Unbiased Offline Evaluation of Contextual-bandit-based News Article Recommendation Algorithms

Lihong Li   Wei Chu   John Langford   Xuanhui Wang

Yahoo! Labs

701 First Ave, Sunnyvale, CA, USA 94089

{lihong,chuwei,jl,xhwang}@yahoo-inc.com

ABSTRACT

Contextual bandit algorithms have become popular for on-line recommendation systems such as Digg, Yahoo! Buzz, and news recommendation in general. *Offline* evaluation of the effectiveness of new algorithms in these applications is critical for protecting online user experiences but very challenging due to their "partial-label" nature. Common practice is to create a simulator which simulates the online environment for the problem at hand and then run an algorithm against this simulator. However, creating simulator itself is often difficult and modeling bias is usually unavoidably introduced. In this paper, we introduce a *replay* methodology for contextual bandit algorithm evaluation. Different from simulator-based approaches, our method is completely data-driven and very easy to adapt to different applications. More importantly, our method can provide provably unbiased evaluations. Our empirical results on a large-scale news article recommendation dataset collected from Yahoo! Front Page conform well with our theoretical results. Furthermore, comparisons between our offline replay and online bucket evaluation of several contextual bandit algorithms show accuracy and effectiveness of our offline evaluation method.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: On-line Information Services; I.2.6 [**Computing Methodologies**]: Learning

General Terms

Algorithms, Experimentation

Keywords

Recommendation, multi-armed bandit, contextual bandit, offline evaluation, benchmark dataset

1. INTRODUCTION

Web-based content recommendation services such as Digg, Yahoo! Buzz and Yahoo! Today Module (Figure 1) leverage user activities such as clicks to identify the most attractive contents. One inherent challenge is how to score newly generated contents such as breaking news, especially when the news first emerges and little data are available. A personalized service which can tailor contents towards individual users is more desirable and challenging.

A distinct feature of these applications is their "partial-label" nature: we observe user feedback (click or not) for an article *only when* this article is displayed. A key challenge thus arises which is known as the exploration/exploitation tradeoff: on one hand, we *want to exploit* (i.e., choose articles of higher quality estimates to promote our business of interest), but on the other than, we *have to explore* (i.e., choose articles with lower quality estimates to collect user feedback so as to improve our article selection strategy in the long run). The balance between exploration and exploitation may be modeled as a "contextual bandit" [13]—a subclass of reinforcement learning problems [20], and is also present in many other important Web-based applications such as online ads display and search query suggestion, etc.

An ideal way to evaluate a contextual-bandit algorithm is to conduct a bucket test, in which we run the algorithm to serve a fraction of live user traffic in the real recommendation system. However, not only is this method expensive, requiring substantial engineering efforts in deploying the method in the real system, but it may also have negative impacts on user experience. Furthermore, it is not easy to guarantee replicable comparison using bucket tests as online metrics vary significantly over time. *Offline* evaluation of contextual-bandit algorighms thus becomes valuable when we try to optimize an online recommendation system.

Although benchmark datasets for supervised learning such as the UCI repository have proved valuable for empirical comparison of algorithms, collecting benchmark data towards reliable offline evaluation has been difficult in bandit problems. In our application of news article recommendation on Yahoo! Front Page, for example, each user visit results in the following information stored in the log: user information, the displayed news article, and user feedback (click or not). When using data of this form to evaluate a bandit algorithm offline, we will not have user feedback if the algorithm recommends a different news article than the one stored in the log. In other words, data in bandit-style applications only contain user feedback for recommendations that were actually displayed to the user, but not all candidates. This "partial-label" nature raises a difficulty that is the key difference between evaluation of bandit algorithms and supervised learning ones.

