Off-policy Learning for Multiple Loggers

Li He
JD.com
heli@amss.ac.cn

Long Xia
JD.com
xialong@jd.com

Wei Zeng
Institute of Computing Technology, CAS
zengwei@software.ict.ac.cn

Zhi-Ming Ma
Academy of Mathematics and Systems Science, CAS
mazm@amt.ac.cn

Yihong Zhao
JD.com
ericzhao@jd.com

Dawei Yin
JD.com
yindawei@acm.org

ABSTRACT

It is well known that the historical logs are used for evaluating and learning policies in interactive systems, e.g. recommendation, search, and online advertising. Since direct online policy learning usually harms user experiences, it is more crucial to apply off-policy learning in real-world applications instead. Though there have been some existing works, most are focusing on learning with one single historical policy. However, in practice, usually a number of parallel experiments, e.g. multiple AB tests, are performed simultaneously. To make full use of such historical data, learning policies from multiple loggers becomes necessary. Motivated by this, in this paper, we investigate off-policy learning when the training data coming from multiple historical policies. Specifically, policies, e.g. neural networks, can be learned directly from multi-logger data, with counterfactual estimators. In order to understand the generalization ability of such estimator better, we conduct generalization error analysis for the empirical risk minimization problem. We then introduce the generalization error bound as the new risk function, which can be reduced to a constrained optimization problem. Finally, we give the corresponding learning algorithm for the new constrained problem, where we can appeal to the minimax problems to control the constraints. Extensive experiments on benchmark datasets demonstrate that the proposed methods achieve better performances than the state-of-the-arts.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Theory of computation** → *Sample complexity and generalization bounds*; • **Computing methodologies** → Learning from implicit feedback.

KEYWORDS

Off-policy Learning; Multiple Loggers; Log Data

1 INTRODUCTION

In many interactive systems, such as search engines, recommender systems, and ad-placement [3, 7], large batches of logs are collected from the past periods for model improvement. Usually, the interactive process can be formulated as follows. Given an input (or context) from users, the system draws an output (or action) based on its current policy. Then we observe feedback of the chosen output for that input. The logged data are counterfactual since they only provide partial information. For example, in a movie recommendation system, we can only observe the feedback for the output chosen by the system (e.g., the recommended movie) but not for all the other movies that the system could have recommended potentially. Although the logs are biased, they are informative and can be exploited for many purposes.

One application of logged data is *off-policy evaluation*, which is to evaluate new given policies offline [26]. Since the online evaluation is prohibitively expensive and may harm the user experiences, leveraging such logged data could provide a useful alternative. Nevertheless, direct evaluation over the logged data which are collected from a historical policy, leads to a biased estimation. To address this issue, many estimators have been proposed [12, 17, 38, 46]. Another important application of such logs is to learn policy with better performance, also known as *off-policy learning* [7, 45, 50]. For example, the counterfactual estimator had been used for learning in advertisement applications [7].

Though there have been some existing works, most are focusing on learning with one single historical policy. However, in practice, usually a number of parallel experiments, e.g. multiple AB tests, are performed simultaneously. This typically generates the logged data from many policies. To make full use of such historical data, learning policies from multiple loggers becomes an important problem. More recently, there have been few preliminary investigations for the case of multiple loggers. The work [1] proposed three estimators for off-policy evaluation, which are named as naive, balanced, and weighted inverse propensity score (IPS) estimators. However, it focuses on the evaluation problem but not on the learning. Following [1], a weighted counterfactual risk minimization (WCRM)

principle was proposed in [42], which combined the weighted IPS estimator with counterfactual risk minimization principle. While they give an empirical sample variance regularization, it results in a more difficult optimization due to its dependency on the whole training data.

In this paper, we investigate off-policy learning in the setting of multiple loggers. Specifically, we study the popular multi-logger counterfactual estimators, and apply them to the off-policy learning, where the popular model, e.g. neural networks based policies, could be learned directly. To better leverage such counterfactual estimators, we conduct generalization error analysis for the empirical risk minimization problem. Then, based on the analysis, we propose to optimize the generalization error bound instead to improve the generalization ability. The optimization of generalization error bound is further reduced to a constrained optimization problem, which is not uncommon and is kind of related with distributionally robust learning [5, 30]. Finally, we propose the learning algorithm for the new constrained problem, appealing to the minimax problems for the constraints. We evaluate our new methods on three benchmark datasets. Empirically, our new methods perform better than the state-of-the-arts, and the results confirmed the theoretical analyses.

To sum up, the main contributions of this paper are summarized as follows.

- Theoretically, we conduct generalization error analyses for popular counterfactual estimators in multi-logger setting;
- Based on the generalization error analyses, we use the generalization error bound as the new risk objectives and formulate them into constrained optimization problems;
- We provide corresponding learning algorithms for the new constrained optimization problems, appealing to the minimax problems to control the constraints;
- Empirically, we carry out experiments and analyses on three benchmark datasets. The results show that our new methods improve over the state-of-the-art methods.

The rest of the paper is organized as follows. We introduce the background of off-policy learning from multiple historical policies and review some related works in Section 2. We conduct the generalization error analysis for λ-weighted risk estimator in Section 3. The constrained learning methods and their corresponding algorithms are proposed in Section 4. Experiments are reported in Section 5 and a variant of estimator is provided in Section 6.

2 PRELIMINARIES

In this section, we describe the off-policy learning problem of multiple loggers and review some related works.

2.1 Problem Setting

We first recall how an interactive system works. Specifically, given an input (or context) $x \in \mathcal{X}$, which is drawn from an unknown distribution $P(\mathcal{X})$, the system selects an output (or action) $y \in \mathcal{Y}$ based on existing policy $h_0(\mathcal{Y}|x) : \mathcal{X} \mapsto \mathcal{Y}$. We denote the probability assigned by $h_0(\mathcal{Y}|x)$ to y as $h_0(y|x)$. Then we observe feedback for the couple (x, y) from an unknown function $\delta : \mathcal{X} \times \mathcal{Y} \to [0, L]$. The lower the value of $\delta(x, y)$, the higher the user satisfaction with this given output y for the input x.

To perform the off-policy evaluation, we need to consider a specific risk objective. Usually, the risk of a new given policy $h(\mathcal{Y}|x)$ can be defined as

$$R(h) = \mathbb{E}_{x \sim P(\mathcal{X}), y \sim h(\mathcal{Y}|x)} [\delta(x, y)].$$

Due to the distribution mismatch between the policy $h(\mathcal{Y}|x)$ and the historical policy $h_0(\mathcal{Y}|x)$, we apply the importance sampling technique [10, 38]. Therefore, the risk can be rewritten as

$$R(h) = \mathbb{E}_{x \sim P(\mathcal{X}), y \sim h_0(\mathcal{Y}|x)} \left[\frac{h(y|x)}{h_0(y|x)} \delta(x, y) \right].$$

In addition, since the distribution $P(\mathcal{X})$ is unknown, we have to use the empirical estimator. Assume we have a dataset from the historical policy $h_0(\mathcal{Y}|x)$, denoted as

$$\mathcal{D} = \{(x_1, y_1, \delta_1, p_1), \dots, (x_n, y_n, \delta_n, p_n)\}$$

where $\delta_i \equiv \delta(x_i, y_i)$ and $p_i \equiv h_0(y_i|x_i)$, $i \in \{1, 2, \dots, n\}$. We can use the following unbiased empirical estimator

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} \frac{h(y_i|x_i)}{h_0(y_i|x_i)} \delta_i,$$

for the expected loss $R(h)$. This is the widely used inverse propensity score (IPS) approach [38].

