for such an approximation, which unfortunately breaks the theoretical promise of olRankNet and directly leads to an unknown gap between its theoretical and empirical performance.

3.3 Scalable Exploration with Perturbed Feedback

To bridge the gap, we develop an efficient and scalable strategy for recognizing the certain and uncertain rank orders without explicitly constructing the confidence set. And our basic idea is to leverage the bootstrapping technique to create randomness in a neural ranker's output. In particular, at each round, we perturb the entire user feedback history for N times with noise freshly and independently sampled from a zero-mean Gaussian distribution, and train the corresponding neural ranker as usual. Denote model $\boldsymbol{\theta}^{(n)}$ for $n \in [N]$ as the solution of minimizing the following objective function

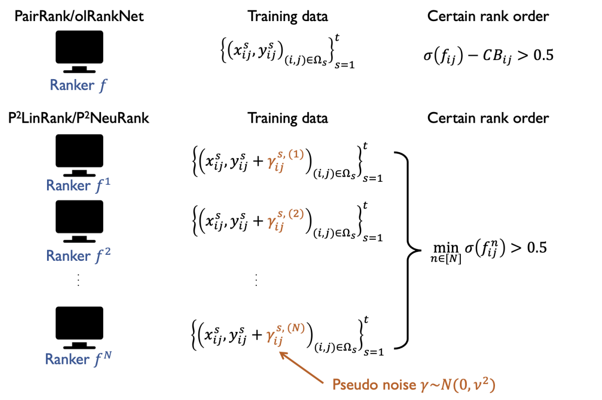

PairRank/olRankNet
Training data
Certain rank order
$\left\{(x_{ij}^s, y_{ij}^s)_{(i,j)\in\Omega_s}\right\}_{s=1}^t$
$\sigma(f_{ij}) - CB_{ij} > 0.5$
Ranker f
P²LinRank/P²NeuRank
Training data
Certain rank order
Ranker f^1
$\left\{(x_{ij}^s, y_{ij}^s + \gamma_{ij}^{s,(1)})_{(i,j)\in\Omega_s}\right\}_{s=1}^t$
Ranker f^2
$\left\{(x_{ij}^s, y_{ij}^s + \gamma_{ij}^{s,(2)})_{(i,j)\in\Omega_s}\right\}_{s=1}^t$
$\min_{n\in[N]} \sigma(f_{ij}^n) > 0.5$
Ranker f^N
$\left\{(x_{ij}^s, y_{ij}^s + \gamma_{ij}^{s,(N)})_{(i,j)\in\Omega_s}\right\}_{s=1}^t$
Pseudo noise $\gamma \sim N(0, \nu^2)$

Figure 1: Comparison between PairRank/olRankNet and P^{2}LinRank/P^{2}NeuRank

with gradient descent,

$$\begin{aligned}\boldsymbol{\theta}^{(n)} = \min \sum_{s=1}^{t} \sum_{(i,j)\in\Omega_s} &-\left(1 - (\mathbf{y}_{ij}^s + \gamma_{ij}^{s,(n)})\right) \log\left(1 - \sigma(f_{ij})\right) \\ &- (\mathbf{y}_{ij}^s + \gamma_{ij}^{s,(n)}) \log\left(\sigma(f_{ij})\right) + m\lambda/2\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2,\end{aligned} \tag{3.3}$$

where $\{\gamma_{ij}^{s,(n)}\}_{s=1}^t \sim \mathcal{N}(0, \nu^2)$ are Gaussian random variables that are independently sampled in each round t, and ν is a hyper-parameter that controls the strength of perturbation (and thus the exploration) in P^{2}NeurRank.

The detailed procedure of P^{2}NeurRank is given in Algorithm 1. The algorithm starts by initializing the N neural rankers. At each round of interaction, given a query q_t, for each pair of candidate documents, N parallel predictions about their rank order will be generated by the set of neural rankers. If all the N estimations give the same prediction about the document pair's rank order, e.g., $i \succ j$ for document i and document j, then (i, j) is considered as a certain rank order (line 9 - line 13 in Algorithm 1). Otherwise, the relation between these two documents is still uncertain and further exploration is needed there when generating the ranked list. Once the sets of certain and uncertain rank orders are determined, we follow the same procedure of olRankNet to generate the ranked list via topological sort with respect to the certain rank orders.

The key intuition for P^{2}NeurRank is to utilize the variance introduced in the randomly perturbed click feedback to encourage exploration. With the injected perturbation, there are two kinds of deviations existing in the estimated pairwise preference in each of the N parallel neural rankers: 1) the deviation caused by the observation noise introduced by the click feedback; 2) the deviation caused by the added perturbations. By properly setting the variance parameter ν for the added perturbation, the corresponding deviation will introduce enough randomness in the model estimation. For each round of interaction, we maintain N models and with high probability, the minimum of the estimated pairwise preference serves as the pessimistic estimate of the preference.

Compared to olRankNet, which requires to maintain the inverse of the covariance matrix, P^{2}NeurRank does not need any added computation for the purpose of exploration, besides the regular neural network updates. As a result, P^{2}NeurRank greatly alleviates the computation burden in neural OL2R. More specifically, in each round, the online neural ranking algorithm generally takes the following steps: (1) predict the rank order in each document pair, (2) generate the ranked list by topology sort via the constructed certain and uncertain rank orders, and (3) update the model according to the newly received click feedback. With p representing the total number of parameters in a neural ranker, olRankNet has the time complexity $O(V_t p + V_t^2)$ for the first step. As the N neural models in P^{2}NeurRank are independent from each other, the time complexity of P^{2}NeurRank in the first step is also $O(V_t p + V_t^2)$ by executing the N ranker's predictions in parallel. For the third step, again by the training the N neural rankers in parallel using gradient descent, both P^{2}NeurRank and olRankNet have the time complexity of $O(\tau p \sum_{s=1}^{t} |\Omega_s|)$ where τ is the number of epochs for training the neural network. The key difference lies in the second step. olRankNet requires the inverse of covariance matrix, which has the time complexity at least $O(p^{2.2373})$. Besides, constructing the confidence interval for all the document pairs has the time complexity of $O(V_t^2 p^2)$. While for P^{2}NeurRank, finding the minimum of the N predictions for all the document pairs costs $O(NV_t^2)$. Once the certain and uncertain rank orders are determined, both algorithms require $O(V_t + E_t)$ for the topological sort, where E_t represents the number of certain rank orders and $E_t \leq V_t$. Therefore, for the second step, olRankNet has the total time complexity as $O(p^{2.2373} + V_t^2 p^2 + V_t + E_t) = O(p^{2.2373} + V_t^2 p^2)$, while P^{2}NeurRank has the time complexity as $O(NV_t^2 + V_t + E_t) = O(NV_t^2)$. As p is oftentimes in the order of tens of thousands (if not less), P^{2}NeurRank greatly reduces the time required for performing exploration in neural OL2R. And also empirically, the number of parallel rankers N in P^{2}NeurRank does not need to be large. For example, in our experiments, we found $N = 2$ already led to promising performance of P^{2}NeurRank comparing to olRankNet.

We want to highlight that our proposed perturbation-based exploration strategy can also be applied to linear ranking models, e.g., PairRank [24]. The procedure is almost the same as described in Algorithm 1, and so is its computational advantage in linear models, especially when the dimension of the feature vectors is large. In our experiments, we empirically evaluated our perturbation-based method in PairRank (named P^{2}LinRank) and observed its expected performance and computational advantages.

4 REGRET ANALYSIS

In this section, we provide the regret analysis of the proposed exploration strategy. For better readibility, we present the analysis of a linear ranker. According to the anlaysis in [23], under the neural tangent technique and the convergence analysis of the gradient descent in neural network optimization, the linear analysis can be readily applied to the neural ranker. And we discuss the difference between the analysis between the linear ranker and neural ranker in the appendix.