Common practice for evaluating bandit algorithms is to create a simulator and then run the algorithm against it. With this approach, we can evaluate any bandit algorithm without having to run it in a real system. Unfortunately, there are two major drawbacks with this approach. First, creating a simulator can be challenging and time-consuming for practical problems. Second, evaluation results based on artificial simulators may not reflect the actual performance since simulators are only rough approximations of real problems and unavoidably contains modeling bias.

Our contributions are two-fold. First, we describe and study an offline evaluation method for bandit algorithms, which enjoys valuable theoretical guarantees including unbiasedness and accuracy. Second, we verify the method's effectiveness by comparing its evaluation results to online bucket results using a large volume of data recorded from Yahoo! Front Page. These positive results not only encourage wide use of the proposed method in other Web-baesd applications, but also suggest a promising solution to create benchmark datasets from real-world applications for bandit algorithms.

Related Work.

Unbiased evaluation has been studied before under different settings. While our unbiased evaluation method is briefly sketched in an earlier paper [14] and may be interpreted as a special case of the *exploration scavenging* technique [12], we conduct a thorough investigation in this work, including improved theoretical guarantees and positive empirical evidence using online bucket data.

2. CONTEXTUAL BANDIT PROBLEMS

The multi-armed bandit problem [7] is a classic and popular model for studying the exploration-exploitation tradeoff. Despite the simplicity of the model, it has found wide applications in important problems like medical treatment allocation, and recently, in challenging, large-scale problems like Web content optimization [2, 14]. Different from the classic multi-armed bandit problems, we are particularly concerned with a more interesting setting where for each round contextual information is available for decision making.

2.1 Notation

For the purpose of this paper, we consider the multi-armed bandit problem with contextual information. Following previous work [13], we call it a *contextual bandit problem*.[1] Formally, we define by $\mathcal{A} = \{1, 2, \ldots, K\}$ the set of arms, and a contextual-bandit algorithm A interacts with the *world* in discrete trials $t = 1, 2, 3, \ldots$. In trial t:

1. The world chooses a feature vector $\mathbf{x}_t$ known as the *context*. Associated with each arm a is a real-valued payoff $r_{t,a} \in [0, 1]$ that can be related to the context $\mathbf{x}_t$ in an arbitrary way. We denote by $\mathcal{X}$ the (possibly infinite) set of contexts, and $(r_{t,1}, \ldots, r_{t,K})$ the payoff vector. Furthermore, we assume $(\mathbf{x}_t, r_{t,1}, \ldots, r_{t,K})$ is drawn i.i.d. from some unknown distribution D.
2. Based on observed payoffs in previous trials and the current context $\mathbf{x}_t$, A chooses an arm $a_t \in \mathcal{A}$, and

receives payoff r_{t,a_t}. It is important to emphasize here that *no* feedback information (namely, the payoff $r_{t,a}$) is observed for *unchosen* arms $a \neq a_t$.

3. The algorithm then improves its arm-selection strategy with all information it observes, $(\mathbf{x}_{t,a_t}, a_t, r_{t,a_t})$.

In this process, the *total T-trial payoff* of A is defined as

$$G_{\mathsf{A}}(T) \stackrel{\text{def}}{=} \mathbf{E}_D \left[\sum_{t=1}^{T} r_{t,a_t} \right],$$

where the expectation $\mathbf{E}_D[\cdot]$ is defined w.r.t. the i.i.d. generation process of $(\mathbf{x}_t, r_{t,1}, \ldots, r_{t,K})$ according to distribution D (and the algorithm A as well if it is not deterministic). Similarly, given a policy π that maps contexts to actions, $\pi : \mathcal{X} \mapsto \mathcal{A}$, we define its total T-trial payoff by

$$G_{\pi}(T) \stackrel{\text{def}}{=} \mathbf{E}_D \left[\sum_{t=1}^{T} r_{t,\pi(\mathbf{x}_t)} \right] = T \cdot \mathbf{E}_D \left[r_{1,\pi(\mathbf{x}_1)} \right],$$

where the second equality is due to our i.i.d. assumption. Given a reference set Π of policies, we define the *optimal expected T-trial payoff with respect to* Π as

