Accelerated Convergence for Counterfactual Learning to Rank

Rolf Jagerman
University of Amsterdam
Amsterdam, The Netherlands
rolf.jagerman@uva.nl

Maarten de Rijke
University of Amsterdam & Ahold Delhaize
Amsterdam, The Netherlands
m.derijke@uva.nl

ABSTRACT

Counterfactual Learning to Rank (LTR) algorithms learn a ranking model from logged user interactions, often collected using a production system. Employing such an offline learning approach has many benefits compared to an online one, but it is challenging as user feedback often contains high levels of bias. Unbiased LTR uses Inverse Propensity Scoring (IPS) to enable unbiased learning from logged user interactions. One of the major difficulties in applying Stochastic Gradient Descent (SGD) approaches to counterfactual learning problems is the large variance introduced by the propensity weights. In this paper we show that the convergence rate of SGD approaches with IPS-weighted gradients suffers from the large variance introduced by the IPS weights: convergence is slow, especially when there are large IPS weights.

To overcome this limitation, we propose a novel learning algorithm, called CounterSample, that has provably better convergence than standard IPS-weighted gradient descent methods. We prove that CounterSample converges faster and complement our theoretical findings with empirical results by performing extensive experimentation in a number of biased LTR scenarios – across optimizers, batch sizes, and different degrees of position bias.

CCS CONCEPTS

• Information systems → Learning to rank; **• Theory of computation → Convex optimization**.

KEYWORDS

Learning to Rank; Unbiased Learning; Counterfactual Learning

ACM Reference Format:
Rolf Jagerman and Maarten de Rijke. 2020. Accelerated Convergence for Counterfactual Learning to Rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401069

1 INTRODUCTION

Learning to Rank (LTR) from *user interactions* [15], as opposed to learning from *annotated datasets* [27], has seen increased research interest due to its immense practical value. Learning from implicit feedback enjoys several advantages over learning from professional annotations: (1) user interaction signals are available at large scale and cost much less than professional annotations, (2) implicit feedback captures the user's true interest more accurately, and (3) interaction data can be utilized in domains where professional annotations are impractical, unethical, or impossible, for example in personal search. However, learning from user interactions is not without difficulty and one of the major challenges is the biased nature of user interactions [12, 22]. For example, in LTR, one of the most important types of bias that affects user interaction data is *position bias*, a phenomenon where users observe, and as a result click on, top-ranked items more than lower ranked ones.

Recent work has focused on removing bias by applying methods from counterfactual learning [24]. Most notably, Inverse Propensity Scoring (IPS) is commonly used to perform unbiased learning. A widely used approach for unbiased learning is to treat inverse propensity scores as weights and solve a weighted optimization problem via Stochastic Gradient Descent (SGD) [4, 5, 23].

A major challenge of learning with IPS-weighted SGD is that the propensity weights introduce a large amount of variance in the gradients. This effect is especially severe in scenarios where the propensity weights can take on extreme values. For example, in product search, the set of candidate results tends to be large [7] and the query distribution can be heavily skewed [16] (e.g., due to periodic or seasonal influences), which means that some items get very little exposure and as a result have extreme IPS weights.

In this paper we investigate the relationship between IPS weights and the convergence rate of IPS-weighted SGD. We prove that IPS-weighted SGD suffers from a slow convergence rate when the propensity weights are large. We also argue that as long as stochastic gradients are scaled with IPS-weights, this slowdown *cannot* be improved. This means that for many practical LTR scenarios, learning a ranking model is inefficient and convergence is slow.

We overcome the above limitation with a novel sample-based learning algorithm, called CounterSample, that samples learning instances proportional to their IPS weight instead of weighting learning instances by their IPS weight. Because of this strategy, we are able to control the variance, which, in turn, leads to accelerated convergence. We show that this new approach provably enjoys faster convergence while remaining unbiased and computationally cheap. We complement these theoretical findings with extensive experiments. CounterSample consistently converges faster, and in some cases learns a better ranker than IPS-weighted SGD, in a number of biased LTR scenarios – across optimizers, across batch sizes and for different severities of position bias.

Our main contributions in this paper are:

- We analyze the convergence rate of IPS-weighted SGD algorithms and formalize the relationship between IPS weights and the convergence rate.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401069

- We introduce a novel learning algorithm for Counterfactual Learning to Rank called CounterSample that has provably faster convergence than IPS-weighted SGD approaches.
- We empirically show that CounterSample converges faster than competing methods in a number of biased LTR scenarios.

The remainder of this paper is organized as follows. We discuss related work in Section 2. Section 3 provides the necessary notation and background for Counterfactual LTR. We analyze the convergence rate of IPS-weighted SGD algorithms for LTR in Section 4. Then, we introduce CounterSample, a method for Counterfactual LTR with faster convergence in Section 5. Sections 6 and 7 describe the experimental setup and empirical results, respectively. We conclude the paper in Section 8.

2 RELATED WORK

2.1 Counterfactual LTR

Recent work on unbiased Learning to Rank (LTR) uses counterfactual learning [37, 38] to remove different types of bias such as position bias from click data to improve ranking performance [24, 42]. These methods typically use Inverse Propensity Scoring (IPS) to enable unbiased learning. A popular approach for solving IPS-weighted learning problems is to use Stochastic Gradient Descent (SGD) algorithms [4, 5, 23]. Existing approaches accomplish this by scaling the loss (and as a result, the gradients) with IPS weights. Although the empirical success of such approaches has been well documented in the literature [4, 5, 23], the impact of IPS weights on the *convergence rate* of SGD is not well understood.

In this paper we address this problem by investigating the relationship between IPS weights and the convergence rate of SGD. Furthermore, we introduce a novel learning method that, unlike previous approaches, does *not* scale the loss or gradients, but instead guarantees unbiasedness by employing a sampling procedure.

2.2 Position Bias

An important aspect of unbiased LTR is estimating the observation probabilities (often called *propensities*), which are necessary to apply IPS-weighting. For example, Wang et al. [43] propose result randomization strategies for obtaining observation probabilities under a position bias user model. Recent work has focused on *intervention harvesting*, a less invasive method for propensity estimation [4, 14].

In our work, we do *not* focus on the propensity estimation aspect of unbiased LTR, instead assuming the propensity scores are known a priori, because our goal is to study the *convergence rate* of IPS-weighted SGD algorithms.

2.3 Convergence Rates for SGD

There is a significant amount of work studying upper bounds for the convergence rate of SGD [33] under varying assumptions of convexity and smoothness of the optimization objective [17, 18, 31, 36]. The convergence rate proofs presented in this paper build on proofs provided by Shalev-Shwartz and Ben-David [35]. However, unlike previous work, our work investigates the role of IPS weights in optimization problems. We note that there is work investigating the use of importance sampling for SGD algorithms [6, 25, 29, 44]. These approaches all start from an unbiased dataset and then improve the convergence rate of SGD by manually perturbing the sampling distribution during learning, which introduces bias, and then applying IPS-weighting to remove the introduced bias.

Our work is different from these approaches because we learn from an *already biased* click log dataset, where the source of bias is outside of our control (e.g., position bias coming from users). This means we do not assume control over how the dataset is generated nor do we assume control over the propensity scores.

3 BACKGROUND

We consider the problem of Counterfactual Learning to Rank (LTR) as described in [24]. First, we introduce our notation for LTR with additive metrics. After that, we describe Counterfactual LTR from biased click feedback and present the IPS-weighted SGD algorithm that is commonly used for Counterfactual LTR.