Follow the standard assumption in [23, 24], we assume that on the *examined* documents where $\pi_t(i) \leq o_t$, the obtained feedback C_t is independent from each other given the *true relevance* of documents, so is their noise [15, 16, 26]. Therefore, the noise in the inferred preference pair becomes the sum of noise from the clicks

Algorithm 1 P^{2}NeurRank

```
1:  Input: Number of rounds $T$, regularization coefficient $\lambda$, perturbation parameter $\nu$, network width $m$, network depth $L$, and number of rankers $N$.
2:  Initialize $N$ neural network models $\{\theta_0^n\}_{n=1}^N$ with $m$ and $L$
3:  for t = 1, ..., T do
4:      $S_t^c = \emptyset, S_t^u = \emptyset$
5:      $q_t \leftarrow$ receive_query(t)
6:      $\mathcal{X}_t = \{\mathbf{x}_1^t, \mathbf{x}_2^t, \dots \mathbf{x}_{V_t}^t\} \leftarrow$ retrieve_candidate_documents($q_t$)
7:      for each document pair $(i, j) \in [V_t]^2$ do
8:          $\{\sigma(f_{ij,t}^n)\}_{n=1}^N \leftarrow$ get_N_estimations($\mathbf{x}_i^t, \mathbf{x}_j^t, \{\theta_t^n\}_{n=1}^N$)
9:          if $\min_{n\in[N]} \sigma(f_{ij,t}^n) > 1/2$ or $\max_{n\in[N]} \sigma(f_{ij,t}^n) < 1/2$ then
10:             $S_t^c \leftarrow S_t^c \cup (i, j)$
11:         else
12:             $S_t^u \leftarrow S_t^u \cup (i, j)$
13:         end if
14:     end for
15:     $\pi_t \leftarrow$ topological_sort($S_t^c, S_t^u$)
16:     $C_t \leftarrow$ collect_click_feedback($\pi_t$)
17:     $\Omega_t, \{y_{ij}\}_{(i,j)\in\Omega_t} \leftarrow$ construct_training_data($C_t$)
18:     for $n = 1, ..., N$ do
19:         Generate $\{\{\gamma_{ij}\}_{(i,j)\in\Omega_s}\}_{s=1}^t \sim \mathcal{N}(0, \nu^2)$
20:         Set $\theta_t^n$ by the output of gradient descent for solving Eq (3.3) with $\{\Omega_s\}_{s=1}^t$.
21:     end for
22: end for
```

in the two associated documents. And we also only use the independent pairs to construct Ω_t as suggested in PairRank and olRankNet. Thus, the pairwise noise satisfies the following proposition.

Proposition 4.1. For any $t \geq 1, \forall (i, j) \in \Omega_t$, the pairwise feedback follows $y_{ij}^t = \sigma\big(f(\mathbf{x}_i; \theta^*) - f(\mathbf{x}_j; \theta^*)\big) + \epsilon_{ij}^t$, where ϵ_{ij}^t satisfies that for all $\beta \in \mathbb{R}$, $\mathbb{E}\Big[\exp(\beta\epsilon_{ij}^t)|\big\{\{\epsilon_{i',j'}^s\}_{i',j'\in\Omega_s}\big\}_{s=1}^{t-1}, \Omega_{1:t-1}\Big] \leq \exp(\beta^2 R^2)$, is an R-sub-Gaussian random variable.

To train a linear ranker, we have the scoring function $f(\mathbf{x}; \theta) = \mathbf{x}^\top \theta$. And we assume that $\|\mathbf{x}\| \leq P$ and $\|\theta\| \leq Q$. The loss function can be rewritten as,

$$\mathcal{L}_t^{(n)}(\theta) = \sum_{s=-d+1}^{t} \sum_{(i,j)\in\Omega_s} -\big(\mathrm{y}_{ij}^s + \gamma_{ij}^{s,(n)}\big) \log(\sigma(\mathbf{x}_{ij}^{s\top}\theta)) - \big(1 - (\mathrm{y}_{ij}^s + \gamma_{ij}^{s,(n)})\big) \log\big(1 - \sigma(\mathbf{x}_{ij}^{s\top}\theta)\big), \tag{4.1}$$

where $\mathbf{x}_{ij}^s = \mathbf{x}_i^s - \mathbf{x}_j^s$ is the difference between the feature vectors of document i and j, and d is the dimension of the feature vectors. With $|\Omega_s| = 1$, $\mathbf{x}_{ij}^s = \sqrt{\lambda}\mathbf{e}_i$, $\mathrm{y}_{ij}^s = 0$ for $s \in [-d + 1, 0]$, this loss function can be interpreted as adding l_2 regularization to the cross-entropy loss.

Given this objective function is log-convex with respect to θ, its solution $\widehat{\theta}_t^{(n)}$ of ranker n for $n \in [N]$ is unique under the following estimation method at each round t,

$$\sum_{s=-d+1}^{t-1} \sum_{(i,j)\in\Omega_s} \Big(\sigma(\mathbf{x}_{ij}^{s\top}\theta) - (\mathrm{y}_{ij}^s + \gamma_{ij}^{s,(n)})\Big) \mathbf{x}_{ij}^s + \lambda\theta = 0 \tag{4.2}$$

Let $g_t(\boldsymbol{\theta}) = \sum_{s=-d+1}^{t-1} \sum_{(i,j)\in\Omega_s} \sigma(\mathbf{x}_{ij}^{s\top}\boldsymbol{\theta})\mathbf{x}_{ij}^s + \lambda\boldsymbol{\theta}$ be the invertible function such that the estimated parameter $\widehat{\boldsymbol{\theta}}_t^{(n)}$ satisfies $g_t(\widehat{\boldsymbol{\theta}}_t^{(n)}) = \sum_{s=-d+1}^{t-1} \sum_{(i,j)\in\Omega_s} (\mathbf{y}_{ij}^s + \gamma_{ij}^{s,(n)})\mathbf{x}_{ij}^s$. As discussed before, there are two kinds of deviations inside this estimation $\widehat{\boldsymbol{\theta}}_t^{(n)}$. To analyze their effect in the estimation, we introduce an auxiliary solution $\bar{\boldsymbol{\theta}}_t$ for solving the linear objective function, which satisfies $g_t(\bar{\boldsymbol{\theta}}_t) = \sum_{s=1}^{t-1} \sum_{(i,j)\in\Omega_s} \mathbf{y}_{ij}^s \mathbf{x}_{ij}^s$. Then, for the two solutions $\widehat{\boldsymbol{\theta}}_t^{(n)}$ and $\bar{\boldsymbol{\theta}}_t$, we have the following lemmas quantifying the deviations in the estimation.

Lemma 4.2. (Deviation from observation noise). At round $t < T$, for any pair of document $(\mathbf{x}_i^t, \mathbf{x}_j^t)$ under query q_t, with probability at least $1-\delta$, we have,

$$|\sigma(\mathbf{x}_{ij}^{t\top}\bar{\boldsymbol{\theta}}_t) - \sigma(\mathbf{x}_{ij}^{t\top}\boldsymbol{\theta}_t^*)| \leq \alpha_t \|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}},$$

where $\alpha_t = (2k_\mu/c_\mu)\left(\sqrt{R^2\log(\det(\mathbf{A}_t)/(\delta^2\det(\lambda\mathbf{I})))} + d\right)$, $\mathbf{A}_t = \lambda\mathbf{I} + \sum_{s=1}^{t-1}\sum_{(i',j')\in\Omega_s} \mathbf{x}_{i',j'}\mathbf{x}_{i',j'}^\top$, k_μ is the Lipschitz constant of the sigmoid link function σ, $c_\mu = \inf_{\boldsymbol{\theta}\in\Theta}\dot{\sigma}(\mathbf{x}^\top\boldsymbol{\theta})$, with $\dot{\sigma}$ as the first derivative of σ.

Accordingly, we define E_t as the success event at round t:

$$E_t = \left\{\forall (i,j)\in[V_t]^2, |\sigma(\mathbf{x}_{ij}^{t\top}\bar{\boldsymbol{\theta}}_t) - \sigma(\mathbf{x}_{ij}^{t\top}\boldsymbol{\theta}^*)| \leq \alpha_t\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}}\right\}.$$

Intuitively, E_t is the event that the auxiliary solution $\bar{\boldsymbol{\theta}}_t$ is "close" to the optimal model $\boldsymbol{\theta}^*$ at round t.

As discussed before, we define the regret as the number of misordered pairs. Therefore, the key step in regret analysis is to quantify the probability that an estimated preference is uncertain. According to Algorithm 1, the certain rank order in the perturbed pairwise ranker is defined as follows,

Definition 4.3. (Certain Rank Order) At round t, the rank order between documents $(i,j)\in[V_t]^2$ belongs to the set of certain rank orders ω_t^c if and only if $\min_{n\in[N]}\sigma\left(f_{ij}^{t,(n)}\right) > \frac{1}{2}$ or $\max_{n\in[N]}\sigma\left(f_{ij}^{t,(n)}\right) < \frac{1}{2}$; otherwise, $(i,j)\in\omega_t^u$.

According to the definition, and the deviations caused by the observation noise and the pseudo noise, we have the following lemma quantifying the probability of an estimation being uncertain.