In this paper, we study off-policy learning in the multi-logger setting, i.e., learning a policy that has low risk by using the logs from multiple policies. This is practical and necessary as the policy gets updated frequently in most online systems. Denote the logs obtained from each logging policy h_j with

$$\mathcal{D}^j = \left\{ (x_1^j, y_1^j, \delta_1^j, p_1^j), \dots, (x_{n_j}^j, y_{n_j}^j, \delta_{n_j}^j, p_{n_j}^j) \right\},$$

where $x_i^j \sim P(\mathcal{X})$ and $y_i^j \sim h_j(\mathcal{Y}|x_i^j)$ for $j \in \{1, 2, \dots, J\}, i \in \{1, 2, \dots, n_j\}$. The feedback $\delta_i^j \equiv \delta(x_i^j, y_i^j)$ and the logging probability $p_i^j \equiv h_j(y_i^j|x_i^j)$. Therefore, we obtain a larger dataset $\mathcal{D} \equiv \cup_{j=1}^{J} \mathcal{D}^j$. Note that we can assume that all of the logging policies have the same input and output spaces. For simplicity, we denote $[J] = \{1, 2, \dots, J\}$ and $n = \sum_{j=1}^{J} n_j$. Sometimes, we will use the abbreviations h and h_j for $h(y|x)$ and $h_j(y|x)$, respectively.

Let us briefly review two recent related works below .

2.1.1 Direct Learning Principle.
In the case of multiple loggers, the work [1] proposed an estimator for off-policy evaluation, which is called naive IPS estimator. Its definition is as follows.
Naive IPS Estimator

$$\hat{R}_{naive}(h) = \frac{1}{n} \sum_{j=1}^{J} \sum_{i=1}^{n_j} \frac{h(y_i^j|x_i^j)}{h_j(y_i^j|x_i^j)} \delta_i^j,$$

This is an unbiased estimator when the logging policies have a full support for the target policy.

However, as stated in [1], it suffers when they diverge to a degree where throwing away data lowers the estimator's variance. As a result, they proposed λ-weighted IPS estimator, which is also unbiased but has a smaller variance than the naive IPS estimator.
λ-**Weighted IPS Estimator**

$$\hat{R}_{\lambda}(h) = \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{n_j} \frac{h(y_i^j|x_i^j)}{h_j(y_i^j|x_i^j)} \delta_i^j,$$

where $\lambda_j \geq 0$ and $\sum_{j=1}^{J} \lambda_j n_j = 1$. When taking $\lambda_j = \frac{1}{n}$, it reduces to the naive IPS estimator. If imposing $\lambda_j = \lambda_j^* = \frac{1}{\sigma_\delta^2(h||h_j) \sum_{j=1}^{J} \frac{n_j}{\sigma_\delta^2(h||h_j)}}$, it becomes the weighted IPS estimator. The term $\sigma_\delta^2(h||h_j)$ is defined as the divergence from h_j to h in terms of the naive IPS estimator variance, which is estimated by the empirical divergence estimator in [1].

Once having an estimator, we can carry out off-policy learning by solving

$$\min_h \hat{R}_\lambda(h).$$

While it is simple and natural, direct learning from the counterfactual estimators and choosing the empirical optimal minimizer still have several pitfalls [45], such as it may have vastly different variances for two loggers. An improved learning method for multiple loggers is needed. In previous works [42, 45], the learned policies are usually formulated as stochastic softmax rules. In this paper, we adopt neural network to express the learned policy.

2.1.2 WCRM Principle. Following the above work, a weighted counterfactual risk minimization (WCRM) principle was proposed in [42]. The WCRM objective is

$$\arg\min \sum_{j=1}^{J} \lambda_j^* \sum_{i=1}^{n_j} u_i^j(h) + \lambda \sqrt{\frac{\hat{Var}\left(\lambda_j^* n_j u_i^j(h)\right)}{n}},$$

$$\hat{Var}\left(\lambda_j^* n_j u_i^j(h)\right) = \frac{1}{n-1} \sum_{j=1}^{J} \sum_{i=1}^{n_j} \left(\lambda_j^* n_j u_i^j - \frac{1}{n} \sum_{j=1}^{J} \sum_{i=1}^{n_j} \lambda_j^* n_j u_i^j\right)^2,$$

where $u_i^j(h) = \frac{h(y_i^j|x_i^j)}{h_j(y_i^j|x_i^j)} \delta_i^j$. The divergence $\sigma_\delta^2(h||h_j)$ is estimated by a self-normalized divergence estimator. The variance term in WCRM principle depends on the whole dataset, which results in a more difficult stochastic optimization.

2.2 Related Work

Our paper is related to off-policy evaluation, which has many applications in practice, such as recommendation, search engine, and learning to rank [20, 21, 25, 27, 47, 49]. The counterfactual estimator can date back to the inverse propensity score (IPS) estimator [16, 38]. It uses importance weighting technique, which solves the mismatch between the training distribution and the test distribution. In [10], the authors firstly give the theoretical learning bound analysis for importance weighting. However, the IPS estimator may have large or unbounded variance. As a result, many new estimators have arisen, such as truncated importance sampling [17], doubly robust estimator [12], and self-normalized estimator [46]. The doubly robust estimator was first developed for regression [8], then it had been brought to contextual bandits. The self-normalized estimator was developed to avoid propensity overfitting problem. Both doubly robust and self-normalized estimators fall into the method of control variate [34]. Recently, researchers proposed many counterfactual estimators with smaller mean square error [31, 43].

The off-policy evaluation can be regarded as counterfactual reasoning for analyzing the causal effect of a new treatment from previous data [7, 39]. It can also be viewed as a special case of off-policy evaluation in reinforcement learning [13, 19, 29, 36, 44, 48],

which also has been applied in real applications [51–56]. Exploiting logs is important in multi-armed bandit and its variants, such as contextual bandit [40, 41].

The counterfactual estimator is often the first step for learning problem. For off-policy learning, there are also some related works [6, 7, 41, 45, 46, 50]. For example, the work [7] used the counterfactual estimator for learning in advertisement application. In [45], the authors developed the counterfactual risk minimization (CRM) principle for batch learning from bandit feedback (BLBF). The key idea lies in controlling the differences in variance between different target policies. They derived an algorithm, policy optimizer for exponential models (POEM) , for learning stochastic linear rules for structured output prediction. The work [46] proposed the Norm-POEM algorithm by combining the self-normalized estimator with POEM algorithm. The work that is the most related to ours is [1], in their work, they pointed out the sub-optimality of the naive IPS estimator and proposed two alternative estimators: balanced IPS estimator and weighted IPS estimator. Recently, in [42], the authors combined the weighted IPS estimator with counterfactual risk minimization principle for learning from multiple loggers.

3 GENERALIZATION ERROR ANALYSIS

Now that we have an empirical risk minimization (ERM) problem, we can study its generalization error bound, which is frequently-used in supervised learning. In this section, we give the generalization error analyses for λ-weighted IPS estimator, while the analysis for naive IPS estimator can be obtained by letting $\lambda_j = \frac{1}{n}$.

Before conducting the generalization error analysis, we first borrow one lemma from [10] as follows.

LEMMA 3.1. *([10]) For a random variable z, let Q and Q_0 be two probability measures, assume q and q_0 are two probability density functions of Q and Q_0, respectively. Let l be a loss function bounded in $[0, 1]$. Let $w = \frac{q}{q_0}$, then the following results hold:*

$$\mathbb{E}_{z \sim Q_0}[w(z)] = 1, \quad \mathbb{E}_{z \sim Q_0}[w^2(z)] = d_2(q||q_0),$$
$$\mathbb{E}_{z \sim Q_0}\left[w^2(z)l^2(z)\right] \leq d_2(q||q_0),$$

where $d_2(q||q_0) \equiv 2^{D_2(q||q_0)}$ and $D_2(q||q_0)$ is Rényi divergence [37].