$$G^*(T) \stackrel{\text{def}}{=} \max_{\pi \in \Pi} G_{\pi}(T).$$

For convenience, we also define the per-trial payoff of an algorithm or policy, which is defined, respectively, by

$$g_{\mathsf{A}} \quad \stackrel{\text{def}}{=} \quad \frac{G_{\mathsf{A}}(T)}{T}$$

$$g_{\pi} \quad \stackrel{\text{def}}{=} \quad \frac{G_{\pi}(T)}{T} = \mathbf{E}_D \left[r_{1,\pi(\mathbf{x}_1)} \right].$$

Much research in multi-armed bandit problems is devoted to developing algorithms with large total payoff. Formally, we may search for an algorithm minimizing *regret* with respect to the optimal arm-selection strategy in Π. Here, the T-trial regret $R_{\mathsf{A}}(T)$ of algorithm A with respect to Π is defined by

$$R_{\mathsf{A}}(T) \stackrel{\text{def}}{=} G^*(T) - G_{\mathsf{A}}(T). \tag{1}$$

An important special case of the general contextual bandit problem is the well-known *K-armed bandit* in which the context $\mathbf{x}_t$ remains constant for all t. Since both the arm set and contexts are constant at every trial, they have no effect on a bandit algorithm, and so we will also refer to this type of bandit as a *context-free* bandit.

In the example of news article recommendation, we may view articles in the pool as arms, and for the t-th user visit (trial t), one article (arm) is chosen to serve the user. When the served article is clicked on, a payoff of 1 is incurred; otherwise, the payoff is 0. With this definition of payoff, the expected payoff of an article is precisely its *click-through rate (CTR)*, and choosing an article with maximum CTR is equivalent to maximizing the expected number of clicks from users, which in turn is the same as maximizing the total expected payoff in our bandit formulation.

2.2 Existing Bandit Algorithms

The fundamental challenge in bandit problems is the need for balancing exploration and exploitation. To minimize the regret in Equation (1), an algorithm A *exploits* its past experience to select the arm that appears best. On the other hand, this seemingly optimal arm may in fact be suboptimal, due to imprecision in A's knowledge. In order to avoid

---

[1]In the literature, contextual bandits are sometimes called bandits with covariate [22], associative reinforcement learning [9], bandits with expert advice [6], bandits with side information [21], and associative bandits [19].

this undesired situation, A has to *explore* the world by actually choosing seemingly suboptimal arms so as to gather more information about them (c.f., step 3 in the bandit process defined in the previous subsection). Exploration can increase *short-term* regret since some suboptimal arms may be chosen. However, obtaining information about the arms' average payoffs (i.e., exploration) can refine A's estimate of the arms' payoffs and in turn reduce *long-term* regret. Clearly, neither a purely exploring nor a purely exploiting algorithm works best in general, and a good tradeoff is needed.

There are roughly two classes of bandit algorithms. The first class of algorithms attempt to minimize the regret as the number of steps increases. Formally, such algorithms A ensure the quantity $R_A(T)/T$ vanishes over time as T grows. While low-regret algorithms have been extensively studied for the *context-free K-armed* bandit problem [7], the more general contextual bandit problem has remained challenging. Another class of algorithms are based on Bayes rule, such as Gittins index methods [8]. Such Bayesian approaches may have competitive performance with appropriate prior distributions, but are often computationally prohibitive without coupling with approximation [2].

The Appendix describes a few representative low-regret algorithms used in our experiments, but it should be noted that our method is algorithm *independent*, and so may be applied to evaluate Bayesian algorithms as well.

