Learning from Logged Implicit Exploration Data

Alexander L. Strehl [*]
Facebook Inc.
1601 S California Ave
Palo Alto, CA 94304
astrehl@facebook.com

John Langford
Yahoo! Research
111 West 40th Street, 9th Floor
New York, NY, USA 10018
jl@yahoo-inc.com

Lihong Li
Yahoo! Research
4401 Great America Parkway
Santa Clara, CA, USA 95054
lihong@yahoo-inc.com

Sham M. Kakade
Department of Statistics
University of Pennsylvania
Philadelphia, PA, 19104
skakade@wharton.upenn.edu

Abstract

We provide a sound and consistent foundation for the use of *nonrandom* exploration data in "contextual bandit" or "partially labeled" settings where only the value of a chosen action is learned. The primary challenge in a variety of settings is that the exploration policy, in which "offline" data is logged, is not explicitly known. Prior solutions here require either control of the actions during the learning process, recorded random exploration, or actions chosen obliviously in a repeated manner. The techniques reported here lift these restrictions, allowing the learning of a policy for choosing actions given features from historical data where no randomization occurred or was logged. We empirically verify our solution on two reasonably sized sets of real-world data obtained from Yahoo!.

1 Introduction

Consider the advertisement display problem, where a search engine company chooses an ad to display which is intended to interest the user. Revenue is typically provided to the search engine from the advertiser only when the user clicks on the displayed ad. This problem is of intrinsic economic interest, resulting in a substantial fraction of income for several well-known companies such as Google, Yahoo!, and Facebook.

Before discussing the proposed approach, we formalize the problem and then explain why more conventional approaches can fail.

The warm-start problem for contextual exploration: Let $\mathcal{X}$ be an arbitrary input space, and $\mathcal{A} = \{1, \ldots, k\}$ be a set of actions. An instance of the *contextual bandit problem* is specified by a distribution D over tuples $(x, \vec{r})$ where $x \in \mathcal{X}$ is an input and $\vec{r} \in [0,1]^k$ is a vector of rewards [6]. Events occur on a round-by-round basis where on each round t:

1. The world draws $(x, \vec{r}) \sim D$ and announces x.
2. The algorithm chooses an action $a \in \mathcal{A}$, possibly as a function of x and historical information.
3. The world announces the reward r_a of action a, but not $r_{a'}$ for $a' \neq a$.

[*] Part of this work was done while A. Strehl was at Yahoo! Research.

It is critical to understand that this is not a standard supervised-learning problem, because the reward of other actions $a' \neq a$ is not revealed.

The standard goal in this setting is to maximize the sum of rewards r_a over the rounds of interaction. In order to do this well, it is essential to use previously recorded events to form a good policy on the first round of interaction. Thus, this is a “warm start” problem. Formally, given a dataset of the form $S = (x, a, r_a)^*$ generated by the interaction of an uncontrolled logging policy, we want to construct a policy h maximizing (either exactly or approximately)

$$V^h := E_{(x,\vec{r})\sim D}[r_{h(x)}].$$

Approaches that fail: There are several approaches that may appear to solve this problem, but turn out to be inadequate:

1. *Supervised learning*. We could learn a regressor $s : X \times A \to [0, 1]$ which is trained to predict the reward, on observed events conditioned on the action a and other information x. From this regressor, a policy is derived according to $h(x) = \text{argmax}_{a\in A}\, s(x, a)$. A flaw of this approach is that the argmax may extend over a set of choices not included in the training data, and hence may not generalize at all (or only poorly). This can be verified by considering some extreme cases. Suppose that there are two actions a and b with action a occurring 10^6 times and action b occuring 10^2 times. Since action b occurs only a 10^{-4} fraction of the time, a learning algorithm forced to trade off between predicting the expected value of r_a and r_b overwhelmingly prefers to estimate r_a well at the expense of accurate estimation for r_b. And yet, in application, action b may be chosen by the argmax. This problem is only worse when action b occurs zero times, as might commonly occur in exploration situations.
2. *Bandit approaches*. In the standard setting these approaches suffer from the curse of dimensionality, because they must be applied conditioned on X. In particular, applying them requires data linear in $X \times A$, which is extraordinarily wasteful. In essence, this is a failure to take advantage of generalization.
3. *Contextual Bandits*. Existing approaches to contextual bandits such as EXP4 [1] or Epoch Greedy [6], require either interaction to gather data or require knowledge of the probability the logging policy chose the action a. In our case the probability is unknown, and it may in fact always be 1.
4. *Exploration Scavenging*. It is possible to recover exploration information from action visitation frequency when a logging policy chooses actions independent of the input x (but possibly dependent on history) [5]. This doesn’t fit our setting, where the logging policy is surely dependent on the input.
5. *Propensity Scores*, naively. When conducting a survey, a question about income might be included, and then the proportion of responders at various income levels can be compared to census data to estimate a probability conditioned on income that someone chooses to partake in the survey. Given this estimated probability, results can be importance-weighted to estimate average survey outcomes on the entire population [2]. This approach is problematic here, because the policy making decisions when logging the data may be deterministic rather than probabilistic. In other words, accurately predicting the probability of the logging policy choosing an ad implies always predicting 0 or 1 which is not useful for our purposes. Although the straightforward use of propensity scores does not work, the approach we take can be thought of as as a more clever use of a propensity score, as discussed below. Lambert and Pregibon [4] provide a good explanation of propensity scoring in an Internet advertising setting.

Our Approach: The approach proposed in the paper naturally breaks down into three steps.

1. For each event (x, a, r_a), estimate the probability $\hat{\pi}(a|x)$ that the logging policy chooses action a using regression. Here, the “probability” is over *time*—we imagine taking a uniform random draw from the collection of (possibly deterministic) policies used at different points in time.
2. For each event (x, a, r_a), create a synthetic controlled contextual bandit event according to $(x, a, r_a, 1/\max\{\hat{\pi}(a|x), \tau\})$ where $\tau > 0$ is some parameter. The quantity, $1/\max\{\hat{\pi}(a|x), \tau\}$, is an *importance weight* that specifies how important the current event is for training. As will be clear, the parameter τ is critical for numeric stability.

3. Apply an offline contextual bandit algorithm to the set of synthetic contextual bandit events. In our second set of experimental results (Section 4.2) a variant of the argmax regressor is used with two critical modifications: (a) We limit the scope of the argmax to those actions with positive probability; (b) We importance weight events so that the training process emphasizes good estimation for each action equally. It should be emphasized that the theoretical analysis in this paper applies to *any* algorithm for learning on contextual bandit events—we chose this one because it is a simple modification on existing (but fundamentally broken) approaches.

The above approach is most similar to the Propensity Score approach mentioned above. Relative to it, we use a different definition of probability which is not necessarily 0 or 1 when the logging policy is completely deterministic.

Three critical questions arise when considering this approach.

1. What does $\hat{\pi}(a|x)$ mean, given that the logging policy may be deterministically choosing an action (ad) a given features x? The essential observation is that a policy which deterministically chooses action a on day 1 and then deterministically chooses action b on day 2 can be treated as randomizing between actions a and b with probability 0.5 when the number of events is the same each day, and the events are IID. Thus $\hat{\pi}(a|x)$ is an estimate of the expected frequency with which action a would be displayed given features x over the timespan of the logged events. In section 3 we show that this approach is sound in the sense that in expectation it provides an unbiased estimate of the value of new policy.
2. How do the inevitable errors in $\hat{\pi}(a|x)$ influence the process? It turns out they have an effect which is dependent on τ. For very small values of τ, the estimates of $\hat{\pi}(a|x)$ must be extremely accurate to yield good performance while for larger values of τ less accuracy is required. In Section 3.1, we prove this robustness property.
3. What influence does the parameter τ have on the final result? While creating a bias in the estimation process, it turns out that the form of this bias is mild and relatively reasonable—actions which are displayed with low frequency conditioned on x effectively have an underestimated value. This is exactly as expected for the limit where actions have *no* frequency. In section 3.1 we prove this.