Based on lemma 3.1, we can obtain an upper bound for the second moment of the importance weighted loss, i.e.,

$$\mathbb{E}_{x \sim P(\mathcal{X}), y \sim h_0(\mathcal{Y}|x)} \left[\left(\frac{h(y|x)}{h_0(y|x)} \delta(x, y) \right)^2 \right] \leq L^2 d_2(h(y|x)||h_0(y|x); P(x)),$$

where $d_2(h(y|x)||h_0(y|x); P(x)) \triangleq d_2(P(x)h(y|x)||P(x)h_0(y|x)) = \int_{\mathcal{X}, \mathcal{Y}} \frac{h^2(y|x)}{h_0(y|x)} P(x) \mathrm{d}x \mathrm{d}y$. This can be easily derived by some substitutions, i.e., letting $z = (x, y), q(z) = P(x)h(y|x), q_0(z) = P(x)h_0(y|x)$, and $l(z) = \delta(x, y) \in [0, L]$, so here we omitted the proof [50].

Now we are ready to give our main theorem and the sketch of the proof as follows.

THEOREM 3.2. *Let $R(h)$ be the risk of a new policy h on the loss function δ, and $\hat{R}_\lambda(h)$ be the λ-weighted empirical risk. Assume the divergence is bounded by M_j, i.e., $d_2(h||h_j) \leq d_\infty(h||h_j) = M_j$ [1],*

---

[1]The divergence $d_\infty(h||h_j) \equiv \sup_y \frac{h(y|x)}{h_j(y|x)}$.

$j \in [J]$ and denote $M_\lambda \equiv max_j\{\lambda_j M_j\}$. Then, for any $\eta > 0$, with probability at least $1 - \eta$, the following bound holds:

$$R(h) \le \hat{R}_\lambda(h) + \frac{2LM_\lambda \log \frac{1}{\eta}}{3} + L\sqrt{2\sum_{j=1}^{J} n_j \lambda_j^2 d_2(h||h_j; P(x)) \log \frac{1}{\eta}}.$$

Proof. By the definition of λ_j, we have

$$R(h) - \hat{R}_\lambda(h) = \sum_{j=1}^{J}\sum_{i=1}^{n_j} \lambda_j \left[R(h) - \frac{h(y_i^j|x_i^j)}{h_j(y_i^j|x_i^j)} \delta(x_i^j, y_i^j) \right],$$

Denote $Z_i^j = R(h) - \frac{h(y_i^j|x_i^j)}{h_j(y_i^j|x_i^j)}\delta(x_i^j, y_i^j)$ and $Z = R(h) - \frac{h(y|x)}{h_j(y|x)}\delta(x, y)$.
We can obtain that $\mathbb{E}_{x\sim P(X), y\sim h_j(\mathcal{Y}|x)}Z = 0$ and $|Z| \le M_j L$.

In addition, by applying lemma 3.1, we have

$$\mathbb{E}_{x\sim P(X), y\sim h_j(\mathcal{Y}|x)}\left[\left(\frac{h(y|x)}{h_j(y|x)}\delta(x, y)\right)^2\right] \le L^2 d_2(h(y|x)||h_j(y|x); P(x)).$$

Thus, we have the following bound for the second moment of Z,

$$\mathbb{E}_{x\sim P(X), y\sim h_j(\mathcal{Y}|x)}Z^2 \le L^2 d_2(h(y|x)||h_j(y|x); P(x)).$$

Applying Bernstein's inequality [4], we have

$$\mathbb{P}\left(\sum_{j=1}^{J}\sum_{i=1}^{n_j}\lambda_j Z_i^j > \epsilon\right)$$

$$\le \exp\left(-\frac{\frac{1}{2}\epsilon^2}{\sum_{j=1}^{J}\sum_{i=1}^{n_j}\mathbb{E}_{x\sim P(X), y\sim h_j(\mathcal{Y}|x)}(\lambda_j Z_i^j)^2 + \frac{1}{3}LM_\lambda\epsilon}\right)$$

$$\le \exp\left(-\frac{\frac{1}{2}\epsilon^2}{\sum_{j=1}^{J}\sum_{i=1}^{n_j}\lambda_j^2 L^2 d_2(h(y|x)||h_j(y|x); P(x)) + \frac{1}{3}LM_\lambda\epsilon}\right),$$

where $M_\lambda = max_j\{\lambda_j M_j\}$.

Let the right hand be equal to η, we can obtain an quadratic function of ϵ. With some calculations and using $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$, we obtain that

$$\epsilon \le \frac{2LM_\lambda \log \frac{1}{\eta}}{3} + L\sqrt{2\sum_{j=1}^{J} n_j \lambda_j^2 d_2(h(y|x)||h_j(y|x); P(x)) \log \frac{1}{\eta}}.$$

Therefore, the following inequality

$$R(h) \le \hat{R}_\lambda(h)$$

$$+ \frac{2LM_\lambda \log \frac{1}{\eta}}{3} + L\sqrt{2\sum_{j=1}^{J} n_j \lambda_j^2 d_2(h(y|x)||h_j(y|x); P(x)) \log \frac{1}{\eta}}$$

holds with probability at least $1 - \eta$. □

When taking λ_j as λ_j^*, we obtain the generalization error bound of weighted IPS estimator. When letting $\lambda_j = \frac{1}{n}, \forall j$, λ-weighted IPS estimator reduces to the naive IPS estimator. Therefore, we give the following corollary.

COROLLARY 3.3. *Let $R(h)$ be the risk of a new policy h on the loss function δ, and $\hat{R}_{naive}(h)$ be the naive empirical risk. Assume that the divergence is bounded by M_j, i.e., $d_2(h||h_j) \le d_\infty(h||h_j) = M_j, j \in [J]$ and denote $M_{naive} \equiv max_j\{M_j\}$. Then, for any $\eta > 0$, with probability at least $1 - \eta$, the following bound holds:*

$$R(h) \le \hat{R}_{naive}(h)$$

$$+ \frac{2LM_{naive} \log \frac{1}{\eta}}{3n} + L\sqrt{\frac{2\sum_{j=1}^{J} n_j d_2(h||h_j; P(x)) \log \frac{1}{\eta}}{n^2}}.$$

In the next section, we show how to apply the generalization error analyses to the off-policy learning.

4 CONSTRAINED LEARNING METHODS

In this section, we introduce our new constrained learning methods and study how to apply them in practice. Based on the results in Section 3, we propose to use the generalization error bound as the new regularized objectives. Specifically, we have

$$\min_h \hat{R}_{weighted}(h) + \beta\sqrt{\sum_{j=1}^{J} n_j(\lambda_j^*)^2 d_2(h||h_j; P(x))}, \quad (1)$$

$$\min_h \hat{R}_{naive}(h) + \beta\sqrt{\frac{\sum_{j=1}^{J} n_j d_2(h||h_j; P(x))}{n^2}}, \quad (2)$$

for the weighted IPS estimator and the naive IPS estimator, respectively. The parameter $\beta = \sqrt{2L^2 \log \frac{1}{\eta}}$ is to trade-off the bias and the regularization, which is challenging to be set empirically and solving the optimization problem. Thus, inspired by the work [50], we study a constrained optimization problem instead. For eq.(1) and eq.(2), we consider the following general constrained problem

$$\min_h \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{n_j} \frac{h(y_i^j|x_i^j)}{h_j(y_i^j|x_i^j)}\delta_i^j \quad (3)$$

$$s.t. \sum_{j=1}^{J} n_j \lambda_j^2 d_2(h||h_j; P(x)) \le \frac{\rho}{n^2},$$

where ρ is a pre-defined constant. Similarly, it corresponds to the formulation of the naive IPS estimator with $\lambda_j = \frac{1}{n}, j \in [J]$.

In the next sections, we first review the derivations of variational divergence minimization. Then we give our algorithms for the proposed constrained formulations.