3.1 Learning to Rank with Additive Metrics

Let $S_{\boldsymbol{w}}(q,d)$ be a scoring function, parameterized by $\boldsymbol{w}$, that, for a given query q and item $d \in D_q$ produces a real-valued ranking score, where D_q is a set of candidate items for query q. Let $rel(q,d) \in \{0,1\}$ be the relevance of item d to query q, where we assume binary relevance for simplicity. We write $rank(d \mid q, D_q, S_{\boldsymbol{w}})$ for the rank of item $d \in D_q$ after sorting all items D_q by their respective scores using the scoring function $S_{\boldsymbol{w}}$. We consider the class of additive ranking metrics, as described in [2]:

$$\Delta(S_{\boldsymbol{w}} \mid q) = \sum_{d \in D_q} \lambda\big(rank(d \mid q, D_q, S_{\boldsymbol{w}})\big) \cdot rel(q,d), \tag{1}$$

where λ is a weighting function of the rank of an item that can capture different ranking metrics such as Average Relevant Rank, DCG, Precision@k and more [2]. For simplicity we will assume $\lambda(x) = x$, but note that our findings hold for any convex, (sub)differentiable and monotonically increasing weighting function λ. It is common practice to make Equation 1 differentiable by upper bounding the $rank(d \mid q, D_q, S_{\boldsymbol{w}})$ term with a pairwise hinge loss [2, 21, 24]:

$$\begin{aligned} &rank(d \mid q, D_q, S_{\boldsymbol{w}}) \leq \\ &\quad 1 + \sum_{d' \in D_q} \max(0, 1 - (S_{\boldsymbol{w}}(q,d) - S_{\boldsymbol{w}}(q,d'))). \end{aligned} \tag{2}$$

Now, suppose we are given a sample Q of i.i.d. queries $q \sim P(q)$. The goal in Learning to Rank (LTR) is to learn a scoring function $S_{\boldsymbol{w}}$ that minimizes risk:

$$\begin{aligned} \arg\min_{\boldsymbol{w}} R(\boldsymbol{w}) &= \arg\min_{\boldsymbol{w}} \int_q \Delta(S_{\boldsymbol{w}} \mid q) dP(q) \\ &= \arg\min_{\boldsymbol{w}} \mathbb{E}_Q\left[\frac{1}{|Q|}\sum_{q \in Q}\sum_{d \in D_q} \lambda\big(rank(d \mid q, D_q, S_{\boldsymbol{w}})\big) \cdot rel(q,d)\right]. \end{aligned} \tag{3}$$

In most practical settings we cannot directly observe the relevance $rel(q,d)$, but only partial feedback in the form of clicks collected on rankings produced by a deployed production ranker. As a result, we cannot directly minimize the empirical risk associated with Equation 3. Furthermore, unlike online LTR approaches and bandit algorithms, we assume that we do not have any control over the deployed production ranker (i.e., we cannot perform *interventions*). In Counterfactual LTR we instead focus on minimizing an unbiased estimate of the empirical risk using a historical click log.

3.2 Counterfactual Learning to Rank with Biased Feedback

Suppose a deployed production ranker is collecting user interactions, i.e., clicks, as follows:

- a user issues a query $q \sim P(q)$;
- the user is presented with a ranking of the candidate items D_q for the issued query; and
- the user observes and clicks on an item $d \in D_q$ with probability $P(c(d) = 1)$, where $c(d) \in \{0, 1\}$ indicates a click on item d.

In line with existing work [12, 24] we assume that the examination hypothesis holds, i.e., that clicks can only occur on observed items. In other words, a click on an item depends on the probability that the user observes the item *and* decides to click on it:

$$P(c(d) = 1) \overset{\text{def}}{=} P(o(d) = 1) \cdot P(c(d) = 1 \mid o(d) = 1), \tag{4}$$

where $o(d) \in \{0, 1\}$ indicates that item d was observed by the user. Finally, for any two observed items d and d' it is more likely that a user clicks on a relevant item than a non-relevant item. More formally:

$$\begin{aligned} &rel(q, d) > rel(q, d') \\ &\implies P(c(d) = 1 \mid o(d) = 1) > P\big(c(d') = 1 \mid o(d') = 1\big). \end{aligned} \tag{5}$$

Under these assumptions a click does not necessarily indicate relevance, nor does a non-click necessarily indicate non-relevance. In general $c(d) \neq rel(q, d)$, and naively optimizing the empirical risk using clicks would be a suboptimal strategy [24]. Instead, it is necessary to correct for the observation probabilities $P(o(d) = 1)$, often called the *propensity*. This motivates the use of Inverse Propensity Scoring (IPS), where inversely weighing the propensities of clicked items can debias the click data.

For our work we assume that the propensities $P(o(d) = 1)$ are known a priori. In practice, the propensities are commonly estimated via A/B testing [42] or through intervention harvesting [4, 14], usually under some assumption of a user behavior model such as the position-bias model [12]. Modeling and estimating these propensity scores falls outside the scope of this paper as our aim is to understand and improve the *convergence rate* of IPS-weighted optimization for LTR. We refer to existing work [4, 5, 9, 10, 14] for more detailed information about how the propensity scores can be modelled and/or estimated.

We now reach the main approach for Counterfactual LTR, which is to apply Inverse Propensity Scoring (IPS) to debias our optimization objective. Suppose we are given a *click log dataset* containing n clicks:

$$\mathcal{D} = \big\{(q_i, D_{q_i}, d_i, p_i)\big\}_{i=1}^{n}, \tag{6}$$

where:

- $q_i \sim P(q_i)$ is the issued query;
- D_{q_i} is the set of candidate items;
- $d_i \in D_{q_i}$ is the clicked item (i.e., $c(d_i) = 1$); and
- $p_i = P(o(d_i) = 1)$ is the propensity of item d_i.

Our goal now is to solve the following optimization problem:

$$\begin{aligned} \arg\min_{\boldsymbol{w}} R_{IPS}(\boldsymbol{w}) &= \arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{p_i} \lambda(rank(d_i \mid q_i, D_{q_i}, S_{\boldsymbol{w}})) \\ &= \arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{p_i} f_i(\boldsymbol{w}). \end{aligned} \tag{7}$$

It is easy to show that $R_{IPS}(\boldsymbol{w})$ is an unbiased estimate of $R(\boldsymbol{w})$, i.e., that $\mathbb{E}[R_{IPS}(\boldsymbol{w})] = \mathbb{E}[R(\boldsymbol{w})]$, as long as $P(o(d) = 1) > 0$ for all d, using the proof of Section 4 in [24]. We note that the minimization problem in Equation 7 permits stochastic optimization and can be efficiently solved with a variety of SGD approaches. A common approach for counterfactual learning is to treat the inverse propensity scores as weights and multiply the gradient $\nabla f_i(\boldsymbol{w})$ with $\frac{1}{p_i}$ to obtain an unbiased gradient estimate [2, 5, 19, 23]. We display this IPS-weighted SGD approach in Algorithm 1.

Algorithm 1 IPS-weighted Stochastic Gradient Descent (SGD)

```
  $\boldsymbol{w}_1 = \boldsymbol{0}$
  for $t \leftarrow 1, \ldots, T$ do
      $i_t \sim Uniform(0, n)$              ▷ sample $i_t$ from uniform distribution
      $\boldsymbol{g}_t = \frac{1}{p_{i_t}} \nabla f_{i_t}(\boldsymbol{w}_t)$              ▷ compute IPS-weighted gradient
      $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t$              ▷ SGD update step
  end for
  $\bar{\boldsymbol{w}} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{w}_t$
  return $\bar{\boldsymbol{w}}$
```

Scaling the gradients with IPS weights, as is done in Algorithm 1, is a common technique for unbiased LTR [2, 5, 8, 19, 43]. It is known that high-variance gradients can cause poor convergence for general SGD-style algorithms [35]. However, to the best of our knowledge, the exact impact of IPS weights on the convergence rate of SGD algorithms for LTR has not been studied. In other words, the nature of the relationship between the IPS weights and the convergence rate of SGD algorithms is an open problem. To address this problem, we will analyze Algorithm 1 and provide results describing the relationship between IPS weights and the convergence rate in the next section.

4 CONVERGENCE OF IPS-WEIGHTED SGD

In this section our aim is to better understand the convergence rate of Algorithm 1 and analyze the impact of IPS weights on the convergence rate.