We close with a generalization from policy evaluation to policy selection with a sample complexity bound in section 3.2 and then experimental results in section 4 using real data.

2 Formal Problem Setup and Assumptions

Let $\pi_1, ..., \pi_T$ be T policies, where, for each t, π_t is a function mapping an input from X to a (possibly deterministic) distribution over A. The learning algorithm is given a dataset of T samples, each of the form $(x, a, r_a) \in X \times A \times [0, 1]$, where (x, r) is drawn from D as described in Section 1, and the action $a \sim \pi_t(x)$ is chosen according to the tth policy. We denote this random process by $(x, a, r_a) \sim (D, \pi_t(\cdot|x))$. Similarly, interaction with the T policies results in a sequence S of T samples, which we denote $S \sim (D, \pi_i(\cdot|x))_{i=1}^T$. The learner is not given prior knowledge of the π_t.

Offline policy estimator: Given a dataset of the form

$$S = \{(x_t, a_t, r_{t,a_t})\}_{t=1}^T, \tag{1}$$

where $\forall t, x_t \in X, a_t \in A, r_{t,a_t} \in [0, 1]$, we form a predictor $\hat{\pi} : X \times A \to [0, 1]$ and then use it with a threshold $\tau \in [0, 1]$ to form an offline estimator for the value of a policy h.

Formally, given a new policy $h : X \to A$ and a dataset S, define the estimator:

$$\hat{V}_{\hat{\pi}}^h(S) = \frac{1}{|S|} \sum_{(x,a,r)\in S} \frac{r_a I(h(x) = a)}{\max\{\hat{\pi}(a|x), \tau\}}, \tag{2}$$

where $I(\cdot)$ denotes the indicator function. The shorthand $\hat{V}_{\hat{\pi}}^h$ will be used if there is no ambiguity. The purpose of τ is to upper-bound the individual terms in the sum and is similar to previous methods like robust importance sampling [10].

The purpose of τ is to upper-bound the individual terms in the sum and is similar to previous methods like robust importance sampling [10].

3 Theoretical Results

We now present our algorithm and main theoretical results. The main idea is twofold: first, we have a policy estimation step, where we estimate the (unknown) logging policy (Subsection 3.1); second, we have a policy optimization step, where we utilize our estimated logging policy (Subsection 3.2). Our main result, Theorem 3.2, provides a generalization bound—addressing the issue of how both the estimation and optimization error contribute to the total error.

The logging policy π_t may be deterministic, implying that conventional approaches relying on randomization in the logging policy are not applicable. We show next that this is ok when the world is IID and the policy varies over its actions. We effectively substitute the standard approach of randomization in the algorithm for randomization in the world.

A basic claim is that the estimator is equivalent, in expectation, to a stochastic policy defined by:

$$\pi(a|x) = \mathbb{E}_{t \sim \mathrm{UNIF}(1,\ldots,T)}[\pi_t(a|x)], \tag{3}$$

where $\mathrm{UNIF}(\cdots)$ denotes the uniform distribution. The stochastic policy π chooses an action uniformly at random over the T policies π_t. Our first result is that the expected value of our estimator is the same when the world chooses actions according to either π or to the sequence of policies π_t. Although this result and its proof are straightforward, it forms the basis for the rest of the results in our paper. Note that the policies π_t may be arbitrary but we have assumed that they do not depend on the data used for evaluation. This assumption is only necessary for the proofs and can often be relaxed in practice, as we show in Section 4.1.