Lemma 4.4. There exist positive constants c, C_1 and C_2, that with $t' = \left(\frac{C_1\sqrt{d}+C_2\sqrt{\log(1/\delta)}+(P^2Rk_\mu)/(\sqrt{\lambda}c_\mu\Delta_{\min})}{\lambda_{\min}(\Sigma)}\right)^2 + \frac{2k_\mu P}{c_\mu\Delta_{\min}}\left(\sqrt{R^2\log(1/\delta)} + \sqrt{\lambda}Q\right)$, $\delta\in(0,1)$, for round $t\geq t'$, with probability at least $1-\delta$, event E_t holds with α_t defined in Lemma 4.2, with $N\geq\log\delta/\log(1-\exp(-\beta^2)/(4\sqrt{\pi}\beta))$, where $\beta = \frac{k_\mu^2\alpha_t^2}{c_\mu^2\nu^2}$, for a document pair (i,j) that $i\succ j$ for the given query, the probability that the estimated pairwise preference is uncertain is upper bounded as $\mathbb{P}\left((i,j)\in\omega_t^u\right)\leq\frac{2N\nu^2k_\mu^2\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}}^2}{c_\mu^2c^2\Delta_{\min}^2}$, where $\Delta_{\min} = \min_{t\in T,(i,j)\in[V_t]^2}|\sigma(\mathbf{x}_{ij}^{t\top}\boldsymbol{\theta}^*) - \frac{1}{2}|$ representing the smallest gap of pairwise difference between any pair of documents associated to the same query over time (across all queries).

The detailed proof is provided in the appendix. This lemma provides the upper bound of the probability that an estimated pairwise preference is uncertain. The key idea is to analyze the concentration and anti-concentration property of the deviation caused by the pseudo noise. In particular, the deviation caused by the pseudo noise γ, and the ensemble of N rankers should be sufficiently large so that for document pairs (i,j), the maximum estimated pairwise preference, $\max_{n\in[N]}\sigma(\mathbf{x}_{ij}^{t\top}\widehat{\boldsymbol{\theta}}_t^{(n)})$ is optimism to trigger exploration. On the other hand, with more observations, the probability of being uncertain will be shrinking with the concentration property of the pseudo noise.

Following the assumption in [23, 24, 31], denote p_t as the probability that the user examines all documents in τ_t at round t, and let $p^* = \min_{1\leq t\leq T} p_t$ be the minimal probability that all documents in a query are examined over time. The regret of the proposed model can be upper bounded as follows.

Theorem 4.5. Assume pairwise query-document feature vector $\mathbf{x}_{ij}^t$ under query q_t, where $(i,j)\in[V_t]^2$ and $t\in[T]$, satisfies Proposition 1. With $\delta\in(0,1)$, with probability at least $1-\delta$, the T-step regret of the proposed model is upper bounded by:

$$R_T \leq R' + \frac{1}{p^*}2dV_{\max}C\log\left(1+\frac{o_{\max}TP^2}{2d\lambda}\right)$$

where $R' = t'V_{\max}$, $V_{\max}$ represents the maximum number of document associated with the same query over time, and t' is defined in Lemma 4.4, and $w = \sum_{s=t'}^{T}\left((V_{\max}^2 - 2V_{\max})P^2/\lambda_{\min}(\mathbf{A}_s)\right)$, and By choosing $\delta_1 = \delta_2 = 1/T$, we have the expected regret at most $R_T \leq O(d\log^2(T))$.

We provide the detailed proof in the appendix. According to the pairwise exploration strategy, the regret only comes from the document pairs that are uncertain, e.g., random shuffling will be conducted to perform the exploration. With the quantified uncertain probability in Lemma 4.4, the pairwise regret can be upper bounded accordingly.

In neural rankers, the neural network approximation error should be considered in addition to the deviations caused by the noise. According to the analysis in [25], the variance of the added noise should be set according to the deviations caused by both the observation noise and the approximation error. Based on the theoretical analysis in [23], by properly setting the width of the neural network and the step size of gradient descent, the model with a neural ranker will still have a sublinear regret.

5 EXPERIMENT

In this section, we empirically compare our proposed model with an extensive list of state-of-the-art OL2R algorithms on two large public learning to rank benchmark datasets.

5.1 Experiment Setup

5.1.1 Dataset. We experimented on the Yahoo! Learning to Rank Challenge dataset [9], which consists of 292,921 queries and 709,877 documents represented by 700 ranking features, and MSLR-WEB10K [42], which contains 10,000 queries, each having 125 documents on average represented by 136 ranking features. Both datasets are labeled on a five-grade relevance scale: from not relevant (0) to perfectly relevant (4). These two datasets are the most popularly used in literature for evaluating OL2R algorithms. We followed the train/test/validation split provided in the datasets to perform the cross-validation to make our results comparable to previously reported results.

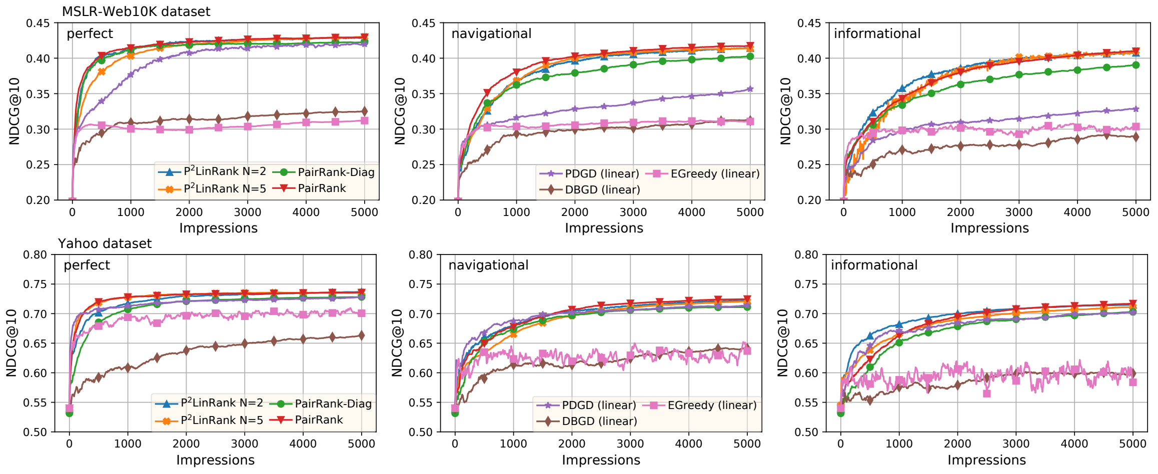


Figure 2: Offline performance of linear OL2R on two benchmark datasets.

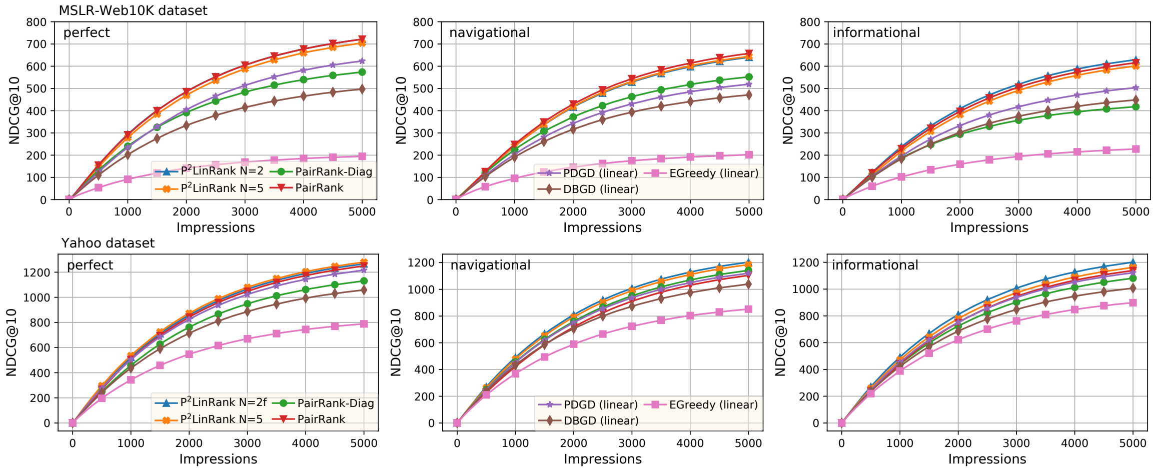


Figure 3: Online performance of linear OL2R on two benchmark datasets.