4.1 Convergence Rate Analysis

Let $R_{IPS}(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{p_i} f_i(\boldsymbol{w})$ be the function that we wish to minimize using Algorithm 1. We assume that each $f_i(\boldsymbol{w})$ is convex and, consequently, that R_{IPS} is convex since each $\frac{1}{p_i} > 0$. In LTR this is a reasonable assumption because the loss can often be convex, for example the pairwise hinge loss formulation presented in Equation 2 is convex. Furthermore, the rank weighting functions $\lambda(\cdot)$ presented in Section 3.1 are convex for, e.g., the average relevant rank or DCG weighting schemes. Additionally, we assume R_{IPS} is minimized at some point $\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}:\|\boldsymbol{w}\| \leq B} R_{IPS}(\boldsymbol{w})$. We denote with M the maximum IPS weight in the dataset: $M = \max_i \frac{1}{p_i}$. As in previous analyses of regular SGD [35], the goal of our analysis is to bound the suboptimality of the solution produced by Algorithm 1:

$$\mathbb{E}\big[R_{IPS}(\bar{\boldsymbol{w}}) - R_{IPS}(\boldsymbol{w}^*)\big]. \tag{8}$$

Theorem 1. *Let $f_i(\boldsymbol{w})$ be a convex function for each i and let $\boldsymbol{w}^*$ be a minimizer of $R_{IPS}(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{p_i} f_i(\boldsymbol{w})$ such that $\|\boldsymbol{w}^*\| \leq B$. Assume that $\|\nabla f_{i_t}(\boldsymbol{w}_t)\| \leq G$ for all t and let $\bar{\boldsymbol{w}}$ be the solution* *produced by running Algorithm 1 for T iterations with $\eta = \sqrt{\frac{B^2}{(MG)^2 T}}$. Then:*

$$\mathbb{E}\big[R_{IPS}(\bar{\boldsymbol{w}}) - R_{IPS}(\boldsymbol{w}^*)\big] \leq \frac{B(MG)}{\sqrt{T}}. \tag{9}$$

Proof. This convergence rate proof is a variant of the proof of Theorem 14.8 from [35], where we use IPS-weighted gradients $\boldsymbol{g}_t$ and construct a bound on the gradient variance in Equation 15. Since our notation deviates slightly from the notation in [35], we include the full proof here for clarity. Denote with $i_{1:T}$ the sequence of random indices $i_1, \ldots, i_T$, then:

$$\begin{aligned}
&\mathbb{E}_{i_{1:T}}\left[R_{IPS}(\bar{\boldsymbol{w}}) - R_{IPS}(\boldsymbol{w}^*)\right] \\
&= \mathbb{E}_{i_{1:T}}\left[R_{IPS}\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{w}_t\right) - R_{IPS}(\boldsymbol{w}^*)\right] && \text{(10a)}\\
&\leq \mathbb{E}_{i_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T}R_{IPS}(\boldsymbol{w}_t) - R_{IPS}(\boldsymbol{w}^*)\right] && \text{(10b)}\\
&= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:T}}\left[R_{IPS}(\boldsymbol{w}_t) - R_{IPS}(\boldsymbol{w}^*)\right]. && \text{(10c)}
\end{aligned}$$

Here, (10a) follows from the definition of $\bar{\boldsymbol{w}}$, (10b) is due to Jensen's inequality [34], and, finally, (10c) is obtained by applying linearity of expectation. Since $\boldsymbol{w}_t$ depends only on the indices $i_{1:t-1}$ we get:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:T}}\left[R_{IPS}(\boldsymbol{w}_t) - R_{IPS}(\boldsymbol{w}^*)\right] \\
&= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:t-1}}\left[R_{IPS}(\boldsymbol{w}_t) - R_{IPS}(\boldsymbol{w}^*)\right]. && \text{(11)}
\end{aligned}$$

Once the indices $i_{1:t-1}$ are known, the value of $\boldsymbol{w}_t$ is no longer random. Furthermore, since each i_t is uniformly sampled from the dataset, it follows that $\boldsymbol{g}_t$ is an unbiased estimate of $\nabla R_{IPS}(\boldsymbol{w}_t)$:

$$\mathbb{E}_{i_t}[\boldsymbol{g}_t \mid i_{1:t-1}] = \mathbb{E}_{i_t}[\boldsymbol{g}_t \mid \boldsymbol{w}_t] = \nabla R_{IPS}(\boldsymbol{w}_t). \tag{12}$$

Continuing from Equation 11, we have:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:t-1}}\left[R_{IPS}(\boldsymbol{w}_t) - R_{IPS}(\boldsymbol{w}^*)\right] \\
&\leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:t-1}}\left[\left\langle \boldsymbol{w}_t - \boldsymbol{w}^*, \nabla R_{IPS}(\boldsymbol{w}_t)\right\rangle\right] && \text{(13a)}\\
&= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:t-1}}\left[\left\langle \boldsymbol{w}_t - \boldsymbol{w}^*, \mathbb{E}_{i_t}[\boldsymbol{g}_t \mid i_{1:t-1}]\right\rangle\right] && \text{(13b)}\\
&= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:t-1}}\mathbb{E}_{i_{1:t}}\left[\left\langle \boldsymbol{w}_t - \boldsymbol{w}^*, \boldsymbol{g}_t\right\rangle \mid i_{1:t-1}\right] && \text{(13c)}\\
&= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:t}}\left[\left\langle \boldsymbol{w}_t - \boldsymbol{w}^*, \boldsymbol{g}_t\right\rangle\right], && \text{(13d)}
\end{aligned}$$

where (13a) follows from the convexity of R_{IPS}, (13b) can be obtained by using Equation 12 ($\nabla R_{IPS}(\boldsymbol{w}_t) = \mathbb{E}[\boldsymbol{g}_t \mid i_{1:t-1}]$), (13c) is due to the linearity of expectation, and, finally, (13d) is obtained by applying the law of total expectation. We can now use Lemma 14.1 from [35] since Equation 13d is of the required form and obtain:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{i_{1:t}}\left[\left\langle \boldsymbol{w}_t - \boldsymbol{w}^*, \boldsymbol{g}_t\right\rangle\right] \leq \frac{1}{T}\left(\frac{\|\boldsymbol{w}^*\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2\right). \tag{14}$$

Here is where we deviate from the standard proof of convergence for SGD and upper bound the IPS-weighted gradients $\boldsymbol{g}_t$ as follows:

$$\|\boldsymbol{g}_t\| = \left\|\frac{1}{p_{i_t}}\nabla f_{i_t}(\boldsymbol{w}_t)\right\| = \frac{1}{p_{i_t}}\left\|\nabla f_{i_t}(\boldsymbol{w}_t)\right\| \leq MG. \tag{15}$$

Next, we can plug this upper bound on $\|\boldsymbol{g}_t\|$ into Equation 14, and use the assumption that $\|\boldsymbol{w}^*\| \leq B$ to obtain:

$$\frac{1}{T}\left(\frac{\|\boldsymbol{w}^*\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2\right) \leq \frac{1}{T}\left(\frac{B^2}{2\eta} + \frac{\eta}{2}T(MG)^2\right). \tag{16}$$

Finally, by plugging in $\eta = \sqrt{\frac{B^2}{(MG)^2 T}}$ and applying algebraic manipulations we obtain:

$$\begin{aligned}
&\frac{1}{T}\left(\frac{B^2}{2\eta} + \frac{\eta}{2}T(MG)^2\right) \\
&= \frac{1}{T}\left(\frac{B^2}{2\sqrt{\frac{B^2}{(MG)^2 T}}} + \frac{\sqrt{\frac{B^2}{(MG)^2 T}}}{2}T(MG)^2\right) && \text{(17a)}\\
&= \frac{1}{T}\left(\frac{B(MG)\sqrt{T}}{2} + \frac{B(MG)\sqrt{T}}{2}\right) && \text{(17b)}\\
&= \frac{B(MG)\sqrt{T}}{T} = \frac{B(MG)}{\sqrt{T}}. \qquad \square && \text{(17c)}
\end{aligned}$$