3. UNBIASED OFFLINE EVALUATION

Compared to machine learning in the more standard supervised learning setting, evaluation of methods in a contextual bandit setting is frustratingly difficult. Our goal here is to measure the performance of a *bandit algorithm* A, that is, a rule for selecting an arm at each time step based on the preceding interactions and current context (such as the algorithms described above). Because of the interactive nature of the problem, it would seem that the only way to do this unbiasedly is to actually run the algorithm online on "live" data. However, in practice, this approach is likely to be infeasible due to the serious logistical challenges that it presents. Rather, we may only have *offline* data available that was collected at a previous time using an entirely *different* logging policy. Because payoffs are only observed for the arms chosen by the logging policy, which are likely to differ from those chosen by the algorithm A being evaluated, it is not at all clear how to evaluate A based only on such logged data. This evaluation problem may be viewed as a special case of the so-called "off-policy policy evaluation problem" in the reinforcement learning literature [17].

One solution is to build a simulator to model the bandit process from the logged data, and then evaluate A with the simulator. Although this approach is straightforward, the modeling step is often very expensive and difficult, and more importantly, it often introduces *modeling bias* to the simulator, making it hard to justify reliability of the obtained evaluation results. In contrast, we propose an approach that is *unbiased*, grounded on data, and simple to implement.

In this section, we describe a sound technique for carrying out such an evaluation, assuming that the individual events are i.i.d., and that the logging policy chose each arm at each time step uniformly at random. Although we omit the details, this latter assumption can be weakened considerably so that any randomized logging policy is allowed and the algo-

---

Algorithm 1 `Policy_Evaluator` (with infinite data stream).

0: Inputs: $T > 0$; bandit algorithm A; stream of events S
1: $h_0 \leftarrow \emptyset$ {An initially empty history}
2: $\hat{G}_A \leftarrow 0$ {An initially zero total payoff}
3: **for** $t = 1, 2, 3, \ldots, T$ **do**
4:     **repeat**
5:         Get next event $(\mathbf{x}, a, r_a)$ from S
6:     **until** $A(h_{t-1}, \mathbf{x}) = a$
7:     $h_t \leftarrow \text{CONCATENATE}(h_{t-1}, (\mathbf{x}, a, r_a))$
8:     $\hat{G}_A \leftarrow \hat{G}_A + r_a$
9: **end for**
10: Output: $\hat{G}_A/T$

---

rithm can be modified accordingly using rejection sampling, but at the cost of decreased data efficiency.

More precisely, we suppose that there is some unknown distribution D from which tuples are drawn i.i.d. of the form $(\mathbf{x}, r_1, \ldots, r_K)$, each consisting of *observed* context and *unobserved* payoffs for all arms. We also posit access to a long sequence of logged events resulting from the interaction of the logging policy with the world. Each such event consists of the context vector $\mathbf{x}$, a selected arm a and the resulting observed payoff r_a. Crucially, only the payoff r_a is observed for the single arm a that was chosen uniformly at random.

Our goal is to use this data to evaluate a bandit algorithm A. Formally, A is a (possibly randomized) mapping for selecting the arm a_t at time t based on the history h_{t-1} of $t-1$ preceding events together with the current context.

It should be noted that this section focuses on contextual bandit problems with constant arm sets of size K. While this assumption leads to easier exposition and analysis, it may not be satisfied in practice. For example, in the news article recommendation problem studied in Section 4, the set of arms is not fixed: new arms may become available while old arms may be dismissed. Consequently, the events are independent but drawn from non-identical distributions. We do not investigate this setting formally although it is possible to generalize our setting in Section 2 to this variable arm set case. Empirically, we find the evaluator is very stable.

3.1 An Unbiased Offline Evaluator

In this subsection, for simplicity of exposition, we take this sequence of logged events to be an *infinitely* long stream. But we also give explicit bounds on the actual finite number of events required by our evaluation method. A variation for finite data streams is studied in the next subsection.