Theorem 3.1. *For any contextual bandit problem D with identical draws over T rounds, for any sequence of possibly stochastic policies $\pi_t(a|x)$ with π derived as above, and for any predictor $\hat{\pi}$,*

$$E_{S \sim (D, \pi_i(\cdot|x))_{i=1}^{T}} \hat{V}_{\hat{\pi}}^{h}(S) = E_{(x,\vec{r}) \sim D, a \sim \pi(\cdot|x)} \frac{r_a I(h(x)=a)}{\max\{\hat{\pi}(a|x), \tau\}} \tag{4}$$

This theorem relates the expected value of our estimator when T policies are used to the much simpler and more standard setting where a single fixed stochastic policy is used.

3.1 Policy Estimation

In this section we show that for a suitable choice of τ and $\hat{\pi}$ our estimator is sufficiently accurate for evaluating new policies h. We aggressively use the simplification of the previous section, which shows that we can think of the data as generated by a fixed stochastic policy π, i.e. $\pi_t = \pi$ for all t.

For a given estimate $\hat{\pi}$ of π define the "regret" to be a function $\mathrm{reg}\colon X \to [0,1]$ by

$$\mathrm{reg}(x) = \max_{a \in \mathcal{A}} \left[(\pi(a|x) - \hat{\pi}(a|x))^2 \right]. \tag{5}$$

We do not use ℓ_1 or ℓ_∞ loss above because they are harder to minimize than ℓ_2 loss. Our next result is that the new estimator is consistent. In the following theorem statement, $I(\cdot)$ denotes the indicator function, $\pi(a|x)$ the probability that the logging policy chooses action a on input x, and $\hat{V}_{\hat{\pi}}^{h}$ our estimator as defined by Equation 2 based on parameter τ.

Lemma 3.1. *Let $\hat{\pi}$ be any function from X to distributions over actions A. Let $h : X \to A$ be any deterministic policy. Let $V^h(x) = \mathbb{E}_{r \sim D(\cdot|x)}[r_{h(x)}]$ denote the expected value of executing policy h on input x. We have that*

$$\mathbb{E}_x \left[I(\pi(h(x)|x) \geq \tau) \cdot \left(V^h(x) - \frac{\sqrt{\mathrm{reg}(x)}}{\tau} \right) \right] \leq \mathbb{E}[\hat{V}_{\hat{\pi}}^{h}] \;\leq\; V^h + \mathbb{E}_x \left[I(\pi(h(x)|x) \geq \tau) \cdot \frac{\sqrt{\mathrm{reg}(x)}}{\tau} \right].$$

In the above, the expectation $\mathbb{E}[\hat{V}_{\hat{\pi}}^{h}]$ is taken over all sequences of T tuples (x, a, r) where $(x, r) \sim D$ and $a \sim \pi(\cdot|x)$.[1]

This lemma bounds the bias in our estimate of $V^h(x)$. There are two sources of bias—one from the error of $\hat{\pi}(a|x)$ in estimating $\pi(a|x)$, and the other from threshold τ. For the first source, it's crucial that we analyze the result in terms of the squared loss rather than (say) ℓ_∞ loss, as reasonable sample complexity bounds on the regret of squared loss estimates are achievable.[2]

[1] Note that varying T does not change the expectation of our estimator, so T has no effect in the theorem.

[2] Extending our results to log loss would be interesting future work, but is made difficult by the fact that log loss is unbounded.

Lemma 3.1 shows that the expected value of our estimate $\hat{V}^h_\pi$ of a policy h is an approximation to a lower bound of the true value of the policy h where the approximation is due to errors in the estimate $\hat{\pi}$ and the lower bound is due to the threshold τ. When $\hat{\pi} = \pi$, then the statement of Lemma 3.1 simplifies to

$$\mathbb{E}_x \left[I(\pi(h(x)|x) \geq \tau) \cdot V^h(x) \right] \;\leq\; \mathbb{E}[\hat{V}^h_{\hat{\pi}}] \;\leq\; V^h.$$

Thus, with a perfect predictor of π, the expected value of the estimator $\hat{V}^h_{\hat{\pi}}$ is a guaranteed lower bound on the true value of policy h. However, as the left-hand-side of this statement suggests, it may be a very loose bound, especially if the action chosen by h often has a small probability of being chosen by π.