Theorem 1 combines several important quantities that determine how fast Algorithm 1 will converge. First, we have $1/\sqrt{T}$, which indicates that, as the number of iterations T grows, the solution $\bar{\boldsymbol{w}}$ gets closer to the optimal solution $\boldsymbol{w}^*$. Second, we have B, which tells us how far away $\boldsymbol{w}^*$ is from the starting point $\mathbf{0}$. Clearly, for large B, the optimum $\boldsymbol{w}^*$ is far away and we require more iterations to converge. Finally, we have the gradient variance term (MG). When the gradients have potentially large variance, we need to correspondingly set a small learning rate to prevent divergent behavior and, as a result, it takes longer to converge. As a consequence of Theorem 1, it is clear that in order to achieve an error of at most ϵ, it suffices to run Algorithm 1 for T iterations where:

$$T \geq \frac{B^2(MG)^2}{\epsilon^2}. \tag{18}$$

4.2 Discussion

The key insight that our analysis provides is that, for IPS-weighted SGD algorithms, the number of iterations required to achieve an ϵ-optimal solution grows with a factor $(MG)^2$. We note that there are known variations of SGD that can improve the convergence rate presented in Theorem 1 from $O\left(1/\sqrt{T}\right)$ to $O(1/T)$, for example see [17, 18, 31, 36]. However, their analyses are considerably more complex and do not remove the dependency on $\|\boldsymbol{g}_t\|^2$, which for the IPS-weighted SGD case remains upper bounded by $(MG)^2$. This means that despite having faster convergence in terms of T, these methods do not improve the slowdown introduced by the IPS-weights.

Moreover, Agarwal et al. [3] show that, for strongly convex and Lipschitz smooth functions, the term $\|\boldsymbol{g}_t\|^2$, and consequently the term $(MG)^2$ in Equation 18, cannot be improved for *any SGD algorithm* (for sufficiently large T). In practice, the value of M can be very large, for example in cases with small propensities p_i (e.g., in situations with a significant amount of position bias such as when the candidate set D_q is very large). The above facts lead us to hypothesize that, as long as the gradients are scaled with IPS weights, the convergence rate is severely slowed by the magnitude of the IPS weights. Our experiments in Section 7 support this hypothesis.

5 IMPROVED CONVERGENCE WITH WEIGHTED SAMPLING

5.1 CounterSample: SGD with IPS-proportional Sampling

Theorem 1 shows that the convergence of IPS-weighted SGD is slowed by a factor M^2. Moreover, as described in Section 4.2, as long as gradients $\boldsymbol{g}_t$ are scaled by IPS weights, this dependency on M cannot be improved [3]. Clearly, for situations where M is large, this can lead to slow convergence and make learning inefficient.

To overcome this problem we propose a sampling-based SGD strategy. The key idea is to debias our optimization objective via *sampling* instead of *weighting*. As we will prove below, this sampling-based approach similarly guarantees unbiasedness of the optimization objective but has a better convergence rate. We call our approach CounterSample and it is displayed in Algorithm 2.

Algorithm 2 CounterSample: SGD with IPS-proportional sampling

```
$\boldsymbol{w}_1 \leftarrow \boldsymbol{0}$
$\bar{M} \leftarrow \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i}$
for $t \leftarrow 1, \ldots, T$ do
    $i_t \sim P(i_t \mid \mathcal{D})$                ▹ sample $i_t$ according to Equation 19
    $\boldsymbol{g}_t = \bar{M}\,\nabla f_{\lambda,i_t}(\boldsymbol{w}_t)$                ▹ compute gradient
    $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\boldsymbol{g}_t$                ▹ SGD update step
end for
$\bar{\boldsymbol{w}} \leftarrow \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{w}_t$
return $\bar{\boldsymbol{w}}$
```

CounterSample functions as follows. First, we assign each data point i in our dataset $\mathcal{D}$ the following probability of being sampled:

$$P(i \mid \mathcal{D}) = \frac{\frac{1}{p_i}}{\sum_{j=1}^{n}\frac{1}{p_j}}. \tag{19}$$

We then proceed exactly like regular SGD, where instead of sampling datapoints uniformly from the dataset, they are sampled using Equation 19. Furthermore, the algorithm does not scale the gradients by the IPS-weights, but by a constant factor:

$$\bar{M} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i}, \tag{20}$$

which is necessary to guarantee unbiasedness (see Section 5.2).

In practice, one would tune the learning rate η in Algorithm 2, making the inclusion of $\bar{M}$ unnecessary as it merely scales the gradients by a constant, which can be offset by the particular η chosen. Therefore, for practical implementations, it is not necessary to include this constant. We include $\bar{M}$ here for the purposes of guaranteeing unbiasedness and analyzing the convergence rate.

5.2 Unbiasedness

To show that Algorithm 2 minimizes the unbiased objective $R_{IPS}(\boldsymbol{w})$, it is sufficient to show that, in expectation, the gradient $\boldsymbol{g}_t$ is an unbiased estimate of $\nabla R_{IPS}(\boldsymbol{w}_t)$ for any t.

Theorem 2. *Let $R_{IPS}(\boldsymbol{w})$ be the function to be optimized and let $\boldsymbol{g}_t$ be the gradient at time t as computed by Algorithm 2, then:*

$$\mathbb{E}_{i_t}[\boldsymbol{g}_t \mid \boldsymbol{w}_t] = \nabla R_{IPS}(\boldsymbol{w}_t). \tag{21}$$

Proof. The proof uses the definition of $\boldsymbol{g}_t$ and linearity of expectation (22a) and the definition of expectation (22b) where we use Equation 19 as the sampling probability. Using the definition of $\bar{M}$ and algebraic manipulations completes the proof ((22c) and (22d)):

$$\mathbb{E}_{i_t}[\boldsymbol{g}_t \mid \boldsymbol{w}_t] = \mathbb{E}\big[\bar{M}\,\nabla f_{i_t}(\boldsymbol{w}_t)\big] = \bar{M}\,\mathbb{E}\big[\nabla f_{i_t}(\boldsymbol{w}_t)\big] \tag{22a}$$

$$= \bar{M}\sum_{i=1}^{n} P(i \mid \mathcal{D})\nabla f_i(\boldsymbol{w}_t) \tag{22b}$$

$$= \frac{1}{n}\left(\sum_{i=1}^{n}\frac{1}{p_i}\right)\sum_{i=1}^{n}\frac{\frac{1}{p_i}}{\sum_{j=1}^{n}\frac{1}{p_j}}\nabla f_i(\boldsymbol{w}_t) \tag{22c}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i}\nabla f_i(\boldsymbol{w}_t) = \nabla R_{IPS}(\boldsymbol{w}_t). \quad \square \tag{22d}$$

5.3 Convergence Rate

We wish to understand the convergence rate of the proposed method CounterSample (Algorithm 2). Similar to Section 4.1, the goal of our analysis is to bound the suboptimality of the solution produced by Algorithm 2:

$$\mathbb{E}\big[R_{IPS}(\bar{\boldsymbol{w}}) - R_{IPS}(\boldsymbol{w}^*)\big]. \tag{23}$$

Theorem 3. *Let $f_i(\boldsymbol{w})$ be a convex function for each i and let $\boldsymbol{w}^*$ be a minimizer of $R_{IPS}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i}f_i(\boldsymbol{w})$ such that $\|\boldsymbol{w}^*\| \leq B$. Assume that $\|\nabla f_{i_t}(\boldsymbol{w}_t)\| \leq G$ for all t and let $\bar{\boldsymbol{w}}$ be the solution produced by running Algorithm 2 for T iterations with $\eta = \sqrt{\frac{B^2}{(\bar{M}G)^2 T}}$. Then:*

$$\mathbb{E}\big[R_{IPS}(\bar{\boldsymbol{w}}) - R_{IPS}(\boldsymbol{w}^*)\big] \leq \frac{B(\bar{M}G)}{\sqrt{T}}. \tag{24}$$