The proposed policy evaluator is shown in Algorithm 1. The method takes as input a bandit algorithm A and a desired number of "valid" events T on which to base the evaluation. We then step through the stream of logged events one by one. If, given the current history h_{t-1}, it happens that the policy A chooses the same arm a as the one that was selected by the logging policy, then the event is retained (that is, added to the history), and the total payoff $\hat{G}_A$ updated. Otherwise, if the policy A selects a different arm from the one that was taken by the logging policy, then the event is entirely ignored, and the algorithm proceeds to the next event without any change in its state.

Note that, because the logging policy chooses each arm uniformly at random, each event is retained by this algorithm with probability exactly $1/K$, independent of every-

thing else. This means that the events which are retained have the same distribution as if they were selected by D. As a result, we can prove that two processes are equivalent: the first is evaluating the policy against T real-world events from D, and the second is evaluating the policy using the policy evaluator on a stream of logged events. Theorem 1 formalizes this intuition.

THEOREM 1. *For all distributions D of contexts and payoffs, all algorithms A, all T, all sequences of events h_T, and all stream S containing i.i.d. events from a uniformly random logging policy and D, we have*

$$\Pr_{Policy\_Evaluator(\mathsf{A},S)}(h_T) = \Pr_{\mathsf{A},D}(h_T).$$

Furthermore, let L be the number of events obtained from the stream to gather the length-T history h_T, then
1. *the expected value of L is KT, and*
2. *for any $\delta \in (0,1)$, with probability at least $1-\delta$, $L \leq 2K(T + \ln(1/\delta))$.*

This theorem says that *every* history h_T has an identical probability in the real world as in the policy evaluator. Any statistics of these histories, such as the estimated per-trial payoff $\hat{G}_{\mathsf{A}}/T$ returned by Algorithm 1, are therefore unbiased estimates of the respective quantities of the algorithm A. Hence, by repeating Algorithm 1 multiple times and then averaging the returned per-trial payoffs, we can accurately estimate the total per-trial payoff g_{A} of any algorithm A and respective confidence intervals. Further, the theorem guarantees that $O(KT)$ logged events are sufficient to retain a sample of size T.

PROOF. The proof is by induction on $t = 1, \ldots, T$ starting with a base case of the empty history which has probability 1 when $t = 0$ under both methods of evaluation. In the inductive case, assume that we have for all $t-1$:

$$\Pr_{Policy\_Evaluator(\mathsf{A},S)}(h_{t-1}) = \Pr_{\mathsf{A},D}(h_{t-1})$$

and want to prove the same statement for any history h_t. Since the data is i.i.d. and any randomization in the policy is independent of randomization in the world, we need only prove that conditioned on the history h_{t-1} the distribution over the t-th event is the same for each process. In other words, we must show:

$$\Pr_{Policy\_Evaluator(\mathsf{A},S)}((\mathbf{x}_t, a, r_{t,a}) \mid h_{t-1})$$
$$= \Pr_D(\mathbf{x}_t, r_{t,a}) \Pr_{\mathsf{A}(h_{t-1})}(a \mid \mathbf{x}_t).$$

Since the arm a is chosen uniformly at random in the logging policy, the probability that the policy evaluator exits the inner loop is identical for any policy, any history, any features, and any arm, implying this happens for the last event with the probability of the last event, $\Pr_D(\mathbf{x}_t, r_{t,a})$. Similarly, since the policy A's distribution over arms is independent conditioned on the history h_{t-1} and features $\mathbf{x}_t$, the probability of arm a is just $\Pr_{\mathsf{A}(h_{t-1})}(a|\mathbf{x}_t)$.

Finally, since each event from the stream is retained with probability exactly $1/K$, the expected number required to retain T events is exactly KT. The high-probability bound is an application of Chernoff's inequality. □

Algorithm 2 `Policy_Evaluator` (with finite data stream).