5.1.2 User interaction simulation. A standard simulation setting for the user clicks is adopted in our experiment [24, 40], which is the most popularly used procedure for OL2R evaluations. At each round, a query is uniformly sampled from the training set. The model will then generate a ranked list and return it to the user. Dependent click model (DCM) [16] is applied to simulate user behaviors, which assumes that the user will sequentially scan the ranked list and make click decisions on the examined documents. In DCM, the probabilities of clicking on a given document and stopping the subsequent examination are both conditioned on the document's true relevance label. Three different model configurations are employed in our experiments to represent three different types of users. The details are shown in Table 1. In particular, *perfect* users will click on all relevant documents and do not stop browsing until the last returned document; *navigational* users are very likely to click on the first encountered highly relevant document and stop there; and *informational* users tend to examine more documents, but sometimes click on irrelevant documents, which contribute a significant amount of noise in the click feedback. To reflect the presentation bias, all the models only return the top 10 ranked results.

Table 1: Configuration of simulated click models.

	Click Probability					Stop Probability				
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

5.1.3 Baselines. We list the OL2R solutions used for our empirical comparisons below. We performed the experiments on both linear and neural rankers to show the general effectiveness of our proposed exploration strategy. For convenience of reference, in the experiment discussion, we name our model with a linear ranker as P^{2}LinRank

- **ϵ-Greedy (linear and neural)** [19]: At each rank position from top to bottom, it randomly samples an unranked document with probability ϵ or selects the next best document based on the currently learned ranker.

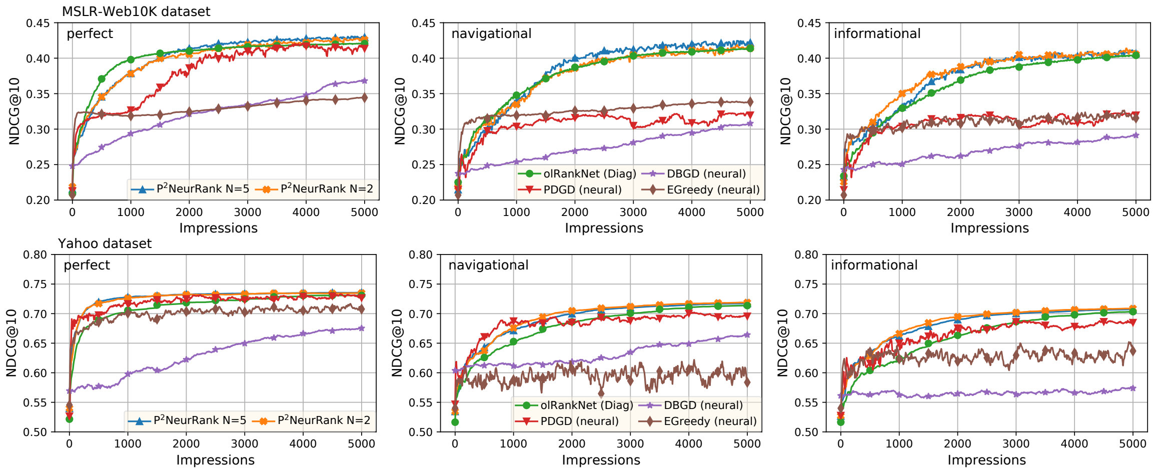


Figure 4: Offline performance of neural OL2R on two benchmark datasets.

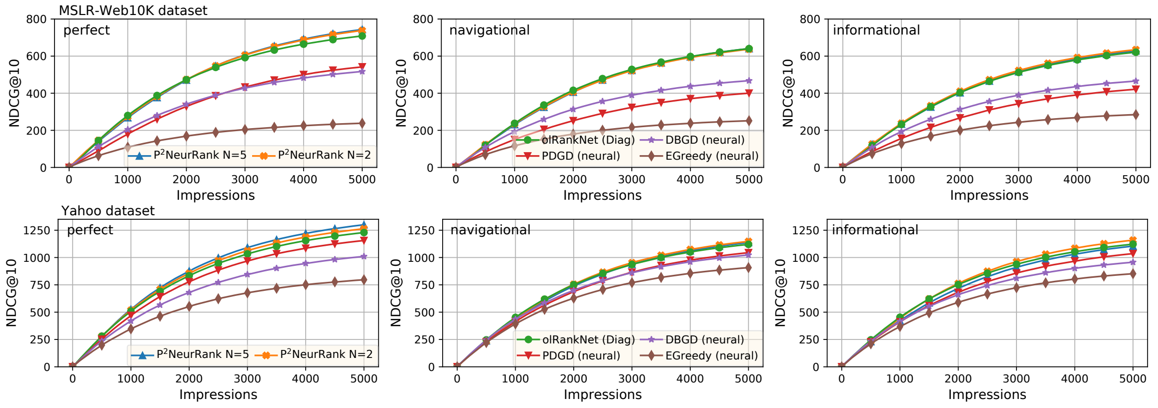
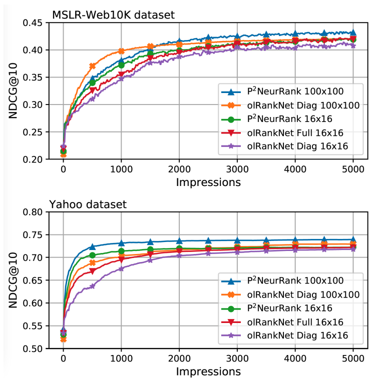


Figure 5: Online performance of neural OL2R on two benchmark datasets.

- **DBGD (linear and neural)** [54]: DBGD uniformly samples a direction in the entire model space for exploration and model update. Its convergence is only proved for linear rankers, though empirically previous studies also applied it to neural rankers.
- **PDGD (linear and neural)** [40]: PDGD samples the next ranked document from a Plackett-Luce model and estimates gradients from the inferred pairwise preferences. The only known theoretical property about PDGD is its estimated gradient is unbiased, but how this unbiased gradient leads model training is still unknown.
- **PairRank** [24]: PairRank learns a pairwise logistic regression ranker online and explores by divide-and-conquer in the pairwise document ranking space. The training of PairRank is known to converge with a sublinear regret defined on the cumulative number of misordered document pairs.
- **PairRank-Diag**: This is a variant of PairRank where the diagonal approximation of its covariance matrix is used to calculate the required confidence interval.
- **olRankNet** [23]: olRankNet is an extension of PairRank with a neural ranker, where the confidence interval is constructed via the neural tangent kernel. It inherits good theoretical properties from PairRank.
- **olRankNet-Diag**: This is a variant of olRankNet, where the diagonal approximation of the covariance matrix is applied to calculate the confidence interval.

5.1.4 Hyper-Parameter Tuning. MSLR-WEB10K dataset is equally partitioned into five folds, while Yahoo Learning to Rank dataset is equally partitioned into two folds. We performed cross validation on each dataset. For each fold, the models are trained on the training set, and the hyper-parameters are tuned based on the performance on the validation set.

In our experiments, for all the neural rankers, a two layer neural network with width $m = 100$ is applied. We did a grid search in olRankNet, PairRank, PairRank-Diag for its regularization parameter λ over $\{10^{-i}\}_{i=1}^{4}$, exploration parameter α over $\{10^{-i}\}_{i=1}^{4}$. For parameter estimation in all neural rankers, we did a grid search for learning rate over $\{10^{-i}\}_{i=1}^{3}$ for gradient descent. PairRank and PairRank-Diag are directly optimized with L-BFGS. The model update in PDGD and DBGD is based on the optimal settings in their original paper (i.e., the exploration step size and learning rate). The hyper-parameters for PDGD and DBGD are the update learning



Model	NDCG@10
P^{2}NeurRank 100x100	0.4318
olRankNet Diag 100x100	0.4201
P^{2}NeurRank 16x16	0.4189
olRankNet Full 16x16	0.4195
olRankNet Diag 16x16	0.4073

Model	NDCG@10
P^{2}NeurRank 100x100	0.7392
olRankNet Diag 100x100	0.7294
P^{2}NeurRank 16x16	0.7223
olRankNet Full 16x16	0.7217
olRankNet Diag 16x16	0.7179

Figure 6: Comparison among neural rankers

rate and the learning rate decay, for which we performed a grid search for learning rate over $\{10^{-i}\}_{i=1}^{3}$, and the learning rate decay is set to 0.999977. For P^{2}NeurRank and P^{2}LinRank, we have an extra hyper-parameter N, which is searched over $\{2, 5, 10\}$. We fixed the total number of iterations T to 5000. The experiments are executed for 10 times with different random seeds and the averaged results are reported in this paper.