Proof. First, we note that in Algorithm 2, the gradients are bounded as follows:

$$\|\boldsymbol{g}_t\| = \big\|\bar{M}\,\nabla f_{i_t}(\boldsymbol{w}_t)\big\| \leq \bar{M}G. \tag{25}$$

Next, we follow the proof of Theorem 1, replacing Equation 15 with Equation 25, using the result of Theorem 2 to ensure $\mathbb{E}_{i_t}[\boldsymbol{g}_t \mid \boldsymbol{w}_t] = \nabla R_{IPS}(\boldsymbol{w}_t)$ and plugging in $\eta = \sqrt{\frac{B^2}{(\bar{M}G)^2 T}}$ to give us the desired result. □

As a result of Theorem 3, we can conclude that CounterSample provides significant advantages in terms of convergence rate over

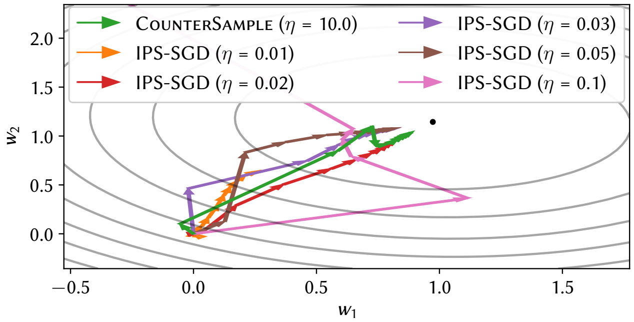

Figure 1: Illustration of the convergence of CounterSample versus IPS-weighted SGD on a synthetic learning example with two weights w_1 and w_2. The algorithms are run for $T = 50$ iterations. Best viewed in color.

standard IPS weighting. Specifically, to obtain an error of at most ϵ, it is sufficient to run CounterSample for:

$$T \geq \frac{B^2(\bar{M}G)^2}{\epsilon^2} \tag{26}$$

iterations. This is strictly better than the bound that was obtained for Algorithm 1 in nearly all cases. The only case where the two methods have the same convergence rate is when $\bar{M} = M$, which can only happen when all the propensity scores are the same, i.e., when $p_i = p_j$ for all i, j. However, this can only be the case when the click log itself is already unbiased, thus negating the need to do counterfactual learning in the first place. Therefore, for any practical Counterfactual LTR scenario, CounterSample is strictly better in terms of convergence rate than naively scaling the gradients with IPS weights.

5.4 Efficiency

Finally, despite the advantages of CounterSample in terms of convergence rate, these benefits may not be useful if they come at the cost of worse computational complexity. A straightforward but naive implementation for sampling from Equation 19 would result in a $O(Tn)$ time complexity for Algorithm 2, which is significantly worse than the $O(T)$ complexity obtained by standard SGD approaches such as Algorithm 1.

Fortunately, sampling from Equation 19 can be done with an amortized $O(1)$ cost using the alias method [40, 41]. To achieve this, there is a one time cost of constructing the alias table, done in $O(n)$. Furthermore, we also need to compute the constant $\bar{M}$ which is also a one time operation of $O(n)$. We note that both of these steps can take place during data pre-processing and are for most practical implementations easily achieved (e.g., PyTorch [30] and Tensorflow [1] both support efficient sampling from a weighed multinomial distribution using the alias method). Overall, this means that the complexity of CounterSample is $O(n + T)$, which is acceptable when $n < T$.

5.5 Illustrative Example

To illustrate the difference in convergence rates between CounterSample and standard IPS-weighted SGD we have created a simple toy example learning problem in Figure 1. We chose two optimal weights $\boldsymbol{w}^* = [w_1, w_2]$ and synthesize an IPS-weighted regression dataset.[1] Notice that, for large learning rates, the IPS-weighted SGD approach leads to unstable learning and diverges from the optimum. The learning rate for IPS-weighted SGD needs to be small enough to ensure that training samples with large IPS weights do not cause too large a step, possibly leading to divergent behavior. As a result, it is necessary to reduce the learning rate to ensure stable learning, however doing so naturally increases the time until convergence for IPS-weighted SGD because samples that do not have extreme IPS weights can only make small progress to the optimum. On the other hand, CounterSample can reliably handle large learning rates because the gradients are not scaled with potentially large IPS weights. Instead, CounterSample samples training instances with high IPS weights more frequently. Overall, this leads to much faster convergence.

Table 1: Datasets used for our experiments.

Dataset	Queries	Avg. docs per query	$rel(q, d) = 0$
Yahoo [11]	36,251	23	26%
Istella-s [28]	33,118	103	89%

6 EXPERIMENTAL SETUP

Our experimental setup is aimed at assessing the convergence rate of CounterSample and to answer the following research question:

> *Does CounterSample, a sampling-based SGD approach, converge faster than IPS-weighted SGD for LTR?*

We use the standard experimental setup for Counterfactual LTR, first described in [24]. This means that we use a *fully supervised* LTR dataset and simulate a biased *click log* according to a position-based user behavior model.

6.1 Datasets

We use two supervised LTR datasets in our experiments: `Yahoo` [11] and `Istella-s` [28]. We choose these two datasets as they complement each other in the number of items per query, which is large for `Istella-s` and small for `Yahoo`, and the sparsity of relevance feedback, which is high for `Istella-s` and low for `Yahoo` (see Table 1).

Both LTR datasets are collected on a large set of queries $Q = \{q_1, \ldots, q_m\}$. For each query q the dataset provides a set of candidate items D_q, where each item $d \in D_q$ is given in the form of a feature vector $x_{(q,d)}$ representing a query-item pair. Furthermore, for each query q the relevance grades $rel(q, d)$ are known for all $d \in D_q$. The relevance grades are scaled from 0 to 4, where 0 indicates no relevance and 4 indicates highly relevant. We note that this violates the binary relevance assumption made in Section 3.1. However, as we will see in Section 6.2, during click simulation the relevance grades are reduced to binary form which is in line with existing experimental setups for Counterfactual LTR [24].

[1] The synthesized dataset comprises 50 training samples: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{50}\}$ where each $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}]$ and each $x_{i,j} \sim \mathcal{N}(0, 1)$. We choose $\boldsymbol{w}^* = [0.973, 1.144]$ and set targets $y_i = \langle \boldsymbol{x}_i, \boldsymbol{w}^* \rangle$. For each $\boldsymbol{x}_i$ we generate a propensity $p_i \sim Uniform(0.05, 1.0)$ and use the IPS-weighted squared loss as our optimization objective: $R_{IPS}(\boldsymbol{w}) = \frac{1}{50} \sum_{i=1}^{50} \frac{1}{p_i} (\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle - y)^2$.

Table 2: $\bar{M}$ and M for varying levels of position bias (γ).

Position bias (γ):	0.5	0.75	1.0	1.25	1.5
Yahoo: $\bar{M}$	3.14	5.12	7.92	11.71	16.66
Yahoo: M	11.79	40.70	129.00	388.91	1265.55
Istella-s: $\bar{M}$	4.21	7.42	12.02	18.12	25.94
Istella-s: M	13.04	48.53	177.00	645.60	2081.04

6.2 Simulation Setup

We simulate clicks using the setup of [24]. In this setup, we repeatedly sample a query q uniformly from the dataset. The candidate items D_q for the sampled query are then sorted by a scoring function S_0, called the *logging policy* (see Section 6.3 for how S_0 is chosen). The simulation introduces position bias: items that are highly ranked by S_0 have a higher probability of being observed and thus clicked. Furthermore, the simulation has some noise: for every observed item, the probability of it being clicked is 1 if the item is relevant ($rel(q,d) \in \{3,4\}$) and 0.1 if the item is not relevant ($rel(q,d) \in \{0,1,2\}$). More formally, for every item $d \in D_q$, clicks are sampled from a Bernoulli distribution with probability:

$$P(c(d)=1) = \begin{cases} P(o(d) \mid q, D_q, S_0) & \text{if } rel(q,d) \in \{3,4\}, \\ P(o(d) \mid q, D_q, S_0) \cdot 0.1 & \text{if } rel(q,d) \in \{0,1,2\}, \end{cases} \tag{27}$$

where

$$P(o(d) \mid q, D_q, S_0) = \left(\frac{1}{rank(d \mid q, D_q, S_0)} \right)^{\gamma}, \tag{28}$$

and $\gamma \geq 0$ is a parameter controlling the severity of position bias. The above formulation is identical to the setup used in [24]. For all our experiments we simulate 1,000,000 clicks. Unless otherwise specified, we use $\gamma = 1$ as the position bias parameter. Table 2 shows the values of M and $\bar{M}$ for the simulated clicks on each of the datasets. We note that even under mild position bias ($\gamma = 0.5$), there is a significant difference between M and $\bar{M}$. The difference between these quantities becomes substantially larger as γ increases.