The dependence on $1/\tau$ in Lemma 3.1 is somewhat unsettling, but unavoidable. Consider an instance of the bandit problem with a single input x and two actions a_1, a_2. Suppose that $\pi(a_1|x) = \tau + \epsilon$ for some positive ϵ and $h(x) = a_1$ is the policy we are evaluating. Suppose further that the rewards are always 1 and that $\hat{\pi}(a_1|x) = \tau$. Then, the estimator satisfies $E[\hat{V}^h_{\hat{\pi}}] = \pi(a_1|x)/\hat{\pi}(a_1|x) = (\tau + \epsilon)/\tau$. Thus, the expected error in the estimate is $E[\hat{V}^h_{\hat{\pi}}] - V^h = |(\tau + \epsilon)/\tau - 1| = \epsilon/\tau$, while the regret of $\hat{\pi}$ is $(\pi(a_1|x) - \hat{\pi}(a_1|x))^2 = \epsilon^2$.

3.2 Policy Optimization

The previous section proves that we can effectively evaluate a policy h by observing a stochastic policy π, as long as the actions chosen by h have adequate support under π, specifically $\pi(h(x)|x) \geq \tau$ for all inputs x. However, we are often interested in choosing the best policy h from a set of policies $\mathcal{H}$ after observing logged data. Furthermore, as described in Section 2, the logged data are generated from T fixed, possibly deterministic, policies $\pi_1, \ldots, \pi_T$ as described in section 2 rather than a single stochastic policy. As in Section 3 we define the stochastic policy π by Equation 3,

$$\pi(a|x) = \mathbb{E}_{t \sim \mathrm{UNIF}(1,\ldots,T)}[\pi_t(a|x)]$$

The results of Section 3.1 apply to the policy optimization problem. However, note that the data are now assumed to be drawn from the execution of a sequence of T policies $\pi_1, \ldots, \pi_T$, rather than by T draws from π.

Next, we show that it is possible to compete well with the best hypothesis in $\mathcal{H}$ that has adequate support under π (even though the data are not generated from π).

Theorem 3.2. *Let $\hat{\pi}$ be any function from X to distributions over actions A. Let $\mathcal{H}$ be any set of deterministic policies. Define $\tilde{\mathcal{H}} = \{h \in \mathcal{H} \mid \pi(h(x)|x) > \tau,\ \forall\, x \in X\}$ and $\tilde{h} = \mathrm{argmax}_{h \in \tilde{\mathcal{H}}}\{V^h\}$. Let $\hat{h} = \mathrm{argmax}_{h \in \mathcal{H}}\{\hat{V}^h_{\hat{\pi}}\}$ be the hypothesis that maximizes the empirical value estimator defined in Equation 2. Then, with probability at least $1 - \delta$,*

$$V^{\hat{h}} \geq V^{\tilde{h}} - \frac{2}{\tau}\left(\sqrt{\mathbb{E}_x[\mathrm{reg}(x)]} + \sqrt{\frac{\ln(2|H|/\delta)}{2T}}\right), \qquad (6)$$

where $\mathrm{reg}(x)$ is defined, with respect to π, in Equation 5.

The proof of Theorem 3.2 relies on the lower-bound property of our estimator (the left-hand side of Inequality stated in Lemma 3.1). In other words, if $\mathcal{H}$ contains a very good policy that has little support under π, we will not be able to detect that by our estimator. On the other hand, our estimation is safe in the sense that we will never drastically overestimate the value of any policy in $\mathcal{H}$. This "underestimate, but don't overestimate" property is critical to the application of optimization techniques, as it implies we can use an unrestrained learning algorithm to derive a warm start policy.

4 Empirical Evaluation

We evaluated our method on two real-world datasets obtained from Yahoo!. The first dataset consists of uniformly random exploration data, from which an unbiased estimate of any policy can be obtained. This dataset is thus used to verify accuracy of our offline evaluator (2). The second dataset then demonstrates how policy optimization can be done from nonrandom offline data.