5.1.5 Evaluations. To evaluate an OL2R model, we report both the offline and online performance during the interactions. The offline performance is evaluated in an "online" fashion where the newly updated ranker is evaluated on a hold-out testing set against its ground-truth relevance labels. This measures how fast an OL2R model improves its ranking quality. Such setting can be viewed as using one portion of the traffic for model update, while serving another portion with the latest model. NDCG@10 is used to assess the ranking performance. In addition to the offline evaluation, we also evaluate the models' online result serving quality. This reflects user experience during the interactions and thus should be seriously considered. Sacrificing user experience for model training will compromise the goal of OL2R. We adopt the cumulative Normalized Discounted Cumulative Gain to assess models' online performance. For T rounds, the cumulative NDCG is calculated as

$$\text{Cumulative NDCG} = \sum_{t=1}^{T} \text{NDCG}(\tau_t) \cdot \gamma^{(t-1)},$$

which computes the expected utility a user receives with a probability γ that he/she stops searching after each query [40]. Following the previous work [23, 24, 40, 48, 49], we set $\gamma = 0.9995$.

5.2 Experiment Results

5.2.1 Offline and online performance. We first compare our proposed model with the baselines using a linear ranker. The results are reported in Figure 2 and 3. For P^{2}LinRank, we reported the best performance with $N = 2$ and $N = 5$. We can clearly observe P^{2}LinRank maintained PairRank's strong advantage over other OL2R solutions, including ϵ-Greedy, DBGD, and PDGD, in both online and offline evaluations across three click models. It is also obvious that a straightforward perturbation of model's output, i.e., ϵ-Greedy, basically led to the worst OL2R performance, although it is often the default choice for exploration in online learning.

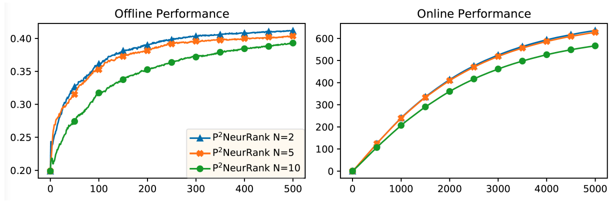


Figure 7: P^{2}NeurRank with variance $\nu^2 = 0.1$

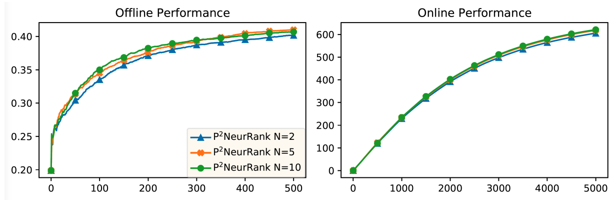


Figure 8: P^{2}NeurRank with variance $\nu^2 = 0.01$

More importantly, PairRank with a diagonal approximated covariance matrix showed serious degradation in its ranking performance, especially its online performance. For example, on MSLR-Web10K dataset, PairRank with diagonal approximation was even worse than PDGD in the online evaluation under both perfect and informational click models. This means an approximated covariance matrix cannot accurately measure the ranker's estimation uncertainty. Furthermore, this inaccuracy's impact is not deterministic: under perfect feedback, the seriously degenerated online performance together with mild decrease in offline performance suggest the model over explored; but under informational feedback, both online and offline performance dropped, which suggests insufficient estimation. This demonstrates the complication of using approximations in OL2R solutions, which loses all theoretical guarantees in the original analysis. As a result, it also strongly suggests olRankNet might not be optimal in practice, given the diagonal approximation employed to make its computation feasible. This will be demonstrated in our experiments next.

It is worth noting that P^{2}LinRank with $N = 2$ already exhibits faster convergence than PairRank, and simply increasing N does not necessarily further improve the model's performance. This result is very promising: as the only computational overhead in our perturbation-based exploration strategy is to estimate $N - 1$ additional rankers, the actual added cost in practice is minimum when $N = 2$. Similar observation is also obtained when applied to neural rankers.

In Figure 4 and 5, we report the results obtained on the neural rankers. First of all, olRankNet and P^{2}NeurRank still showed significant improvement over other OL2R solutions, including ϵ-Greedy, DBGD, MGD and PDGD. This means the pairwise exploration implemented in olRankNet is still effective for neural OL2R. The most important finding in this experiment is that P^{2}NeurRank outperformed olRankNet, where $m = 100$ for the neural network structure. As we have repeatedly mentioned, though enjoying nice theoretical advantages, in practice it is impossible to use the required full covariance matrix to compute the confidence interval in olRankNet, the diagonal approximation creates an unknown gap from its theoretical guarantee to practical performance. In Figure 6, we compare the offline performance of P^{2}NeurRank with $m = 100$, olRankNet with $m = 100$, and neural models with simpler neural structures

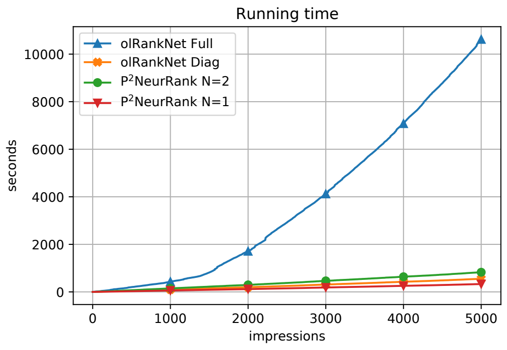


Figure 9: Efficiency comparison

under the perfect click model. The results demonstrate that for the models with 16×16 neural network, the diagonal approximation hurts the performance compared to using the full covariance matrix. As proved in our theoretical analysis, P^2NeurRank enjoys the same theoretical regret guarantee as olRankNet; but because it does not need to endure the approximation, all its nice theoretical properties are preserved in its actual implementation. Compared to the results obtained in linear models, we have good reasons to believe olRankNet with full covariance matrix could perform even better, but with a much larger (if not infeasible) computational overhead.

Again the impact from the number of parallel rankers one needs to maintain for perturbation-based exploration in P^2NeurRank is still not sensitive. As shown in both Figure 4 and 5, $N = 2$ gave us the most promising empirical performance, with the minimum added computational overhead. This is a strongly desired property for applying P^2NeurRank in practice.

5.2.2 Zoom into P^2NeurRank. In this experiment, we provide detailed analysis on P^2NeurRank. P^2NeurRank has only two hyperparameters, in addition to those inherited from olRankNet, i.e., the number of parallel rankers N and the scaled of pesudo noise ν^2. In Figure 7 and 8, we report the online and offline performance of P^2NeurRank with varying value of N for a fixed variance scale ν^2 of the added noise. We can clearly observe that with a larger noise scale, e.g., $\nu = 0.1$, setting $N = 2$ gives the best performance, comparing to $N = 5$ and $N = 10$. When the added noise scale is small, e.g., $\nu = 0.01$, setting $N = 10$ demonstrates better performance than those with fewer number of models. This indicates that the variance of the added noise ν and the number of parallel rankers N together control the exploration in P^2NeurRank. A larger variance scale, e.g., $\nu = 0.1$, together with too many models, e.g., $N = 10$, lead to more aggressive exploration and less effective model training. A smaller variance, e.g., $\nu = 0.01$, together with fewer models might not lead to sufficient exploration for model update, which also leads to worse performance. Therefore, in practice, the value of ν and N should be carefully handled to perform effective exploration. And considering the added computational overhead, using fewer parallel rankers with larger scale of added noise should be a preferred solution.

5.2.3 Efficiency comparison. In this section, we compare the running time of our proposed P^2NeurRank and the olRankNet models. We performed the experiments on a NVIDIA GeForce RTX 2080Ti graphical card. As discussed before, with complex neural networks, e.g., $m = 100$, it is impossible to perform the inverse of the full covariance matrix due to both high space and time complexity. Therefore, to compare running time, we perform the experiments on the MSLR-Web10K dataset using a simpler neural network with $m = 16$, where the perfect click model is adopted to generate the clicks. The result is reported in Figure 9. We compare the olRankNet with a full covariance matrix, olRankNet with diagonal covariance matrix, P^2NeurRank with N=2 and P^2NeurRank with N = 1. For P^2NeurRank, no extra computation is required for exploration. Therefore, the running time of P^2NeurRank with N = 1 can be viewed as the time used for the model training. We can clearly notice the big gap between the running time of olRankNet with full covariance matrix and P^2NeurRank with N = 1, which indicates the computational overhead caused by constructing the confidence interval with the full covariance matrix. Using diagonal approximation greatly reduces the total running time. However, according to our previous discussion, there is no theoretical performance guarantee for such an approximation, and our empirical results show that the diagonal approximation often leads to decreased performance in both offline and online evaluations. On the other hand, P^2NeurRank with N=2 takes slightly more time than olRankNet with diagonal approximation, while the empirical performance is significantly better (shown in Figure 2 and Figure 6). Besides, in practice, the N models can be trained in parallel, which will further reduce the running time. This demonstrate the feasibility and advantage of our proposed OL2R model in real applications.