4.1 About the Constraints

For eq.(3), we have to analyze the constraint, where the term $d_2(h||h_j; P(x))$ is defined as

$$d_2(h||h_j; P(x)) = \int_{X, \mathcal{Y}} \frac{h^2(y|x)}{h_j(y|x)}P(x)\mathrm{d}x\mathrm{d}y.$$

With some derivations, we can obtain

$$d_2(h||h_j; P(x)) = D_f(h||h_j; P(x)) + 1,$$

where $D_f(h||h_j; P(x)) = \int_X D_f(h||h_j)P(x)\mathrm{d}x$ and $f(t) = t^2 - 1$. Here the term $D_f(h||h_j)$ is the f-divergence [11]. Hence, if we are able to control the part of $D_f(h||h_j; P(x))$ well, we can obtain the upper bound of $d_2(h||h_j; P(x))$ immediately. Thus, we omit the constant 1 without loss of generality.

By following the techniques in [32, 33], we obtain the lower bound of $D_f(h||h_j; P(x))$,

$$D_f(h||h_j; P(x)) \ge sup_{T\in\mathcal{T}}\left\{\mathbb{E}_{x, y\sim h}T(x, y) - \mathbb{E}_{x, y\sim h_j}f^*(T(x, y))\right\}, \quad (4)$$

where f^* is a convex conjugate function of f [14] [2], and $\mathcal{T}$ is an arbitrary class of functions $T : X \to \mathbb{R}$. For the inequality, under mild conditions on f [32], the bound is tight for $T^*(x) = f'(\frac{h}{h_j})$, where f' is the first derivative of f. On the other hand, we can

---

[2]The convex conjugate function is defined as $f^*(t) = \sup_{u\in dom_f}\{ut - f(u)\}$.

Algorithm 1: Constrained Naive and Weighted Learning Algorithm

---

Input: Dataset $\mathcal{D}^j, j \in [J]$, learning rate η_1, threshold ρ, max iteration I, max epochs MAX

Output: Optimized generator $h_{\theta_*}(y|x)$ that is an approximate minimizer of $R(h)$

initialization

1: **Repeat**
2: Sample a mini-batch of B real samples (x_i^j, y_i^j) from $\mathcal{D}^j$
3: Calculate $\hat{R}^{mini} = n\frac{1}{JB}\sum_{j=1}^J \lambda_j \sum_{i=1}^B \frac{h_{\theta_t}(y_i^j|x_i^j)}{h_j(y_i^j|x_i^j)}\delta_i^j$ and the gradient $g = \partial_\theta \hat{R}^{mini}$
4: Update $\theta_{t+1} = \theta_t - \eta_1 g$
5: Call Algorithm 2 to minimize $\sum_{j=1}^J n_j \lambda_j^2 \hat{D}_f(h||h_j; P(x))$ with threshold $\frac{\rho}{n^2}$ and max iteration I
6: **Until** epoch $> MAX$

---

choose T as the family of neural networks to obtain tight bound which benefits from the universal approximation theorem [15].

Therefore, in order to control the constraint, we need to solve a minimax problem. More specifically, we first maximize the lower bound for the establishment of eq.(4), then we minimize the f-divergence by choosing the optimal h. We represent T function as a discriminative network parameterized with w and express the policy $h(\mathcal{Y}|x)$ as a generator network with parameter θ. Define $F(\theta, w^j)$ as

$$F(\theta, w^j) = \mathbb{E}_{x,y \sim h_\theta} T_{w^j}(x, y) - \mathbb{E}_{x,y \sim h_j} f^*(T_{w^j}(x, y)). \quad (5)$$

To optimize eq.(5) on a finite training dataset, we can use mini-batch version to approximate the expectation. To approximate $\mathbb{E}_{x,y \sim h_j}[\cdot]$, we sample B instances without replacement from the training set. To approximate $\mathbb{E}_{x,y \sim h_\theta}[\cdot]$, we can sample B instances from the current generative policy h_θ.

4.2 Learning Algorithms

For constrained naive and weighted learning methods, we propose their algorithms in Algorithms 1–2. Correspondingly, for $\lambda_j = \frac{1}{n}, \forall j \in [J]$, it becomes the constrained naive learning algorithm, and if letting $\lambda_j = \lambda_j^*$, it turns to the learning algorithm of weighted IPS estimator.

In Algorithm 2, we leverage the Gumbel-softmax estimator in step 3. For structured output problem with discrete values, the gradients of samples obtained from the distribution $h(\mathcal{Y}|x)$ cannot backpropagate to all other parameters. The works [18] and [28] developed a continuous relaxation of discrete random variables in stochastic computational graphs, which can generate approximated and differentiable samples. The main ideas are as follows. They first use Gumbel-Max trick to represent a multinomial distribution, then it can be approximated by Gumbel-softmax distribution. Mathematically, given a categorical distribution with class probabilities $\pi_1, \pi_2, \ldots, \pi_k$, the Gumbel-softmax estimator generates an approximate one-hot sample y with

$$y_i = \frac{\exp\left((\log(\pi_i) + g_i)/\tau\right)}{\sum_{j=1}^k \exp\left((\log(\pi_j) + g_j)/\tau\right)}, i = 1, \ldots, k, \quad (6)$$

Algorithm 2: Variational Minimization of the Constraint

Input: Dataset $\mathcal{D}^j, j \in [J]$, threshold D, an initial generator $h_{\theta_0}(y|x)$, discriminator function $T_{w_0^j}(x, y), j \in [J]$, learning rates η_h, η_T, max iteration I

Output: Optimized generator $h_{\theta_*}(y|x)$ which satisfies the constraint

initialization

1: **Repeat**
2: Sample a mini-batch of B real samples (x_i^j, y_i^j) from $\mathcal{D}^j$ for each $j \in [J]$
3: Sample a mini-batch of B input x from $\mathcal{D}^j$ for each $j \in [J]$, and construct fake samples $(x_i^j, \hat{y}_i^j)$ by sampling from $h_{\theta_t}(y|x)$ with Gumbel-softmax sampling
4: Update $\theta_{t+1} = \theta_t - \eta_h \partial_\theta \left(\sum_{j=1}^J n_j \lambda_j^2 F(\theta_t, w_t^j)\right)$
5: Update
$w_{t+1}^j = w_t^j + \eta_T \partial_{w^j}\left(n_j \lambda_j^2 F(\theta_t, w_t^j)\right), j \in [J]$
6: **Until** $\sum_{j=1}^J n_j \lambda_j^2 \hat{D}_f(h||h_j; P(x)) \le D$ or iteration $> I$

Table 1: Statistics on Scene, Yeast, and TMC.

Dataset	p_F (#feature)	q_L (#label)	n_{train}	n_{test}
Scene	294	6	1211	1196
Yeast	103	14	1500	917
TMC	30438	22	21519	7077

where τ is the temperature and $g_1, \ldots, g_k$ are i.i.d samples drawn from $Gumbel(0, 1)$ distribution. The term $\hat{D}_f(h||h_j; P(x))$ denotes

$$\frac{1}{B}\sum_{(x_i^j, \hat{y}_i^j) \sim h_{\theta_t}} T_{w_t^j}(x, y) - \frac{1}{B}\sum_{(x_i^j, y_i^j) \sim h_j} f^*(T_{w_t^j}(x, y)),$$

i.e., the mini-batch version of $F(\theta_t, w_t^j)$.

5 EXPERIMENTS

In this section, we empirically evaluate the proposed algorithms, i.e., naive and weighted constrained algorithms on three benchmark datasets.

5.1 Experimental Settings

We first introduce the datasets and the experimental methodology.