4.1 Experiment I

The first experiment involves news article recommendation in the "Today Module", on the Yahoo! front page. For every user visit, this module displays a high-quality news article out of a small candidate pool, which is hand-picked by human editors. The pool contains about 20 articles at any given time. We seek to maximize the click probability (aka click-through rate, or CTR) of the highlighted article. This problem is modeled as a contextual bandit problem, where the context consists of both user and article information, the arms correspond to articles, and the reward of a displayed article is 1 if there is a click and 0 otherwise. Therefore, the value of a policy is exactly its overall CTR. To protect business-sensitive information, we only report normalized CTR (nCTR) which is defined as the ratio of the true CTR and the CTR of a random policy.

Our dataset, denoted D_0, was collected from real traffic on the Yahoo! front page during a two-week period in June 2009. It contains $T = 64.7$M events in the form of triples (x, a, r), where the context x contains user/article features, arm a was chosen *uniformly at random* from a dynamic candidate pool A, and r is a binary reward signal indicating whether the user clicked on a. Since actions are chosen randomly, we have $\hat{\pi}(a|x) = \pi(a|x) \equiv 1/|A|$ and $\mathrm{reg}(x) \equiv 0$. Consequently, Lemma 3.1 implies $\mathbb{E}[\hat{V}^h_{\hat{\pi}}] = V^h$ provided $\tau < 1/|A|$. Furthermore, a straightforward application of Hoeffding's inequality guarantees that $\hat{V}^h_{\hat{\pi}}$ concentrates to V^h at the rate of $O(1/\sqrt{T})$ for any policy h, which is also verified empirically [9]. Given the size of our dataset, therefore, we used this dataset to calculate $\hat{V}_0 = \hat{V}^h_{\hat{\pi}}$ using $\hat{\pi}(a|x) = 1/|A|$ in (2). The result $\hat{V}_0$ was then treated as "ground truth", with which we can evaluate how accurate the offline evaluator (2) is when non-random log data are used instead.

To obtain non-random log data, we ran the LinUCB algorithm using the offline bandit simulation procedure, both from [8], on our random log data D_0 and recorded events (x, a, r) for which LinUCB chose arm a for context x. Note that π is a deterministic learning algorithm, and may choose different arms for the same context at different timesteps. We call this subset of recorded events D_π. It is known that the set of recorded events has the same distribution as if we ran LinUCB on real user visits to Yahoo! front page. We used D_π as non-random log data and do evaluation.

To define the policy h for evaluation, we used D_0 to estimate each article's overall CTR across all users, and then h was defined as selecting the article with highest estimated CTR.

We then evaluated h on D_π using the offline evaluator (2). Since the set A of articles changes over time (with news articles being added and old articles retiring), $\pi(a|x)$ is very small due to the large number of articles over the two-week period, resulting in large variance. To resolve this problem, we split the dataset D_π into subsets so that in each subset the candidate pool remains constant,[3] and then estimate $\pi(a|x)$ for each subset separately using ridge regression on features x. We note that more advanced conditional probability estimation techniques can be used.

Figure 1 plots $\hat{V}^h_{\hat{\pi}}$ with varying τ against the ground truth $\hat{V}_0$. As expected, as τ becomes larger, our estimate can become more (downward) biased. For a large range of τ values, our estimates are reasonably accurate, suggesting the usefulness of our proposed method. In contrast, a naive approach, which assumes $\pi(a|x) = 1/|A|$, gives a very poor estimate of 2.4.

For extremely small values of τ, however, there appears to be a consistent trend of over-estimating the policy value. This is due to the fact that negative moments of a positive random variable are often larger than the corresponding moments of its expectation [7].

Note that the logging policy we used, π, violates one of the assumptions used to prove Lemma 3.1, namely that the exploration policy at timestep t not be dependent on an earlier event. Our offline evaluator is accurate in this setting, which suggests that the assumption may be relaxable in practice.