6 CONCLUSION

In this work, we developed a provable efficient exploration strategy for neural OL2R based on bootstrapping. Previous solutions for the purpose either do not have theoretical guarantees [40, 54] or are too expensive to be exactly implemented in practice [23]. Our solution has an edge on both sides: it is proved to induce a sublinear upper regret bound counted over the number of mis-ordered pairs during online result serving, and its added computational overhead is feasible. Our extensive empirical evaluations demonstrate that our perturbation-based exploration unleashes the power of neural rankers in OL2R, with minimally added computational overhead (e.g., oftentimes only one additional ranker is needed to introduce the required exploration). And our perturbation-based exploration is general and can also be used in linear models when the input feature dimension is very large.

Our current theoretical analysis depends on gradient descent over the entire training set for model update in each round, which is still expensive and should be further optimized. We would like to investigate the possibility of more efficient model update, e.g., online stochastic gradient descent or continual learning, and the corresponding effect on model convergence and regret analysis. In addition, how to generalize our neural ranker architecture to more flexible choices, e.g., recurrent neural networks and Transformers, is another important direction of our future work.

7 ACKNOWLEDGEMENTS

We want to thank the reviewers for their insightful comments. This work is based upon work supported by National Science Foundation under grant IIS-2128019, IIS-1718216 and IIS-1553568, and Google Faculty Research Award.

REFERENCES

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*. 2312–2320.

[2] Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th ACM SIGIR*. ACM, 19–26.

[3] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. 2019. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*.

[4] Jordan T Ash, Cyril Zhang, Surbhi Goel, Akshay Krishnamurthy, and Sham Kakade. 2021. Anti-Concentrated Confidence Bonuses for Scalable Exploration. *arXiv preprint arXiv:2110.11202* (2021).

[5] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, Nov (2002), 397–422.

[6] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[7] Yuan Cao and Quanquan Gu. 2019. Generalization Bounds of Stochastic Gradient Descent for Wide and Deep Neural Networks. In *Advances in Neural Information Processing Systems*.

[8] Yuan Cao and Quanquan Gu. 2020. Generalization Error Bounds of Gradient Descent for Learning Over-parameterized Deep ReLU Networks. In *the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

[9] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*. 1–24.

[10] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale validation and analysis of interleaved search evaluation. *ACM TOIS* 30, 1 (2012), 6.

[11] Zixiang Chen, Yuan Cao, Difan Zou, and Quanquan Gu. 2019. How Much Over-parameterization Is Sufficient to Learn Deep ReLU Networks? *arXiv preprint arXiv:1911.12360* (2019).

[12] Amit Daniely. 2017. SGD learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems*. 2422–2430.

[13] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 65–74.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[15] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. 2009. Click chain model in web search. In *Proceedings of the 18th WWW*. 11–20.

[16] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient multiple-click models in web search. In *Proceedings of the 2nd WSDM*. 124–131.

[17] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2020. A deep look into neural ranking models for information retrieval. *Information Processing & Management* 57, 6 (2020), 102067.

[18] Boris Hanin and Mark Sellke. 2017. Approximating Continuous Functions by ReLU Nets of Minimal Width. *arXiv preprint arXiv:1710.11278* (2017).

[19] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval* 16, 1 (2013), 63–90.

[20] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[21] Haque Ishfaq, Qiwen Cui, Viet Nguyen, Alex Ayoub, Zhuoran Yang, Zhaoran Wang, Doina Precup, and Lin F Yang. 2021. Randomized Exploration for Reinforcement Learning with General Value Function Approximation. *arXiv preprint arXiv:2106.07841* (2021).

[22] Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*. 8571–8580.

[23] Yiling Jia and Hongning Wang. 2022. Learning Neural Ranking Models Online from Implicit User Feedback. *arXiv preprint arXiv:2201.06658* (2022).

[24] Yiling Jia, Huazheng Wang, Stephen Guo, and Hongning Wang. 2021. PairRank: Online Pairwise Learning to Rank by Divide-and-Conquer. In *Proceedings of the Web Conference 2021*. 146–157.

[25] Yiling Jia, Weitong Zhang, Dongruo Zhou, Quanquan Gu, and Hongning Wang. 2022. Learning Contextual Bandits Through Perturbed Rewards. *arXiv preprint arXiv:2201.09910* (2022).

[26] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th ACM SIGIR*. ACM, 154–161.

[27] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. 2007. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM TOIS* 25, 2 (2007), 7.

[28] Branislav Kveton, Chang Li, Tor Lattimore, Ilya Markov, Maarten de Rijke, Csaba Szepesvari, and Masrour Zoghi. 2018. Bubblerank: Safe online learning to rerank. *arXiv preprint arXiv:1806.05819* (2018).

[29] Branislav Kveton, Csaba Szepesvári, Mohammad Ghavamzadeh, and Craig Boutilier. 2019. Perturbed-History Exploration in Stochastic Linear Bandits. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*. 176.

[30] Branislav Kveton, Csaba Szepesvari, Sharan Vaswani, Zheng Wen, Tor Lattimore, and Mohammad Ghavamzadeh. 2019. Garbage in, reward out: Bootstrapping exploration in multi-armed bandits. In *International Conference on Machine Learning*. PMLR, 3601–3610.

[31] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. 2015. Combinatorial cascading bandits. In *NIPS*. 1450–1458.

[32] Branislav Kveton, Manzil Zaheer, Csaba Szepesvári, Lihong Li, Mohammad Ghavamzadeh, and Craig Boutilier. 2020. Randomized Exploration in Generalized Linear Bandits. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*.

[33] Tor Lattimore, Branislav Kveton, Shuai Li, and Csaba Szepesvari. 2018. Toprank: A practical algorithm for online stochastic ranking. In *NIPS*. 3945–3954.

[34] Tor Lattimore and Csaba Szepesvári. 2019. *Bandit Algorithms*. Cambridge University Press. In press.

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[36] Shuai Li, Tor Lattimore, and Csaba Szepesvári. 2018. Online learning to rank with features. *arXiv preprint arXiv:1810.02567* (2018).

[37] Shiyu Liang and R Srikant. 2016. Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161* (2016).

[38] Tie-Yan Liu. 2011. Learning to rank for information retrieval. (2011).

[39] Haihao Lu and Kenji Kawaguchi. 2017. Depth Creates No Bad Local Minima. *arXiv preprint arXiv:1702.08580* (2017).

[40] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable unbiased online learning to rank. In *Proceedings of the 27th ACM CIKM*. 1293–1302.

[41] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2970–2978.

[42] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. arXiv:1306.2597 [cs.IR]

[43] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In *Proceedings of the 9th ACM WSDM*. 457–466.

[44] Anne Schuth, Floor Sietsma, Shimon Whiteson, Damien Lefortier, and Maarten de Rijke. 2014. Multileaved comparisons for fast online evaluation. In *Proceedings of the 23rd ACM CIKM*. ACM, 71–80.

[45] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 373–382.

[46] Matus Telgarsky. 2015. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101* (2015).

[47] Matus Telgarsky. 2016. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485* (2016).

[48] Huazheng Wang, Sonwoo Kim, Eric McCord-Snook, Qingyun Wu, and Hongning Wang. 2019. Variance Reduction in Gradient Exploration for Online Learning to Rank. In *SIGIR 2019*. 835–844.

[49] Huazheng Wang, Ramsey Langley, Sonwoo Kim, Eric McCord-Snook, and Hongning Wang. 2018. Efficient exploration of gradient space for online learning to rank. In *SIGIR 2018*. 145–154.

[50] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *CIKM '18*. ACM, 1313–1322.

[51] Virginia Vassilevska Williams. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 887–898.

[52] Dmitry Yarotsky. 2017. Error bounds for approximations with deep ReLU networks. *Neural Networks* 94 (2017), 103–114.

[53] Dmitry Yarotsky. 2018. Optimal approximation of continuous functions by very deep ReLU networks. *arXiv preprint arXiv:1802.03620* (2018).

[54] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*. 1201–1208.

[55] Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. 2020. Neural Thompson Sampling. *arXiv preprint arXiv:2010.00827* (2020).

[56] Dongruo Zhou, Lihong Li, and Quanquan Gu. 2019. Neural Contextual Bandits with UCB-based Exploration. *arXiv preprint arXiv:1911.04462* (2019).