Simulating clicks from supervised datasets has several advantages over using existing click logs. First, by simulating clicks we can explicitly control the severity of position bias (by controlling the value of γ) and therefore test its impact in a controlled environment. Second, we can evaluate the learned rankers on the *true relevance labels* as they are provided by the supervised datasets. This means that we do not have to resort to performance estimation techniques that may be unreliable.

6.3 Choice of Logging Policy

We need to build a logging policy S_0 that can be used to rank items for our click simulation. A good candidate logging policy is one that can produce rankings of sufficient quality to generate a useful number of relevant clicks, but not perfectly optimal so that learning can still occur. To do so we train a linear ranker with full supervision (using the pairwise hinge loss formulation of Equation 3) on 0.1% of the queries for each of the datasets. Building a logging policy in this manner represents a realistic deployment scenario: practitioners of LTR systems would typically train a ranker on a small amount of manually annotated data before deploying it to collect a large amount of click data.

Table 3: Average regret (×100) for different optimizers. Smaller values indicate faster convergence. Statistically significantly lower and higher regret compared to IPS-SGD is denoted with $\triangledown$ and $\triangle$ respectively.

Optimizer:	SGD	Adam	Adagrad
Yahoo			
Biased-SGD	2.64$^{\triangle}$	2.72$^{\triangle}$	2.76$^{\triangle}$
IPS-SGD	0.41	0.97	0.64
CounterSample	0.33$^{\triangledown}$	0.35$^{\triangledown}$	0.44$^{\triangledown}$
Istella-s			
Biased-SGD	2.07$^{\triangle}$	2.04$^{\triangledown}$	2.06$^{\triangle}$
IPS-SGD	1.33	2.16	1.24
CounterSample	1.19$^{\triangledown}$	1.24$^{\triangledown}$	1.15$^{\triangledown}$

6.4 Evaluation

To measure the performance of the rankers learned by the various algorithms, we use nDCG@10 [20] on held-out test data. We denote with *nDCG@10*($\boldsymbol{w}$), the average nDCG@10 on held-out test data when items are ranked using the scoring function $S_{\boldsymbol{w}}$.

We are interested in measuring the *convergence rate* of the learning algorithms. To do so, we measure average regret in terms of nDCG@10 (with respect to the optimal model $\boldsymbol{w}^*$):

$$Regret(T) = \frac{1}{T} \sum_{t=1}^{T} \left(nDCG@10(\boldsymbol{w}^*) - nDCG@10(\bar{\boldsymbol{w}}_t) \right), \tag{29}$$

where $\bar{\boldsymbol{w}}_t = \frac{1}{t} \sum_{t'=1}^{t} \boldsymbol{w}_{t'}$ is the learned model after t iterations. To obtain the gold standard $\boldsymbol{w}^*$ we train a linear ranker with full supervision (using the relevance labels of the LTR dataset).

Measuring regret should help us confirm our theoretical results about convergence rates since lower values indicate faster convergence to the optimal solution $\boldsymbol{w}^*$. For each of our results we consider statistical significance with a t-test ($p < 0.01$).

6.5 Methods to Compare

In our experiments we compare the following methods:

- CounterSample: our sample-based method (Algorithm 2);
- IPS-SGD: IPS-weighted SGD (Algorithm 1) [2, 24]; and
- Biased-SGD: naive SGD without any propensity weighting.

We use a linear scoring function for our experiments, as this more closely matches the convexity assumptions made in our analysis:

$$S_{\boldsymbol{w}}(q,d) = \langle \boldsymbol{w}, x_{q,d} \rangle. \tag{30}$$

For each experiment and each method we tune the learning rate η to minimize regret (see Section 6.4) on held-out validation data, where we try the following values of η:

$$\eta \in \{1 \times 10^{-10}, 3 \times 10^{-10}, 1 \times 10^{-9}, \ldots, 1 \times 10^{0}, 3 \times 10^{0}\}. \tag{31}$$

7 EXPERIMENTAL RESULTS

7.1 Effect of Optimizer

First, we investigate the impact of the optimizer on the convergence rate of the different Counterfactual LTR approaches. We consider three commonly used optimization methods: Regular SGD,

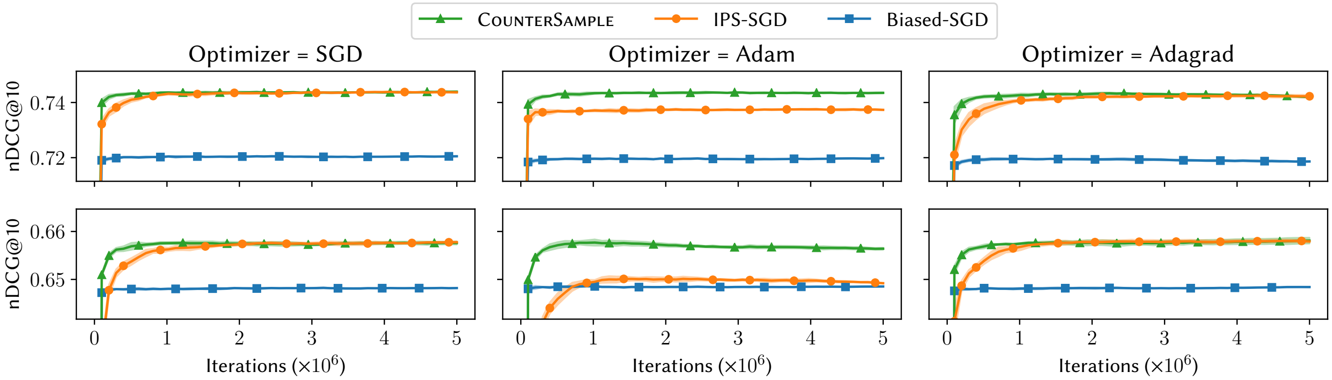

Figure 2: The learning performance on held-out test data for different optimizers (top row is Yahoo, bottom row is `Istella-s`). CounterSample is significantly faster to converge in all scenarios. For the Adam optimizer, the IPS-weighted SGD approach produces a suboptimal performance.

Adam [26] and Adagrad [13]. We apply these methods by replacing the update rule in Algorithms 1 and 2 with either the update rule from Adam or Adagrad.

In Figure 2 we plot the learning curves on held-out test data for both the Yahoo and `Istella-s` dataset. Interestingly, IPS-SGD does not work well with Adam, converging to a suboptimal solution, whereas CounterSample is able to converge to a much higher level of performance. This result is surprising as both CounterSample and IPS-SGD optimize the same unbiased objective. Recent work has shown that Adam is not guaranteed to converge to the optimal solution for some convex optimization problems and this may in part explain the behavior we observe here [32]. Regardless, we observe that in all cases CounterSample converges significantly faster than IPS-SGD. We note that the naive Biased-SGD approach converges to a lower level of performance, which is as expected since it ignores the impact of position bias. We confirm these findings by reporting the average regret in Table 3, observing a significantly lower regret for CounterSample than IPS-SGD.

Our results indicate that CounterSample is superior to IPS-weighting across all optimizers. We find that regular SGD outperforms other optimizers in the majority of cases. A possible reason for this behavior is that our scoring function is linear. Optimizers such as Adam and Adagrad may not provide significant benefits over SGD when applied to linear functions as opposed to non-linear functions (e.g., deep neural networks). We leave studying other scoring functions such as neural networks as future work.