5.1.1 Datasets and Methodology. In our experiments, we choose multi-label classification task due to the access of a rich feature space and an easily scalable label space. Three multi-label datasets are collected from the LibSVM repository [9] for the following experiments. Each dataset consists of feature $x \in \mathbb{R}^{p_F}$ and its corresponding supervised label $y^* \in \{0, 1\}^{q_L}$. The datasets have different feature dimension p_F, label dimension q_L, and sample number n. Statistics on the datasets are given in Table 1. For the dataset TMC, since it has sparse features with high dimension, we reduce the feature dimension to 1000 via truncated singular value decomposition (latent semantic analysis).

To control the experiments more effectively, we derive bandit data from these three full-information datasets. We follow the supervised $\mapsto$ bandit conversion method in [2]. For supervised data $\mathcal{D}^* = \{(x_i, y_i^*)\}_{i=1}^n$, we first train the conditional random fields (CRF) method [23] on a part of $\mathcal{D}^*$ to obtain logging policies. For the simplest setting, CRF actually performs logistic regression for each label independently. Following [45], we consider using the stochastic softmax rules

$$h_{logger}^w(y|x) = \frac{\exp(w^T \phi(x, y))}{\mathbb{Z}(x)}$$

as the hypothesis space. The $\phi(x, y)$ is the joint feature map of input x and output y, and $\mathbb{Z}(x) = \sum_{y' \in \mathcal{Y}} \exp(w^T \phi(x, y'))$ is the partition function. We also use a stochastic multiplier α in the map of $w \mapsto \alpha w$ to control the "stochasticity" of the logging policies, where larger α will induce a more deterministic variant of h_{logger}^w.

For simplicity and ease of interpretability, we consider two logging policies in the following experiments. We train the first logger h_1 on 20% of data with $\alpha = 0.05$. The second logger h_2 is trained on the same data with $\alpha = 2$. To create bandit feedback datasets $\mathcal{D} \equiv \mathcal{D}^1 \cup \mathcal{D}^2$, we take 4 passes through $\mathcal{D}^*$ and sample labels by simulating $y_i^j \sim h_j(x_i)$. We use the Hamming loss as the loss function $\delta(x_i^j, y_i^j)$, which is the number of incorrectly assigned labels between the sampled label y_i^j and the supervised label y_i^*. For the test dataset, we report the expected loss per test instance

$$EXP = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \mathbb{E}_{y_i \sim h(\mathcal{Y}|x)}[\delta(y_i^*, y_i)]$$

for the learned policy $h(\mathcal{Y}|x)$.

5.1.2 Baselines and Implementations. We compare our constrained naive and weighted algorithms, denoted by **Naive-Reg** and **Weighted-Reg**, with the following baselines:

- **WCRM**: We compare with the weighted counterfactual risk minimization (WCRM) principle [42]. They used the clipped version of estimator and applied the limited memory-BFGS (L-BFGS) algorithm [24] from scikit-learn [35] for optimization. We conduct it by following their experimental descriptions.
- **Naive/Weighted**: We utilize neural networks to present the policies for direct naive and weighted learning principles in Section 2.

For references, we also report the results from a supervised CRF (denoted as **CRF**), and the expected Hamming losses of h_1 (denoted as **Logger 1**) and h_2 (denoted as **Logger 2**). All CRF, Logger 1 and Logger 2 are actually built in supervised learning, where **CRF** is trained on the whole training dataset and the stochastic multiplier α is set as 1. We follow the implementations in [45] and the details can be found in their paper.

We keep aside 25% of the bandit dataset $\mathcal{D}$ as the validation set. The EXP is chosen according to the performance of the validation loss. For loggers and WCRM principle, we follow the experimental setup in [42, 45]. Direct learning principle and our algorithms are implemented with TensorFlow [3] in the experiments. We use Adam [22] to train the neural networks. The learning rate of the re-weighted loss is set as 0.0001. For the regularization part, we

[3] https://www.tensorflow.org/

Table 2: The comparisons of the expected Hamming loss on three datasets.

Method	Scene	Yeast	TMC
Logger 1	2.866	6.898	10.322
Logger 2	0.960	4.306	2.014
WCRM	1.088	3.908	4.232
Naive	1.056	4.001	4.452
Naive-Reg	1.037	3.551	**2.415**
Weighted	1.011	3.756	3.041
Weighted-Reg	**0.994**	**3.263**	2.748
CRF	0.942	4.133	1.612

set 0.0001 for the learned policy network. The learning rates for the discriminative networks are not fixed, one need to adjust it and we usually choose 0.00025. We set the batch size as 500 for Yeast and Scene datasets, and 4096 for TMC dataset. Some detailed configurations are put in the appendix. In addition, we leverage the Gumbel-softmax estimator for differential sampling, which was developed for variational methods [18, 28]. For the weight λ_j, we apply the self-normalized divergence estimator in [42], i.e.,

$$\tilde{\sigma}_\delta^2(h||h_j) = \frac{1}{n_j - 1} \sum_{i=1}^{n_j} \left(\frac{u_i^j(h)}{S^j(h)} - \bar{u}(h) \right)^2,$$

$$S^j(h) = \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{h(y_i^j|x_i^j)}{h_j(y_i^j|x_i^j)}, \quad \bar{u}(h) = \frac{1}{n} \sum_{j=1}^{J} \sum_{i=1}^{n_j} u_i^j(h).$$

5.2 Experimental Results

The expected Hamming loss EXP on three datasets are reported in Table 2. Lower loss is better. From the results we can see that, Naive-Reg beats the baseline of WCRM and Naive, while Weighted-Reg beats the baseline of WCRM and Weighted. Not surprised, Weighted-Reg achieves better performance than Naive-Reg in Scene and Yeast data set. The results indicate that our constrained learning method is effective due to the improvement of the generalization ability. Although Naive-Reg/Weighted-Reg can not surpass the better logger (i.e., Logger 2, trained in supervised learning), both of them perform much better than the baselines. We also notice that in some cases (e.g., on Yeast dataset), Weighted-Reg even gets competitive results, against supervised CRF method which is trained on the whole training dataset.

5.3 Experiments on Varying Replay Count

In this section, we aim to explore how the constrained naive and weighted algorithms work with varying replay count.

In the previous section, we both take 4 passes through $\mathcal{D}^*$ and sample labels for the two loggers. The stochastic multipliers are still set as 0.05 and 2 for logger h_1 and h_2, respectively. Keeping 4 passes for logger h_2, we vary the number of times that we replayed the training set (replay count) to collect labels from logger h_1 over $\{2^0, 2^1, 2^2, 2^3, 2^4\}$. The purpose of varying h_1 replay count rather than h_2, is to see the performance change of two proposed algorithms Naive-Reg and Weighted-Reg under different proportions of noisy data (recall that h_1 is more stochastic, i.e. more noisy).
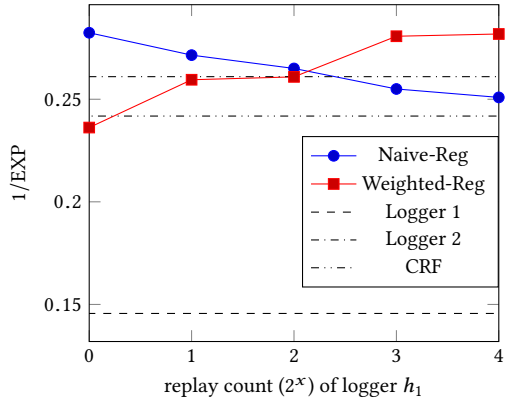
Figure 1: Generalization performance of Naive-Reg and Weighted-Reg as varying h_1 replay count on the Yeast dataset.

We report the results of Logger 1 and Logger 2, CRF, Naive-Reg and Weighted-Reg on Yeast dataset in Figure 1. The horizontal axis denotes the replay count number and the vertical axis is the inverse of the expected Hamming loss. The blue curve denotes the constrained naive method and the red curve denotes the constrained weighted method.