0: bandit algorithm A; stream of events S of length L
1: $h_0 \leftarrow \emptyset$ {An initially empty history}
2: $\hat{G}_{\mathsf{A}} \leftarrow 0$ {An initially zero total payoff}
3: $T \leftarrow 0$ {An initially zero counter of valid events}
4: **for** $t = 1, 2, 3, \ldots, L$ **do**
5:    Get the t-th event $(\mathbf{x}, a, r_a)$ from S
6:    **if** $\mathsf{A}(h_{t-1}, \mathbf{x}) = a$ **then**
7:       $h_t \leftarrow \text{CONCATENATE}(h_{t-1}, (\mathbf{x}, a, r_a))$
8:       $\hat{G}_{\mathsf{A}} \leftarrow \hat{G}_{\mathsf{A}} + r_a$
9:       $T \leftarrow T + 1$
10:   **else**
11:       $h_t \leftarrow h_{t-1}$
12:   **end if**
13: **end for**
14: Output: $\hat{G}_{\mathsf{A}}/T$

3.2 Sample Complexity Result

Next, we consider a situation that may be more relevant to practical evaluation of a *static* policy when we have a *finite* data set S containing L logged events. Roughly speaking, the algorithm steps through every event in D as in Algorithm 1 and obtains an estimate of the policy's average per-trial payoff based on a *random* number of valid events. The detailed pseudocode in Algorithm 2.

Algorithm 2 is very similar to Algorithm 1. The only difference is that the number of valid events, denoted T in the pseudocode, is a random number with mean L/K. For this reason, the output of Algorithm 2 (namely, $\hat{G}_{\mathsf{A}}/T$) may not be an unbiased estimate of the true per-trial payoff of A. However, the next theorem shows that the final value of T will be arbitrarily close to L/K with high probability as long as L is large enough. Using this fact, the theorem further shows that the returned value of Algorithm 2 is an accurate estimate of the true per-trial payoff with high probability when A is a *fixed* policy that chooses action a_t independent of the history h_{t-1}. To emphasize that A is a fixed policy, the following theorem and its proof use π instead of A.

THEOREM 2. *For all distributions D over contexts and payoffs, all policies π, all data stream S containing L i.i.d. events drawn from a uniformly random logging policy and D, and all $\delta \in (0,1)$, we have, with probability at least $1-\delta$, that*

$$\left| \frac{\hat{G}_\pi}{T} - g_\pi \right| = O\left(\sqrt{\frac{K g_\pi}{L} \ln \frac{1}{\delta}} \right).$$

Therefore, for any $g \geq g_\pi$, with high probability, the theorem guarantees that the returned value $\hat{G}_\pi/T$ is a close estimate of the true value g_π with error on the order of $\tilde{O}\left(\sqrt{Kg/L}\right)$. As L increases, the error decreases to 0 at the rate of $O(1/\sqrt{L})$. This error bound improves a previous result [12, Theorem 5] for a similar offline evaluation algorithm and similarly provides a sharpened analysis for the $T = 1$ special case for policy evaluation in reinforcement learning [10]. Section 4 provides empirical evidence matching our bound.

PROOF. The proof involves a couple applications of the multiplicative Chernoff/Hoeffding bound [15, Corollary 5.2]. To simplify notation, we use $\Pr(\cdot)$ and $\mathbf{E}[\cdot]$ in the proof to

denote the probability and expectation with respect to randomness generated by π and S. Let $(\mathbf{x}_t, a_t, r_{t,a_t})$ be the t-th event in the stream S, V_t be the (random) indicator that a_t matches the arm chosen by policy π in the context $(\mathbf{x}_t)$. Then, $T = \sum_{t=1}^{L} V_t$, $\hat{G}_\pi = \sum_{t=1}^{L} V_t r_{t,a_t}$, and the returned value of Algorithm 2 is $\hat{G}_\pi / T$. We bound the denominator and numerator, respectively.