7.2 Impact of Batch Size

In this section we investigate the effect of the batch size. We hypothesize that large batch sizes reduce the variance of individual update steps, as many gradients are averaged in a single update step, and as a result the convergence rate of CounterSample and IPS-SGD should be comparable. We try batch sizes 10, 20 and 50.

We plot the learning curves for varying batch sizes in Figure 3. Once again we find that, unsurprisingly, Biased-SGD converges to a suboptimal solution. For both datasets we observe that CounterSample is able to converge faster than IPS-SGD, regardless of the chosen batch size. For the `Istella-s` dataset, CounterSample is able to converge to a slightly higher level of performance than IPS-SGD when using a batch size of 50. The average regret in Table 4 suggests that the convergence rate of CounterSample is not affected by batch size; it is able to converge faster than IPS-SGD in all cases.

Table 4: Average regret (×100) for different batch sizes. Statistical significance is denoted the same as Table 3.

Batch size:	10	20	50
Yahoo			
Biased-SGD	2.49$^\triangle$	2.36$^\triangle$	2.31$^\triangle$
IPS-SGD	0.41	0.42	0.44
CounterSample	0.33$^\triangledown$	0.34$^\triangledown$	0.37$^\triangledown$
`Istella-s`			
Biased-SGD	2.07$^\triangle$	2.08$^\triangle$	2.07$^\triangle$
IPS-SGD	1.34	1.31	1.32
CounterSample	1.20$^\triangledown$	1.21$^\triangledown$	1.21$^\triangledown$

Table 5: Average regret (×100) for different levels of position bias γ. Statistical significance is denoted the same as Table 3.

Position bias (γ):	0.5	0.75	1.0	1.25	1.5
Yahoo					
Biased-SGD	0.89$^\triangle$	1.78$^\triangle$	2.64$^\triangle$	3.32$^\triangle$	3.83$^\triangle$
IPS-SGD	0.34	0.35	0.41	0.56	0.75
CounterSample	0.30	0.27$^\triangledown$	0.33$^\triangledown$	0.38$^\triangledown$	0.51$^\triangledown$
`Istella-s`					
Biased-SGD	1.37$^\triangle$	1.75$^\triangle$	2.07$^\triangle$	2.30$^\triangle$	2.53$^\triangle$
IPS-SGD	1.09	1.12	1.33	1.53	1.79
CounterSample	1.05	1.08	1.19$^\triangledown$	1.26$^\triangledown$	1.44$^\triangledown$

7.3 Severity of Position Bias

Finally, we look at the impact of position bias, controlled by the position bias parameter γ. Position bias has an effect on the nature of the clicks collected and changes the distribution of propensity scores (see Table 2). For large γ, the propensity scores will be heavily skewed: the majority of observations and propensities will be on the top-ranked items while lower-ranked items are only very rarely

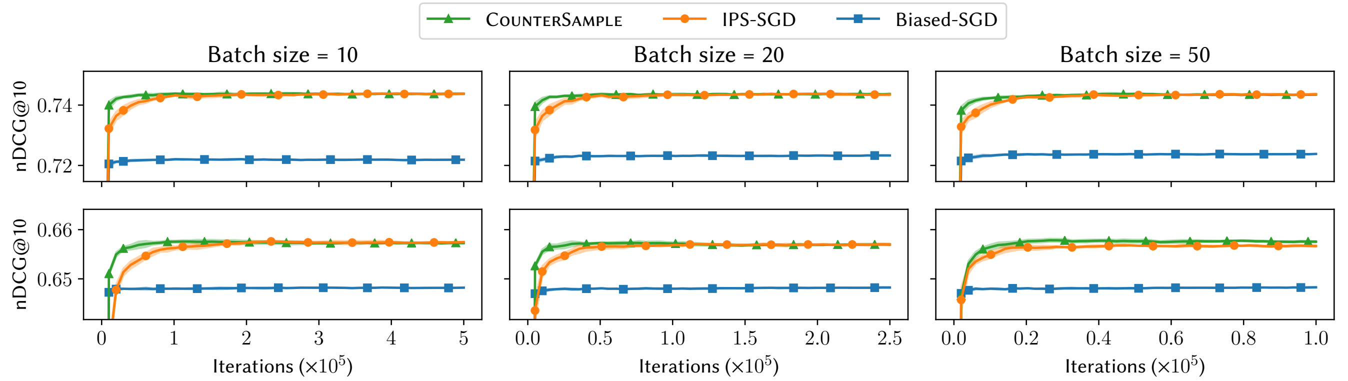

Figure 3: The learning performance on held-out test data for different batch sizes (top row is `Yahoo`, bottom row is `Istella-s`). CounterSample is faster to converge in all scenarios, however the differences are less pronounced for larger batch sizes.

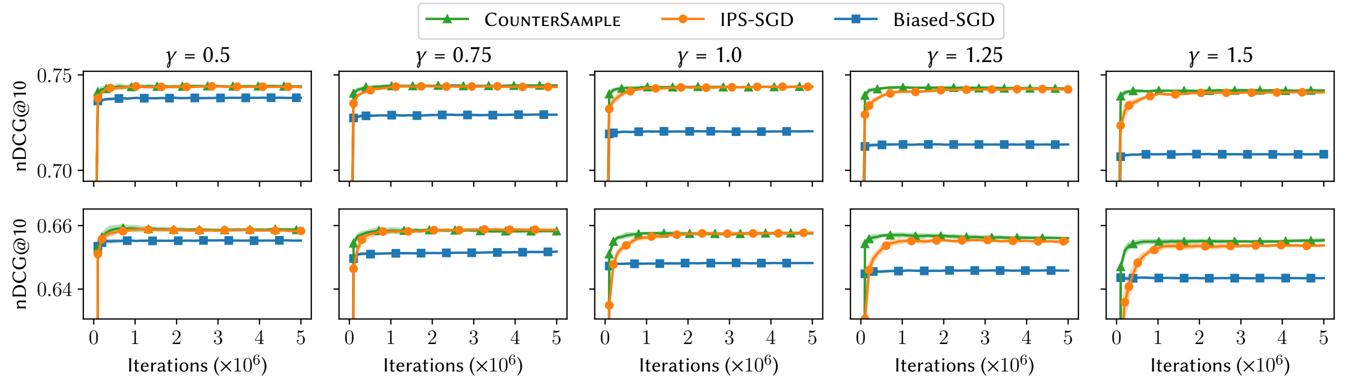

Figure 4: The learning performance on held-out test data for varying levels of position bias (top row is `Yahoo`, bottom row is `Istella-s`). CounterSample's convergence rate is robust to larger values of γ, whereas IPS-SGD suffers when γ is large.

observed and clicked, resulting in more extreme IPS weights for those clicks. Conversely, a small γ makes the propensity scores more heavy tailed, generating more observations on lower ranked items and consequently more clicks on those items with less extreme IPS weights. We expect that, as we increase γ, CounterSample should outperform IPS-SGD in terms of convergence rate since in this case $M \gg \bar{M}$. Conversely, for smaller values of γ we expect that the methods perform comparably.

Figure 4 provides learning curves for various levels of γ. We observe that the performance of Biased-SGD goes up as γ goes down, which is in line with our expectations since smaller values of γ result in less position bias. In all cases IPS-SGD and CounterSample perform strictly better than Biased-SGD. The convergence of CounterSample is comparable to IPS-SGD for smaller γ, but as γ grows, CounterSample is significantly faster to converge than IPS-SGD. This confirms our expectation that CounterSample is able to reliably handle situations where $M \gg \bar{M}$, i.e. when there are more extreme IPS weights. Table 5 confirms these findings in terms of average regret: for larger values of γ, CounterSample is able to obtain significantly lower regret than competing approaches.