As shown in the figure, the performance of constrained naive algorithm gets worse with the increasing replay count. As mentioned above, smaller multiplier (i.e. h_1) would lead to worse logging policy. The data size of h_1 increases with the variation of the replay count from 2^0 to 2^4. Intuitively, this would deteriorate the performance of the learned policy. Whereas the cases are different for the constrained weighted algorithm, it performs better along with the increasing replay count. Since the constrained weighted learning method assigns different weight λ for each sample, it can take advantage of the growing training data size and get rid of the effects from noisy data.

5.4 Experiments on Varying Temperature

As mentioned in sec.4.2, we leverage the Gumbel-softmax trick for differential sampling. There is a temperature parameter τ in the Gumbel-softmax estimator, in this section, we study whether our learning methods are robust to this parameter.

We conduct the following analyses on two bandit datasets generated from Yeast. To eliminate the effects of parameters, the parameters for constrained naive learning method are set to be same with direct learning, except those parameters for the regularization. Furthermore, we keep the same parameters for all of the constrained naive learning methods except for the varying temperature τ. Specifically, naive method use one hidden layer with 10 hidden nodes for the learned policy network. The learning rate is set as 0.0001 and the batch size is set as 500. For the regularization, we also adopt one hidden layer but with 59 hidden nodes. The learning rates are set to be 0.0001 and 0.00025 for the step 4 and step 5 in algorithm 2, respectively.

For constrained weighted learning, we hold the same parameter set with that of direct weighted learning. All of the constrained
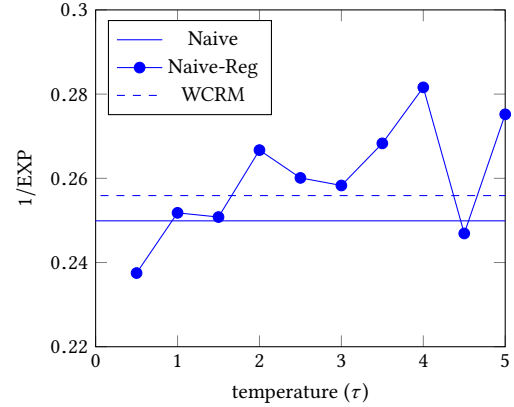

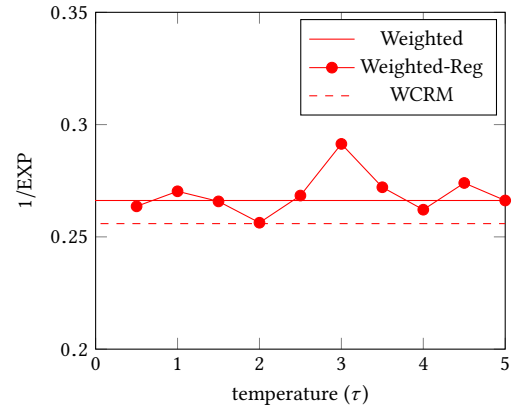Figure 2: Generalization performance of Naive-Reg as varying temperature parameter τ on the Yeast dataset.

Figure 3: Generalization performance of Weighted-Reg as varying temperature parameter τ on the Yeast dataset.

weighted learning methods use identical structure for the regularization except for the varying temperature τ. Specifically, weighted method has two hidden layers with 7 nodes in each hidden layer. The learning rate is also set as 0.0001 and the batch size is 500. For the regularization, we choose to use one hidden layer with 30 nodes, and also set 0.0001 and 0.00025 for step 4 and step 5 in algorithm 2.

We vary the temperature τ over $\{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$ and report the experimental results in Figures 2-3. The figures include WCRM, the direct learning version, and the constrained version. The vertical axis still denotes the inverse of the expected Hamming loss. As shown in the figures, we can see that the constrained versions are usually better than the direct learning versions. This shows that our new constrained learning methods are stable with the Gumbel-softmax temperature parameter.

6 OTHER VARIANTS OF ESTIMATORS

Besides the λ-weighted IPS estimator, in this section, we introduce the balanced IPS estimator [1], provide its generalization error analysis and the constrained learning algorithm correspondingly.

Balanced IPS Estimator

$$\hat{R}_{bal}(h) = \frac{1}{n} \sum_{j=1}^{J} \sum_{i=1}^{n_j} \frac{h(y_i^j | x_i^j)}{h_{avg}(y_i^j | x_i^j)} \delta_i^j,$$

where $h_{avg}(y|x) \equiv \frac{\sum_{j=1}^{J} n_j h_j(y|x)}{n}$. This is also an unbiased estimator with a smaller variance than naive IPS estimator.

THEOREM 6.1. *Let $R(h)$ be the risk of a new policy h on the loss function δ, and $\hat{R}_{bal}(h)$ be the balanced empirical risk. Assume that the divergence is bounded by M_{avg}, i.e., $d_2(h||h_{avg}) \leq d_{\infty}(h||h_{avg}) = M_{avg}$. Then, for any $\eta > 0$, with probability at least $1 - \eta$, the following bound holds:*

$$R(h) \leq \hat{R}_{bal}(h) + \frac{2LM_{avg} \log \frac{1}{\eta}}{3n} + L\sqrt{\frac{2d_2(h||h_{avg}; P(x)) \log \frac{1}{\eta}}{n}}.$$

If the last term is reformulated as $L\sqrt{\frac{2\sum_{j=1}^{J} n_j d_2(h||h_{avg}; P(x)) \log \frac{1}{\eta}}{n^2}}$, it becomes similar to that of the naive IPS estimator.

We propose to use the following regularized objective, i.e.,

$$\min_{h} \hat{R}_{bal}(h) + \beta \sqrt{\frac{d_2(h||h_{avg}; P(x))}{n}}. \quad (7)$$

Similarly, we minimize the following constrained optimization problem instead, i.e.,

$$\min_{h} \frac{1}{n} \sum_{j=1}^{J} \sum_{i=1}^{n_j} \frac{h(y_i^j | x_i^j)}{h_{avg}(y_i^j | x_i^j)} \delta_i^j \quad (8)$$

$$s.t. d_2(h||h_{avg}; P(x)) \leq \frac{\rho}{n}.$$

We can give the corresponding algorithm for the constrained balanced learning method. Compared with Algorithm 1, we should replace step 3, i.e., the computation of minibatch gradient, with $\hat{R}_{bal}^{mini} = \frac{1}{JB} \sum_{j=1}^{J} \sum_{b=1}^{B} \frac{h_{\theta_t}(y_i^j | x_i^j)}{h_{avg}(y_i^j | x_i^j)} \delta(x_i^j, y_i^j)$. The step 2 in Algorithm 2 should be modified by constructing samples from the policy h_{avg}. Since the quantities of $h_j(y_i^j | x_i^j)$ are assumed to be available for all possible (x, y) pairs, we are able to calculate $h_{avg}(y_i^j | x_i^j)$ and sample from it. For the constrained balanced learning algorithm, we use $\hat{D}_f(h||h_{avg}; P(x))$ to denote the mini-batch version of $F(\theta_t, w_t^{avg})$, i.e.,

$$\hat{D}_f(h||h_{avg}; P(x)) =$$

$$\frac{1}{JB} \sum_{(x_i^j, \hat{y}_i^j) \sim h_{\theta_t}} T_{w_t^{avg}}(x, y) - \frac{1}{JB} \sum_{(x_i^j, y_i^{avg}) \sim h_{avg}} f^*(T_{w_t^{avg}}(x, y)).$$

Please refer to the appendix for the complete algorithm.

7 CONCLUSION

Preforming off-policy learning is becoming more important than online policy learning in real-world applications. Most of previous works are focused on the off-policy learning with one single historical policy. In this paper, we studied the off-policy learning from multiple historical policies, which is important and realistic. The deep neural nets are adopted as the learned policy and the discriminative networks of logging policies for learning. The generalization error analysis for the empirical risk minimization problem is provided. Based on the analysis, we proposed to use the generalization error bound as the new risk function, which can be alternatively transformed into a constrained optimization problem. Learning algorithm for the optimization problem is designed, through a minimax setting, to solve the constraint of the optimization problem. In experiments, we test the new methods on three benchmark datasets. Compared with direct learning principle and the WCRM principle, the performances of proposed algorithms outperform the state-of-the-art baselines. In the future, we will try to find other measures to control the differences between the logging policies and the learned policy.