First, since a_t is chosen uniformly at random, we have $\mathbf{E}[V_t] = 1/K$ for all t and thus $\mathbf{E}\left[\sum_{t=1}^{L} V_t\right] = L/K$. Using the multiplicative form of Chernoff's bound, we have

$$\Pr\left(\left|T - \frac{L}{K}\right| \geq \frac{\gamma_1 L}{K}\right) \leq 2\exp\left(-\frac{L\gamma_1^2}{3K}\right) \qquad (2)$$

for any $\gamma_1 > 0$. Let the right-hand side above be $\delta/2$ and solve for γ_1:

$$\gamma_1 = \sqrt{\frac{3K}{L} \ln \frac{4}{\delta}}.$$

Similarly, since a_t is uniformly chosen, we have $\mathbf{E}\left[\hat{G}_\pi\right] = Lg_\pi/K$. Applying the multiplicative Chernoff bound again, we have for any $\gamma_2 > 0$ that

$$\Pr\left(\left|\hat{G}_\pi - \frac{Lg_\pi}{K}\right| > \frac{\gamma_2 Lg_\pi}{K}\right) \leq 2\exp\left(-\frac{Lg_\pi \gamma_2^2}{3K}\right). \qquad (3)$$

Let the right-hand side above be $\delta/2$ and solve for γ_2:

$$\gamma_2 = \sqrt{\frac{3K}{Lg_\pi} \ln \frac{4}{\delta}}.$$

Now applying a union bound over the probabilistic statements in Equations (2) and (3), we can see that, with probability at least $1 - \delta$, the following holds:

$$\frac{1-\gamma_1}{K} \leq \frac{T}{L} \leq \frac{1+\gamma_1}{K}$$

$$\frac{g_\pi(1-\gamma_2)}{K} \leq \frac{\hat{G}_\pi}{L} \leq \frac{g_\pi(1+\gamma_2)}{K}.$$

These two inequalities together imply

$$\left|\frac{\hat{G}_\pi}{T} - g_\pi\right| \leq \frac{(\gamma_1 + \gamma_2)g_\pi}{1 - \gamma_1} = O\left(\sqrt{\frac{Kg_\pi}{L} \ln \frac{1}{\delta}}\right),$$

which finishes the proof. $\square$

Given Theorem 2, one might wonder if a similar result holds for general bandit *algorithms*. Unfortunately, the following example shows that such a concentration result is impossible in general.

EXAMPLE 3. *Consider a contextual bandit problem with $K = 2$ and $x \in \{0, 1\}$ in which $r_{t,1} = 1$ and $r_{t,2} = 0$ for all $t = 1, 2, \ldots$. Suppose x is defined by a uniform random coin flip. Let A be an algorithm that operates as follows: if $x_1 = 1$ the algorithm chooses $a_t = 1$ for all t; otherwise, it always chooses $a_t = 2$. Therefore, the expected per-trial payoff of A is $g_A = 0.5$. However, in any individual run of the algorithm, its T-step total reward $\hat{G}_A$ is either T (if A always chooses $a_t = 1$) or 0 (if A always chooses $a_t = 0$), and therefore, $\left|\hat{G}_A/T - g_A\right| \equiv 0.5$ no matter how large T is.*

This counterexample shows that an exponential tail style deviation bound does not hold for general bandit algorithms

Figure 1: A snapshot of the "Featured" tab in the Today Module on the Yahoo! Front Page [14]. By default, the article at F1 position is highlighted at the story position.

that are dependent on history. Not all hope is lost though—there are some known algorithms for which deviation bounds are provable; for example, `epoch-greedy` algorithm [13], `UCB1` [5], and `EXP3.P` [6]. Furthermore, as commented earlier, we can always repeat the evaluation process multiple times and then average the outcomes to get accurate estimate of the algorithm's performance. In the next section, we show empirically that Algorithm 1 returns highly stable results for all algorithms we have tried.

4. CASE STUDY

In this section, we apply the offline evaluation method in the previous section to a large-scale, real-world problem with variable arm sets to validate the effectiveness of our offline evaluation methodology. Specifically, we provide empirical evidence for: (i) the unbiasedness guarantee in Theorem 1, (ii) the convergence rate in Theorem 2, (iii) the low variance of the evaluation result, and (iv) the effectiveness of the evaluation method when the arm set may change over time.