[57] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. 2019. Stochastic gradient descent optimizes over-parameterized deep ReLU networks. *Machine Learning*

(2019).
[58] Difan Zou and Quanquan Gu. 2019. An Improved Analysis of Training Over-parameterized Deep Neural Networks. In *Advances in Neural Information Processing Systems.*

A APPENDIX

A.1 Proof of Lemma 4.2

Proof. According to the definition of $g_t(\boldsymbol{\theta})$ in Section 4, we have the following equation for the auxiliary solution $\bar{\boldsymbol{\theta}}_t$,

$$g_t(\bar{\boldsymbol{\theta}}_t) = \sum_{s=-d+1}^{t-1}\sum_{(i,j)\in\Omega_s} \mathbf{y}_{ij}^s\mathbf{x}_{ij}^s$$

Then, to bound the deviation caused by the observation noise, we have the following derivations for any input $\mathbf{x}$

$$|\sigma(\mathbf{x}^\top\bar{\boldsymbol{\theta}}_t) - \sigma(\mathbf{x}^\top\boldsymbol{\theta}^*)| \le k_\mu|\mathbf{x}^\top(\bar{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^*)|$$
$$=k_\mu|\mathbf{x}^\top\mathbf{G}_t^{-1}(g_t(\bar{\boldsymbol{\theta}}_t) - g_t(\boldsymbol{\theta}^*))| \le \frac{k_\mu}{c_\mu}|\mathbf{x}^\top\mathbf{A}_t^{-1}(g_t(\bar{\boldsymbol{\theta}}_t) - g_t(\boldsymbol{\theta}^*))|$$
$$\le\frac{k_\mu}{c_\mu}\|\mathbf{x}\|_{\mathbf{A}_t^{-1}}\|g_t(\bar{\boldsymbol{\theta}}_t) - g_t(\boldsymbol{\theta}^*)\|_{\mathbf{A}_t^{-1}}$$

The first inequality is due to the Lipschitz continuity of the logistic function. As logistic function is continuously differentiable, the second equality is according to the Fundamental Theorem of Calculus, where $\mathbf{G}_t = \sum_{s=1}^{t-1}\sum_{(i,j)\in\Omega_s}\dot{\sigma}(\mathbf{x}_{ij}^{s\top}\boldsymbol{\theta})\mathbf{x}_{ij}^s\mathbf{x}_{ij}^{s\top} + \lambda\mathbf{I}$. In the third inequality, $\mathbf{A}_t = \sum_{s=1}^{t-1}\sum_{(i,j)\in\Omega_s}\mathbf{x}_{ij}^s\mathbf{x}_{ij}^{s\top} + \lambda\mathbf{I}$. And this inequality holds as $\mathbf{G}_t \succ c_\mu\mathbf{A}_t$, with $c_\mu = \inf_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\dot{\sigma}(\mathbf{x}^\top\boldsymbol{\theta})$.

Next, we will bound $\|g_t(\bar{\boldsymbol{\theta}}_t) - g_t(\boldsymbol{\theta}^*)\|_{\mathbf{A}_t^{-1}}$.

$$\|g_t(\bar{\boldsymbol{\theta}}_t) - g_t(\boldsymbol{\theta}^*)\|_{\mathbf{A}_t^{-1}}$$
$$=\|\sum_{s=-d+1}^{t-1}\mathbf{y}_{ij}^s\mathbf{x}_{ij}^s - \sigma(\mathbf{x}_{ij}^{s\top}\boldsymbol{\theta}^*)\mathbf{x}_{ij}^s\|_{\mathbf{A}_t^{-1}}$$
$$\le\|\sum_{s=-d+1}^{0}\sigma(\mathbf{x}_{ij}^{s\top}\boldsymbol{\theta}^*)\mathbf{x}_{ij}^s\|_{\mathbf{A}_t^{-1}} + \|\sum_{s=1}^{t-1}\epsilon_{ij}^s\mathbf{x}_{ij}^s\|_{\mathbf{A}_t^{-1}}$$
$$\le d + R\sqrt{\log(\frac{\det(\mathbf{A}_t)}{\delta^2\det(\lambda\mathbf{I})})}$$

The first equality is based on the definition of function g_t, and $\bar{\boldsymbol{\theta}}_t$. The last inequality is according to the self-normalized bound for martingales in [1]. This completes the proof. □

A.2 Proof of Lemma 4.4

Proof. According to Lemma 4.2, with probability at least $1-\delta$, event E_t defined in Section 4 occurs. Under event E_t, for document pair (i,j) satisfying $i \succ j$ for the given query (e.g., $\sigma(\mathbf{x}_{ij}^{t\top}\boldsymbol{\theta}^*) > \frac{1}{2}$), we first analyze the probability that at least one estimate $\sigma(\mathbf{x}_{ij}^{t\top}\widehat{\boldsymbol{\theta}}_t^{(n)}) > \frac{1}{2}$ for $n\in[N]$, e.g., $\mathbb{P}(\max_{n\in[N]}\sigma(\mathbf{x}_{ij}^{t\top}\widehat{\boldsymbol{\theta}}_t^{(n)})) > \frac{1}{2}$. For simplicity, in the following analysis, we use $\widehat{\sigma}_t^{(n)}$, $\bar{\sigma}_t$and σ^* to present $\sigma\left(\mathbf{x}_{ij}^{t\top}\widehat{\boldsymbol{\theta}}_t^{(n)}\right)$, $\sigma\left(\mathbf{x}_{ij}^{t\top}\bar{\boldsymbol{\theta}}_t\right)$and $\sigma\left(\mathbf{x}_{ij}^{t\top}\boldsymbol{\theta}^*\right)$ respectively.

$$\mathbb{P}(\max_{n\in[N]}\widehat{\sigma}_t^{(n)} \ge \frac{1}{2}) = 1 - \prod_{n=1}^{N}\mathbb{P}(\sigma_t^{(n)} < \frac{1}{2})$$

For any $n\in[N]$, we have the following bound for $\mathbb{P}(\sigma_t^{(n)} < \frac{1}{2})$.

$$\mathbb{P}(\sigma_t^{(n)} < \frac{1}{2}) \le\mathbb{P}(\sigma_t^{(n)} < \sigma^*)$$
$$=\mathbb{P}(\sigma_t^{(n)} - \bar{\sigma}_t < \sigma^* - \bar{\sigma}_t)$$
$$\le\mathbb{P}(\sigma_t^{(n)} - \bar{\sigma}_t < \alpha_t\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}})$$
$$\le\mathbb{P}(c_\mu\mathbf{x}_{ij}^{t\top}(\widehat{\boldsymbol{\theta}}_t - \bar{\boldsymbol{\theta}}) < \alpha_t\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}})$$
$$\le\mathbb{P}(c_\mu\mathbf{x}_{ij}^{t\top}\mathbf{G}_t^{-1}(g_t(\widehat{\boldsymbol{\theta}}_t) - g_t(\bar{\boldsymbol{\theta}})) < \alpha_t\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}})$$
$$\le\mathbb{P}(\frac{c_\mu}{k_\mu}\mathbf{x}_{ij}^{t\top}\mathbf{A}_t^{-1}(g_t(\widehat{\boldsymbol{\theta}}_t) - g_t(\bar{\boldsymbol{\theta}})) < \alpha_t\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}})$$
$$=\mathbb{P}(U(\mathbf{x}_{ij}^t) < \frac{k_\mu}{c_\mu}\alpha_t\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}})$$

where $U(\mathbf{x}_{ij}^t) = \mathbf{x}_{ij}^{t\top}\mathbf{A}_t^{-1}(g_t(\widehat{\boldsymbol{\theta}}_t) - g_t(\bar{\boldsymbol{\theta}}))$. According to the definition of $\bar{\boldsymbol{\theta}}_t$, we know that,

$$U(\mathbf{x}_{ij}^t) = \mathbf{x}_{ij}^{t\top}\mathbf{A}_t^{-1}\sum_{s=-d+1}^{t-1}\gamma_{ij}^s\mathbf{x}_{ij}^s$$

It is easy to obtain that $\mathbb{E}[U(\mathbf{x}_{ij}^t)] = 0$ with $\mathbb{E}[\gamma] = 0$. And the variance of $U(\mathbf{x}_{ij}^t)$ is,

$$Var[U(\mathbf{x}_{ij}^t)] =\nu^2\mathbf{x}_{ij}^{t\top}\mathbf{A}_t(\sum_{s=-d+1}^{t-1}\mathbf{x}_{ij}^s\mathbf{x}_{ij}^{s\top})\mathbf{A}_t\mathbf{x}_{ij}^t$$
$$=\nu^2\mathbf{x}_{ij}^{t\top}\mathbf{A}_t^{-1}(\lambda\mathbf{I} + \sum_{s=1}^{t-1}\mathbf{x}_{ij}^s\mathbf{x}_{ij}^{s\top})\mathbf{A}_t^{-1}\mathbf{x}_{ij}^t$$
$$=\nu^2\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}}^2$$

Therefore, we have $U(\mathbf{x}_{ij}^t) \sim \mathcal{N}(0, \nu^2\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}}^2)$, and the probability can be upper bounded as,

$$\mathbb{P}(U(\mathbf{x}_{ij}^t) < \frac{k_\mu}{c_\mu}\alpha_t\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}}) =1 - \mathbb{P}(U(\mathbf{x}_{ij}^t) > \frac{k_\mu}{c_\mu}\alpha_t\|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}})$$
$$\le 1 - \frac{\exp(-\beta^2)}{4\sqrt{\pi}\beta}$$

where $\beta = \frac{k_\mu\alpha_t}{c_\mu\nu}$. By chaining all the inequalities, we have that with $N \ge \log\delta/\log(1-\exp(-\beta^2)/(4\sqrt{\pi}\beta))$, with probability at least $1-\delta$, $\mathbb{P}(\max_{n\in[N]}\widehat{\sigma}_t^{(n)} \ge \frac{1}{2})$.