REFERENCES

[1] Aman Agarwal, Soumya Basu, Tobias Schnabel, and Thorsten Joachims. 2017. Effective Evaluation Using Logged Bandit Feedback from Multiple Loggers. In *SIGKDD'17*. ACM, 687–696.
[2] Alekh Agarwal, Daniel J. Hsu, Satyen Kale, John Langford, Lihong Li, and Robert E. Schapire. 2014. Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits. In *ICML'14*. 1638–1646.
[3] Susan Athey and Stefan Wager. 2017. Efficient Policy Learning. *CoRR* abs/1702.02896 (2017).
[4] George Bennett. 1962. Probability inequalities for the sum of independent random variables. *J. Amer. Statist. Assoc.* 57, 297 (1962), 33–45.
[5] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. 2018. Data-driven robust optimization. *Math. Program.* 167, 2 (2018), 235–292.
[6] Alina Beygelzimer and John Langford. 2009. The offset tree for learning with partial labels. In *SIGKDD'09*. ACM, 129–138.
[7] Léon Bottou, Jonas Peters, Joaquin Quiñonero Candela, Denis Xavier Charles, Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Y. Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: the example of computational advertising. *Journal of Machine Learning Research* 14, 1 (2013), 3207–3260.
[8] Claes M Cassel, Carl E Särndal, and Jan H Wretman. 1976. Some results on generalized difference estimation and generalized regression estimation for finite populations. *Biometrika* 63 (1976), 615–620.
[9] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM TIST* 2, 3 (2011), 27:1–27:27.
[10] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. 2010. Learning Bounds for Importance Weighting. In *NIPS'10*. 442–450.
[11] Imre Csiszár and Paul C. Shields. 2004. Information Theory and Statistics: A Tutorial. *Foundations and Trends in Communications and Information Theory* 1, 4 (2004).
[12] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly Robust Policy Evaluation and Learning. In *ICML'11*. 1097–1104.
[13] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. 2018. More Robust Doubly Robust Off-policy Evaluation. In *ICML'18*. 1446–1455.
[14] LemarÁĺchal Claude Hiriart-Urruty, Jean-Baptiste. 2012. *Fundamentals of convex analysis.* Springer.
[15] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366.
[16] D. G. Horvitz and D. J. Thompson. 1952. A Generalization of Sampling Without Replacement from a Finite Universe. *J. Amer. Statist. Assoc.* 47, 260 (1952), 663–685.
[17] Edward L Ionides. 2008. Truncated Importance Sampling. *Journal of Computational and Graphical Statistics* 17, 2 (2008), 295–311.
[18] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. In *ICLR'16*.
[19] Nan Jiang and Lihong Li. 2016. Doubly Robust Off-policy Value Evaluation for Reinforcement Learning. In *ICML'16*. 652–661.
[20] Thorsten Joachims and Adith Swaminathan. 2016. Counterfactual Evaluation and Learning for Search, Recommendation and Ad Placement. In *SIGIR'16*. ACM, 1199–1201.
[21] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2018. Unbiased Learning-to-Rank with Biased Feedback. In *IJCAI'18*. 5284–5288.
[22] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. (2015).

[23] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML'01*. 282–289.

[24] Adrian S. Lewis and Michael L. Overton. 2013. Nonsmooth optimization via quasi-Newton methods. *Math. Program.* 141, 1-2 (2013), 135–163.

[25] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. 2014. Counterfactual Estimation and Optimization of Click Metrics for Search Engines. *CoRR* abs/1403.1891 (2014).

[26] Lihong Li, Wei Chu, John Langford, Taesup Moon, and Xuanhui Wang. 2011. An Unbiased Offline Evaluation of Contextual Bandit Algorithms with Generalized Linear Models. In *Proceedings of the 2011 International Conference on On-line Trading of Exploration and Exploitation 2 - Volume 26 (OTEAE'11)*. JMLR.org, 19–36.

[27] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM'11*. ACM, 297–306.

[28] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *CoRR* abs/1611.00712 (2016).

[29] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. 2016. Safe and Efficient Off-Policy Reinforcement Learning. In *NIPS'16*. 1046–1054.

[30] Hongseok Namkoong and John C. Duchi. 2017. Variance-based Regularization with Convex Objectives. In *NIPS'17*. 2975–2984.

[31] Yusuke Narita, Shota Yasui, and Kohei Yata. 2018. Efficient Counterfactual Learning from Bandit Feedback. *CoRR* abs/1809.03084 (2018).

[32] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. 2010. Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization. *IEEE Trans. Information Theory* 56, 11 (2010), 5847–5861.

[33] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *NIPS'16*. 271–279.

[34] Art B. Owen. 2013. *Monte Carlo theory, methods and examples*.

[35] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[36] Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility Traces for Off-Policy Policy Evaluation. In *ICML'00*. 759–766.

[37] Alfréd Rényi. 1961. On Measures of Entropy and Information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. University of California Press, 547–561.

[38] PAUL R. ROSENBAUM and DONALD B. RUBIN. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70, 1 (1983), 41–55.

[39] Uri Shalit, Fredrik D. Johansson, and David Sontag. 2017. Estimating individual treatment effect: generalization bounds and algorithms. In *ICML'17*. 3076–3085.

[40] Pannagadatta K. Shivaswamy and Thorsten Joachims. 2012. Multi-armed Bandit Problems with History. In *AISTATS'12*. 1046–1054.

[41] Alexander L. Strehl, John Langford, Lihong Li, and Sham Kakade. 2010. Learning from Logged Implicit Exploration Data. In *NIPS'10*. 2217–2225.

[42] Yi Su, A. Agarwal, and T. Joachims. 2018. Learning from Logged Bandit Feedback of Multiple Loggers. In *ICML Workshop on Machine Learning for Causal Inference, Counterfactual Prediction, and Autonomous Action (CausalML)*.

[43] Yi Su, Lequn Wang, Michele Santacatterina, and Thorsten Joachims. 2019. CAB: Continuous Adaptive Blending for Policy Evaluation and Learning. In *ICML'19*.

[44] Richard S. Sutton and Andrew G. Barto. 1998. Reinforcement Learning: An Introduction. *IEEE Trans. Neural Networks* 9, 5 (1998), 1054–1054.

[45] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual Risk Minimization: Learning from Logged Bandit Feedback. In *ICML'15*. 814–823.

[46] Adith Swaminathan and Thorsten Joachims. 2015. The Self-Normalized Estimator for Counterfactual Learning. In *NIPS'15*. 3231–3239.

[47] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudík, John Langford, Damien Jose, and Imed Zitouni. 2017. Off-policy evaluation for slate recommendation. In *NIPS'17*. 3635–3645.

[48] Philip S. Thomas and Emma Brunskill. 2016. Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning. In *ICML'16*. 2139–2148.

[49] 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'18*. ACM, 610–618.

[50] Hang Wu and May D. Wang. 2018. Variance Regularized Counterfactual Risk Minimization via Variational Divergence Minimization. In *ICML'18*. 5349–5358.

[51] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Deep Reinforcement Learning for Search, Recommendation, and Online Advertising: A Survey. *CoRR* abs/1812.07127v4 (2018).

[52] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *RecSys'18*. ACM, 95–103.

[53] Xiangyu Zhao, Long Xia, Yihong Zhao, Dawei Yin, and Jiliang Tang. 2019. Model-Based Reinforcement Learning for Whole-Chain Recommendations. *CoRR* abs/1902.03987 (2019).