7.4 Discussion

Finally, we reflect on the research question posed in Section 6: *Does CounterSample, a sampling-based SGD approach, converge faster than IPS-weighted SGD for LTR?* We answer our research question positively: CounterSample consistently converges faster than IPS-SGD – across optimizers, batch sizes and different levels of position bias (γ). These findings support the theoretical results obtained in Sections 4 and 5. In some cases, for example when using the Adam optimizer, CounterSample is not only able to converge faster but able to converge to a higher level of performance than IPS-SGD.

8 CONCLUSION

We have studied the convergence rate for Stochastic Gradient Descent (SGD) approaches in Counterfactual Learning to Rank (LTR). A common approach to Counterfactual LTR is IPS-weighted SGD, where the loss or gradients are scaled by IPS weights. We prove that, for IPS-weighted SGD, the IPS weights play an important role in the convergence rate: the time to converge is slowed by a factor $O(M^2)$ where M is the maximum IPS weight in the dataset.

To overcome the slow convergence of IPS-weighted SGD we propose a sample-based Counterfactual LTR learning algorithm called CounterSample. We prove that CounterSample reduce the convergence rate slowdown from $O(M^2)$ to $O(\bar{M}^2)$ where $\bar{M}$ is the average IPS weight in the dataset. When $M \gg \bar{M}$, this improvement leads to significantly faster convergence of the learning algorithm.

We support our findings with extensive experimentation across a number of biased LTR scenarios, comparing CounterSample to

SGD with and without IPS weighting. In all cases CounterSample is able to converge faster than standard IPS-weighted SGD. In some scenarios CounterSample is even able to converge to a better level of performance than IPS-weighted SGD.

There are several directions for future work: First, the convexity assumptions made in the analysis may not hold in practice, particularly when implementing deep neural networks. Showing the convergence rate of IPS-weighted SGD for non-convex problems remains an open problem. Second, optimizing an IPS-weighted objective is arguably the simplest approach to Counterfactual LTR and in future work we would like to consider more sophisticated objectives such as self-normalized IPS [39] and variance regularization [38]. Third, our experiments are conducted on click simulations, giving us experimental control to test our hypotheses. We leave applying CounterSample to large-scale industrial click logs as future work. Finally, our work assumes that propensity scores are known a priori which is not always realistic. Robustness against misspecified propensity scores remains an open problem.

CODE AND DATA

To facilitate reproducibility of our work, we are sharing all resources used in this paper at http://github.com/rjagerman/sigir2020.

ACKNOWLEDGMENTS

We thank Chang Li, Harrie Oosterhuis and Ilya Markov for helpful discussions and feedback. We thank the anonymous reviewers for their feedback. This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) under project nr 612.001.551 and the Innovation Center for AI (ICAI). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, et al. 2016. TensorFlow: A System for Large-scale Machine Learning. In *USENIX OSDI*. 265–283.

[2] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A General Framework for Counterfactual Learning-to-Rank. In *SIGIR*. ACM, 5–14.

[3] Alekh Agarwal, Martin J Wainwright, Peter L Bartlett, and Pradeep K Ravikumar. 2009. Information-theoretic Lower Bounds on the Oracle Complexity of Convex Optimization. In *NIPS*. 1–9.

[4] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating Position Bias without Intrusive Interventions. In *WSDM*. ACM, 474–482.

[5] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased Learning to Rank with Unbiased Propensity Estimation. In *SIGIR*. ACM, 385–394.

[6] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. 2015. Variance Reduction in SGD by Distributed Importance Sampling. *arXiv preprint arXiv:1511.06481* (2015).

[7] Muhammad Umer Anwaar, Dmytro Rybalko, and Martin Kleinsteuber. 2019. Counterfactual Learning from Logs for Improved Ranking of E-Commerce Products. *arXiv preprint arXiv:1907.10409* (2019).

[8] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from user interactions in personal search via attribute parameterization. In *WSDM*. ACM, 791–799.

[9] Ben Carterette and Praveen Chandar. 2018. Offline Comparative Evaluation with Incremental, Minimally-invasive Online Feedback. In *SIGIR*. ACM, 705–714.

[10] Praveen Chandar and Ben Carterette. 2018. Estimating Clickthrough Bias in the Cascade Model. In *CIKM*. ACM, 1587–1590.

[11] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the Learning to Rank Challenge*. 1–24.

[12] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-bias Models. In *WSDM*. ACM, 87–94.

[13] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *JMLR* 12, Jul (2011), 2121–2159.

[14] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. 2019. Intervention Harvesting for Context-dependent Examination-bias Estimation. In *SIGIR*. 825–834.

[15] Artem Grotov and Maarten de Rijke. 2016. Online Learning to Rank for Information Retrieval: SIGIR 2016 Tutorial. In *SIGIR*. ACM, 1215–1218.

[16] Mohammad Al Hasan, Nish Parikh, Gyanit Singh, and Neel Sundaresan. 2011. Query Suggestion for E-commerce Sites. In *WSDM*. ACM, 765–774.

[17] Elad Hazan, Amit Agarwal, and Satyen Kale. 2007. Logarithmic Regret Algorithms for Online Convex Optimization. *Machine Learning* 69, 2-3 (2007), 169–192.

[18] Elad Hazan and Satyen Kale. 2014. Beyond the Regret Minimization Barrier: Optimal Algorithms for Stochastic Strongly-convex Optimization. *JMLR* 15, 1 (2014), 2489–2512.

[19] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. In *SIGIR*. ACM, 15–24.

[20] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *TOIS* 20, 4 (2002), 422–446.

[21] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *KDD*. ACM, 133–142.

[22] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2017. Accurately Interpreting Clickthrough Data as Implicit Feedback. In *SIGIR Forum*, Vol. 51. Acm New York, NY, USA, 4–11.

[23] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep Learning with Logged Bandit Feedback. In *ICLR*.

[24] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *WSDM*. ACM, 781–789.

[25] Angelos Katharopoulos and François Fleuret. 2018. Not All Samples are Created Equal: Deep Learning with Importance Sampling. *arXiv preprint arXiv:1803.00942* (2018).

[26] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

[27] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[28] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. 2016. Post-learning Optimization of Tree Ensembles for Efficient Ranking. In *SIGIR*. ACM, 949–952.

[29] Deanna Needell, Rachel Ward, and Nati Srebro. 2014. Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz Algorithm. In *NIPS*. 1017–1025.

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS*.

[31] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. 2011. Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization. *arXiv preprint arXiv:1109.5647* (2011).

[32] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2019. On the Convergence of Adam and Beyond. *arXiv preprint arXiv:1904.09237* (2019).

[33] Herbert Robbins and Sutton Monro. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics* (1951), 400–407.

[34] Walter Rudin. 1987. *Real and Complex Analysis*. McGraw-Hill.

[35] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

[36] Ohad Shamir and Tong Zhang. 2013. Stochastic Gradient Descent for Non-smooth Optimization: Convergence Results and Optimal Averaging Schemes. In *ICML*. 71–79.

[37] Adith Swaminathan and Thorsten Joachims. 2015. Batch Learning from Logged Bandit Feedback Through Counterfactual Risk Minimization. *JMLR* 16, 1 (2015), 1731–1755.

[38] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual Risk Minimization: Learning from Logged Bandit Feedback. In *ICML*. 814–823.

[39] Adith Swaminathan and Thorsten Joachims. 2015. The Self-normalized Estimator for Counterfactual Learning. In *NIPS*. 3231–3239.

[40] Alastair J Walker. 1974. New Fast Method for Generating Discrete Random Numbers with Arbitrary Frequency Distributions. *Electronics Letters* 10, 8 (1974), 127–128.

[41] Alastair J Walker. 1977. An Efficient Method for Generating Discrete Random Variables with General Distributions. *TOMS* 3, 3 (1977), 253–256.

[42] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *SIGIR*. ACM, 115–124.

[43] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *WSDM*. ACM, 610–618.

[44] Peilin Zhao and Tong Zhang. 2015. Stochastic Optimization with Importance Sampling for Regularized Loss Minimization. In *ICML*. 1–9.