4.2 Experiment II

In the second experiment, we investigate our approach to the warm-start problem. The dataset was provided by Yahoo!, covering a period of one month in 2008. The data are comprised of logs of events (x, a, y), where each event represents a visit by a user to a particular web page x, from a set of web pages X. From a large set of advertisements A, the commercial system chooses a single ad

[3] We could do so because we know A for every event in D_0.

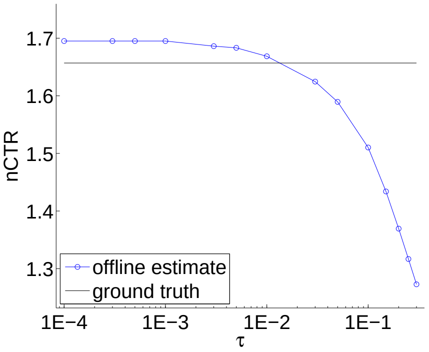

Figure 1: Accuracy of offline evaluator with varying τ values.

Method	τ	Estimate	Interval
Learned	0.01	0.0193	[0.0187,0.0206]
Random	0.01	0.0154	[0.0149,0.0166]
Learned	0.05	0.0132	[0.0129,0.0137]
Random	0.05	0.0111	[0.0109,0.0116]
Naive	0.05	0.0	[0,0.0071]

Figure 2: Results of various algorithms on the ad display dataset. Note these numbers were computed using a not-necessarily-uniform sample of data.

a for the topmost, or most prominent position. It also chooses additional ads to display, but these were ignored in our test. The output y is an indicator of whether the user clicked on the ad or not.

The total number of ads in the data set is approximately $880,000$. The training data consist of 35 million events. The test data contain 19 million events occurring after the events in the training data. The total number of distinct web pages is approximately 3.4 million.

We trained a policy h to choose an ad, based on the current page, to maximize the probability of click. For the purposes of learning, each ad and page was represented internally as a sparse high-dimensional feature vector. The features correspond to the words that appear in the page or ad, weighted by the frequency with which they appear. Each ad contains, on average, 30 ad features and each page, approximately 50 page features. The particular form of f was linear over features of its input (x, a)[4]

The particular policy that was optimized, had an argmax form: $h(x) = \text{argmax}_{a \in C(X)}\{f(x, a)\}$, with a crucial distinction from previous approaches in how $f(x, a)$ was trained. Here $f : X \times A \to [0, 1]$ is a regression function that is trained to estimate probability of click, and $C(X) = \{a \in A \mid \hat{\pi}(a|x) > 0\}$ is a set of feasible ads.

The training samples were of the form (x, a, y), where $y = 1$ if the ad a was clicked after being shown on page x or $y = 0$ otherwise. The regressor f was chosen to approximately minimize the *weighted* squared loss: $\frac{(y-f(x,a))^2}{\max\{\hat{\pi}(a_t|x_t),\tau\}}$. Stochastic gradient descent was used to minimize the squared loss on the training data.

During the evaluation, we computed the estimator on the test data (x_t, a_t, y_t):

$$\hat{V}_{\hat{\pi}}^{h} = \frac{1}{T}\sum_{t=1}^{T}\frac{y_t I(h(x_t) = a_t)}{\max\{\hat{\pi}(a_t|x_t), \tau\}}. \tag{7}$$

As mentioned in the introduction, this estimator is biased due to the use of the parameter $\tau > 0$. As shown in the analysis of Section 3, this bias typically underestimates the true value of the policy h.

We experimented with different thresholds τ and parameters of our learning algorithm.[5] Results are summarized in the Table 2.

The Interval column is computed using the relative entropy form of the Chernoff bound with $\delta = 0.05$ which holds under the assumption that variables, in our case the samples used in the computation of the estimator (Equation 7), are IID. Note that this computation is slightly complicated because the range of the variables is $[0, 1/\tau]$ rather than $[0, 1]$ as is typical. This is handled by rescaling by τ, applying the bound, and then rescaling the results by $1/\tau$.