[54] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *SIGKDD'18*. ACM, 1040–1048.

[55] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2018. Deep Reinforcement Learning for List-wise Recommendations. *CoRR* abs/1801.00209 (2018).

[56] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems. In *SIGKDD'19*. ACM.

APPENDIX

A   PROOFS OF THEOREM 6.1

Theorem A.1. *Let $R(h)$ be the risk of a new policy h on the loss function δ, and $\hat{R}_{bal}(h)$ be the balanced empirical risk. Assume that the divergence is bounded by M_{avg}, i.e., $d_2(h||h_{avg}) \leq d_\infty(h||h_{avg}) = M_{avg}$. Then, for any $\eta > 0$, with probability at least $1 - \eta$, the following bound holds:*

$$R(h) \leq \hat{R}_{bal}(h) + \frac{2LM_{avg}\log\frac{1}{\eta}}{3n} + L\sqrt{\frac{2d_2(h||h_{avg};P(x))\log\frac{1}{\eta}}{n}}.$$

Proof. By definition, we have

$$R(h) - \hat{R}_{bal}(h) = \frac{1}{n}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\left[R(h) - \frac{h(y_i^j|x_i^j)}{h_{avg}(y_i^j|x_i^j)}\delta(x_i^j, y_i^j)\right].$$

Denote $X_i^j = R(h) - \frac{h(y_i^j|x_i^j)}{h_{avg}(y_i^j|x_i^j)}\delta(x_i^j, y_i^j)$ and $X = R(h) - \frac{h(y|x)}{h_{avg}(y|x)}\delta(x, y)$. Taking expectation, we have $\mathbb{E}_{x \sim P(X), y \sim h_{avg}(\mathcal{Y}|x)}X = 0$. We can also derive that

$$|X| \leq \left|\frac{h(y|x)}{h_{avg}(y|x)}\delta(x,y)\right| \leq M_{avg}L.$$

If $\frac{h(y|x)}{h_{avg}(y|x)} \geq M_{avg}$, then $d_2(h||h_{avg}) \equiv \int_y \frac{h(y|x)}{h_{avg}(y|x)}h(y|x)dy \geq \int_y M_{avg}h(y|x)dy = M_{avg}$. This contradicts with the assumption.

In addition, by applying lemma 3.1, we have

$$\mathbb{E}_{x \sim P(X), y \sim h_{avg}(\mathcal{Y}|x)}\left[\left(\frac{h(y|x)}{h_{avg}(y|x)}\delta(x,y)\right)^2\right] \leq L^2d_2(h(y|x)||h_{avg}(y|x); P(x)).$$

Thus, we have the following bound for the second moment of X,

$$\mathbb{E}_{x \sim P(X), y \sim h_{avg}(\mathcal{Y}|x)}X^2 \leq L^2d_2(h(y|x)||h_{avg}(y|x); P(x)).$$

Applying Bernstein's inequality [4], we have

$$\mathbb{P}\left(\frac{1}{n}\sum_{j=1}^{J}\sum_{i=1}^{n_j}X_i^j > \epsilon\right)$$

$$\leq \exp\left(-\frac{\frac{1}{2}n^2\epsilon^2}{\sum_{j=1}^{J}\sum_{i=1}^{n_j}\mathbb{E}_{x \sim P(X), y \sim h_{avg}(\mathcal{Y}|x)}(X_i^j)^2 + \frac{1}{3}LM_{avg}n\epsilon}\right)$$

$$\leq \exp\left(-\frac{\frac{1}{2}n^2\epsilon^2}{\sum_{j=1}^{J}n_jL^2d_2(h(y|x)||h_{avg}(y|x); P(x)) + \frac{1}{3}LM_{avg}n\epsilon}\right).$$

Let the right hand be equal to η, then we can obtain that

$$\log\frac{1}{\eta} = \frac{\frac{1}{2}n^2\epsilon^2}{\sum_{j=1}^{J}n_jL^2d_2(h(y|x)||h_{avg}(y|x); P(x)) + \frac{1}{3}LM_{avg}n\epsilon}.$$

This is an quadratic function of ϵ and we solve that

$$\epsilon \leq \frac{2LM_{avg}\log\frac{1}{\eta}}{3n} + L\sqrt{\frac{2\sum_{j=1}^{J}n_jd_2(h(y|x)||h_{avg}(y|x); P(x))\log\frac{1}{\eta}}{n^2}}.$$

Therefore, the following inequality

$$R(h) \leq \hat{R}_{bal}(h) +$$

$$\frac{2LM_{avg}\log\frac{1}{\eta}}{3n} + L\sqrt{\frac{2\sum_{j=1}^{J}n_jd_2(h(y|x)||h_{avg}(y|x); P(x))\log\frac{1}{\eta}}{n^2}}$$

holds with probability at least $1 - \eta$.   □

B   CONSTRAINED BALANCED LEARNING ALGORITHM

In this section, we will give the corresponding algorithm for the balanced IPS estimator.

Algorithm 3: Constrained Balanced Learning Algorithm

Input: Dataset $\mathcal{D}^j, j \in [J]$, threshold D, an initial generator $h_{\theta_0}(y|x)$, discriminator function $T_{w_0}^{avg}(x, y)$, max iteration I

Output: Optimized generator $h_{\theta_*}(y|x)$ that is an approximate minimizer of $R(w)$

1: **Repeat**
2: Sample each mini-batch of B real samples (x_i^j, y_i^j) from $\mathcal{D}^j, j \in [J]$
3: Calculate $\hat{R}_{bal}^{mini} = \frac{1}{JB}\sum_{j=1}^{J}\sum_{i=1}^{B}\frac{h_{\theta_t}(y_i^j|x_i^j)}{h_{avg}(y_i^j|x_i^j)}\delta(x_i^j, y_i^j)$ and the gradient $g = \partial_\theta \hat{R}_{bal}^{mini}$
4: Update $\theta_{t+1} = \theta_t - \eta_2 g$
5: Call Algorithm 4 to minimize the divergence $\hat{D}_f(h||h_{avg}; P(x))$ with threshold $\frac{\rho}{n}$, and max iteration I
6: **Until** epoch > MAX

Algorithm 4: Variational Minimization of the Constraint

Input: Dataset $\mathcal{D}^j, j \in [J]$, threshold D, an initial generator $h_{\theta_0}(y|x)$, discriminator function $T_{w_0}^{avg}(x, y)$, learning rates η_h, η_T, max iteration I

Output: Optimized generator $h_{\theta_*}(y|x)$ that has minimum divergence to h_{avg}

1: **Repeat**
2: Sample a mini-batch of B real samples (x_i^j, y_i^j) from $\mathcal{D}^j$ for each $j \in [J]$, and construct JB samples (x_i^j, y_i^{avg}) by sampling from $h_{avg}(y|x)$
3: Sample a mini-batch B of input x_i^j from $\mathcal{D}^j$ for each $j \in [J]$, and construct fake samples $(x_i^j, \hat{y}_i^j)$ by sampling from $h_{\theta_t}(y|x)$ with Gumbel-softmax sampling
4: Update $\theta_{t+1} = \theta_t - \eta_h\partial_\theta F(\theta_t, w_t^{avg})$
5: Update $w_{t+1}^{avg} = w_t^{avg} + \eta_T\partial_{w^{avg}}F(\theta_t, w_t^{avg})$
6: **Until** $\hat{D}_f(h||h_{avg}; P(x)) \leq D$ or iteration > I

C   NETWORK CONFIGURATIONS

For the discriminative network T and the generative network h, we use structures like below:

Discrimitive NN: *Linear→ BatchNorm → ReLU → Linear → BatchNorm → ReLU → Linear*

Generative NN: *Linear→ BatchNorm → ReLU → Linear → BatchNorm → ReLU → Linear → Sigmoid*