Therefore, under event E_t, with $N \ge \log\delta/\log(1-\exp(-\beta^2)/(4\sqrt{\pi}\beta))$, based on the definition of ω_t^u in Section 3.2, we know that for document i and j at round t, $(i,j)\in\omega_t^c$ if and only if $\min_{n\in[N]}\sigma\left(\mathbf{x}_{ij}^{t\top}\widehat{\boldsymbol{\theta}}_t^{(n)}\right) > \frac{1}{2}$. For simplicity, in the following analysis, we use $\widehat{\sigma}_t^{(n)}$, $\bar{\sigma}_t$and σ^* to present $\sigma\left(\mathbf{x}_{ij}^{t\top}\widehat{\boldsymbol{\theta}}_t^{(n)}\right)$, $\sigma\left(\mathbf{x}_{ij}^{t\top}\bar{\boldsymbol{\theta}}_t\right)$and $\sigma\left(\mathbf{x}_{ij}^{t\top}\boldsymbol{\theta}^*\right)$ respectively. Then the probability of being uncertain can be bounded as,

$$\mathbb{P}\big((i,j)\in\omega_t^u\big) = 1 - \mathbb{P}(\min_{n\in[N]}\widehat{\sigma}_t^{(n)} > 1/2)$$
$$=1 - \prod_{n=1}^{N}\mathbb{P}(\widehat{\sigma}_t^{(n)} > 1/2) = 1 - \big(\mathbb{P}(\widehat{\sigma}_t^{(n)} > 1/2)\big)^N$$

Based on the definition of $\Delta_{\min}$, $\sigma^* - 1/2 \ge \Delta_{\min}$ and $\sigma^* - \bar{\sigma}_t \le CB$. And according to the random matrix theory and Lemma 2 in [24], when $t > t'$, $\Delta_{\min} - CB > 0$ which can be viewed as a constant $c\Delta_{\min}$. Therefore, we have the following inequalities,

$$\mathbb{P}\big(\widehat{\sigma}_t^{(m)} > 1/2\big) = \mathbb{P}\big(\widehat{\sigma}_t^{(m)} - \sigma^* + \sigma^* - \bar{\sigma}_t + \bar{\sigma}_t > 1/2\big)$$
$$\geq \mathbb{P}\big(\bar{\sigma}_t - \widehat{\sigma}_t^{(m)} < \Delta_{\min} - CB\big) = 1 - \frac{1}{2}\mathbb{P}\big(|\bar{\sigma}_t - \widehat{\sigma}_t^{(m)}| \geq \Delta_{\min} - CB\big)$$
$$\geq 1 - \frac{1}{2}\mathbb{P}\big(k_\mu |U(x)| \geq \Delta_{\min} - CB\big) \geq 1 - \exp\Big(-\frac{c_\mu^2 c^2 \Delta_{\min}^2}{2k_\mu^2 \nu^2 \|x\|_{\mathbf{A}_t^{-1}}^2}\Big)$$

Let $B = \frac{c_\mu^2 c^2 \Delta_{\min}^2}{2k_\mu^2 \nu^2 \|x\|_{\mathbf{A}_t^{-1}}^2}$, we have the probability of being uncertain rank order upper bounded as,

$$\begin{aligned}\mathbb{P}\big((i,j) \in \omega_t^u\big) \leq & 1 - \big(1 - \exp(-B)\big)\big)^N \\ = & 1 - \exp N \log(1 - \exp(-B)) \\ \leq & -N \log(1 - \exp(-B)) \leq -N \log \exp(-1/B))\end{aligned}$$

Therefore, $\mathbb{P}\big((i,j) \in \omega_t^u\big) \leq \min\Big\{1, \frac{2Nk_\mu^2 \nu^2 \|x_{i,j}\|_{\mathbf{A}_t^{-1}}^2}{c_\mu^2 c^2 \Delta_{\min}^2}\Big\}$. This completes the proof. □

A.3 Proof of Theorem 4.5

Proof. Once the certain and uncertain rank orders are determined, our proposed model will generate the ranked list by topological sort with respect to the certain rank orders. Therefore, the regret only comes from the uncertain rank orders. In each round of result serving, as the model $\boldsymbol{\theta}_t$ would not change until the next round, the expected number of uncertain rank orders, U_t, can be estimated by summing the uncertain probabilities over all possible pairwise comparisons under the current query q_t, e.g., $\mathbb{E}[U_t] = \frac{V_t(V_t-1)}{2}\mathbb{P}((i,j) \in \omega_t^u)$.

Based on Lemma 4.4, the cumulative number of mis-ordered pairs can be bounded by the probability of observing uncertain rank orders in each round, which shrinks with more observations become available over time,

$$\mathbb{E}\Big[\sum_{s=t'}^T U_t\Big] \leq \mathbb{E}\Big[\frac{1}{2}\sum_{s=t'}^T \sum_{(i,j)\in[V_t]^2} \frac{2Nk_\mu^2 \nu^2 \|\mathbf{x}_{ij}^s\|_{\mathbf{A}_s^{-1}}^2}{c_\mu^2 c^2 \Delta_{\min}^2}\Big].$$

As $\mathbf{A}_t$ only contains information of observed document pairs so far, the number of mis-ordered pairs among the observed documents is guaranteed to be upper bounded. To reason about the number of mis-ordered pairs in those unobserved documents (i.e., from o_t to V_t for each query q_t), we leverage the constant p^*, which is defined as the minimal probability that all documents in a query are examined over time,

$$\begin{aligned}&\mathbb{E}\Big[\sum_{t=t'}^T \sum_{(i,j)\in[V_t]^2} \|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}}\Big] \\ =&\mathbb{E}\Big[\sum_{t=t'}^T \sum_{(i,j)\in[V_t]^2} \|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}} \times \mathbb{E}\big[p_t^{-1}\mathbf{1}\{o_t = V_t\}\big]\Big] \\ \leq & p^{*-1}\mathbb{E}\Big[\sum_{t=t'}^T \sum_{(i,j)\in[V_t]^2} \|\mathbf{x}_{ij}^t\|_{\mathbf{A}_t^{-1}}\mathbf{1}\{o_t = V_t\}\Big]\end{aligned}$$

Besides, we only use the independent pairs Ω_t to update the model and the corresponding $\mathbf{A}_t$ matrix. Therefore, to bound the regret, the pairs can be divided into two parts based on whether they are belonging to the observed set Ω_t. Then, we have the following inequalities,

$$\begin{aligned}&\sum_{s=t'}^T \sum_{(i,j)\in\Omega_s} \sum_{k\in[V_t]\setminus\{i,j\}} 2\mathbf{x}_{ik}^{s\ \top}\mathbf{A}_s^{-1}\mathbf{x}_{jk}^s \\ \leq &\sum_{s=t'}^T (V_{\max}^2 - 2V_{\max})P^2/\lambda_{\min}(\mathbf{A}_s).\end{aligned}$$

And for the pairs belonging to the $\{\Omega_s\}_{s=1}^T$, based on Lemma 10 and Lemma 11 in [1], we have,

$$\sum_{s=t'}^T \sum_{(i,j)\in\Omega_s} (V_t - 1)\|\mathbf{x}_{ij}^s\|_{\mathbf{A}_s^{-1}}^2 \leq 2dV_{\max}\log\Big(1 + \frac{o_{\max}TP^2}{2d\lambda}\Big)$$

Then, chaining all the inequalities, we have when event E_t happens, the regret can be upper bounded as,

$$R_T \leq R' + \sum_{s=t'}^T r_s \leq R' + \frac{1}{p^*}2dV_{\max}C\log\Big(1 + \frac{o_{\max}TP^2}{2d\lambda}\Big)$$

where $C = 2Nk_\mu^2\nu^2/c_\mu^2 c^2 \Delta_{\min}^2$, $R' = t'V_{\max}$, with t' defined in Lemma 4.4. This completes the proof. □