[4] Technically the feature vector that the regressor uses is the Cartesian product of the page and ad vectors.

[5] For stochastic gradient descent, we varied the learning rate over 5 fixed numbers ($0.2, 0.1, 0.05, 0.02, 0.01$) using 1 pass over the data. We report on the test results for the value with the best training error.

The “Random” policy is the policy that chooses randomly from the set of feasible ads: $\mathrm{Random}(x) = a \sim \mathrm{UNIF}(C(X))$, where $\mathrm{UNIF}(\cdot)$ denotes the uniform distribution.

The “Naive” policy corresponds to the theoretically flawed supervised learning approach detailed in the introduction. The evaluation of this policy is quite expensive, requiring one evaluation per ad per example, so the size of the test set is reduced to 8373 examples with a click, which reduces the significance of the results. We bias the results towards the naive policy by choosing the chronologically first events in the test set (i.e. the events most similar to those in the training set). Nevertheless, the naive policy receives 0 reward, which is significantly less than all other approaches. A possible fear with the evaluation here is that the naive policy is always finding good ads that simply weren’t explored. A quick check shows that this is not correct–the naive argmax simply makes implausible choices. Note that we report only evaluation against $\tau = 0.05$, as the evaluation against $\tau = 0.01$ is not significant, although the reward obviously remains 0.

The “Learned” policies do depend on τ. As suggested by Theorem 3.2, as τ is decreased, the effective set of hypotheses we compete with is increased, thus allowing for better performance of the learned policy. Indeed, the estimates for both the learned policy and the random policy improve when we decrease τ from 0.05 to 0.01.

The empirical click-through rate on the test set was 0.0213, which is slightly larger than the estimate for the best learned policy. However, this number is not directly comparable since the estimator provides a lower bound on the true value of the policy due to the bias introduced by a nonzero τ and because any deployed policy chooses from only the set of ads which are available to display rather than the set of all ads which might have been displayable at other points in time.

The empirical results are generally consistent with the theoretical approach outlined here—they provide a consistently pessimal estimate of policy value which nevertheless has sufficient dynamic range to distinguish learned policies from random policies, learned policies over larger spaces (smaller τ) from smaller spaces (larger τ), and the theoretically unsound naive approach from sounder approaches which choose amongst the the explored space of ads. It would be interesting future work to compare our approach to a full-fledged production online advertising system.

5 Conclusion

We stated, justified, and evaluated theoretically and empirically the first method for solving the warm start problem for exploration from logged data with controlled bias and estimation. This problem is of obvious interest to applications for internet companies that recommend content (such as ads, search results, news stories, etc...) to users.

However, we believe this also may be of interest for other application domains within machine learning. For example, in reinforcement learning, the standard approach to offline policy evaluation is based on importance weighted samples [3, 11]. The basic results stated here could be applied to RL settings, eliminating the need to know the probability of a chosen action explicitly, allowing an RL agent to learn from external observations of other agents.

References

[1] Peter Auer, Nicolò C. Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[2] D. Horvitz and D. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47, 1952.

[3] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. Approximate planning in large pomdps via reusable trajectories. In *NIPS*, 2000.

[4] Diane Lambert and Daryl Pregibon. More bang for their bucks: Assessing new features for online advertisers. In *ADKDD 2007*, 2007.

[5] John Langford, Alexander L. Strehl, and Jenn Wortman. Exploration scavenging. In *ICML-08: Proceedings of the 25rd international conference on Machine learning*, 2008.

[6] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20*, pages 817–824, 2008.

[7] Robert A. Lew. Bounds on negative moments. *SIAM Journal on Applied Mathematics*, 30(4):728–731, 1976.

[8] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the Nineteenth International Conference on World Wide Web (WWW-10)*, pages 661–670, 2010.

[9] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the Fourth International Conference on Web Search and Web Data Mining (WSDM-11)*, 2011.

[10] Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95:135–143, 1998.

[11] Doina Precup, Rich Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *ICML*, 2000.