While the proposed evaluation methodology has been applied to the same application [14], our focus here is on the effectiveness of the offline evaluation method itself. More importantly, we also provide empirical evidence of unbiasedness for not only fixed policies but also learning algorithms, by relating offline evaluation metric to online performance in large-scale production buckets on Yahoo! Front Page.

We will first describe the application and show how it can be modeled as a contextual bandit problem. Second, we compare the offline evaluation result of a policy to its online evaluation to show our evaluation approach is indeed unbiased and it gives results that are asymptotically consistent when the number of valid events (the quantity T in Algorithms 1 and 2) is large. Third, we provide empirical evidence that our offline evaluation method gives very stable results for a few representative algorithms. Finally, we study the relationship between offline evaluation results to online bucket performance for three bandit algorithms.

4.1 News Article Recommendation on Yahoo! Front Page Today Module

The Today Module is the most prominent panel on the Yahoo! Front Page, which is also one of the most visited

pages on the Internet; see a snapshot in Figure 1. The default "Featured" tab in the Today Module highlights four high-quality news articles, selected from an hourly-refreshed article pool maintained by human editors. As illustrated in Figure 1, there are four articles at footer positions, indexed by F1–F4. Each article is represented by a small picture and a title. One of the four articles is highlighted at the story position, which is featured by a large picture, a title and a short summary along with related links. By default, the article at F1 is highlighted at the story position. A user can click on the highlighted article at the story position to read more details if interested in the article. The event is recorded as a story click. To draw visitors' attention, we would like to rank available articles according to individual interests, and highlight the most attractive article for each visitor at the story position. In this paper, we focus on selecting articles for the story position.

This problem can be naturally modeled as a contextual bandit problem. Here, it is reasonable to assume each user visits and their click probabilities on articles to be (approximately) i.i.d. Furthermore, each user has a set of features (such as age, gender, etc.) from which the click probability of a specific article may be inferred; these features are the contextual information used in the bandit process. Finally, we may view articles in the pool as arms, and the payoff is 1 if the user clicks on the article and 0 otherwise. With this definition of payoff, the expected payoff of an article is precisely its CTR, , and choosing an article with maximum CTR is equivalent to maximizing the expected number of clicks from users, which in turn is the same as maximizing the per-trial payoff g_π in our bandit formulation.

We setup cookie-based buckets for evaluation. A bucket consists of a certain amount of visitors. A cookie is a string of 13 letters randomly generated by the web browser as an identifier. We can specify a cookie pattern to create a bucket. For example, we could let users with the starting letter "a" in their cookies fall in one bucket. In a cookie-based bucket, a user is served by the same policy, unless the user changes the cookie and then belongs to another bucket.

For offline evaluation, millions of events were collected from a "random bucket" from Nov. 1, 2009 to Nov. 10, 2009. In the random bucket, articles are randomly selected from the article pool to serve users. There are about 40 million events in the offline evaluation data set, and about 20 articles available in the pool at every moment.

We focused on user interactions with the story article at the story position only. The user interactions are recorded as two types of events, user visit event and story click event. We chose CTR as the metric of interest, which is defined as the ratio between the number of story click events and the number of user visits. To protect business-sensitive information, we only report relative CTRs which are defined as the ratio between true CTRs and a hidden constant.

4.2 Unbiasedness Analysis

Given a policy, the unbiasedness of the offline evaluation methodology can be empirically verified by comparing offline metrics with online performance. We set up another cookie-based bucket, noted as "serving bucket", to evaluate online performance. In the serving bucket, a spatio-temporal algorithm [3] was deployed to estimate article CTRs.[2] The article with the highest CTR estimate (also known as the

---

[2]Note that the CTR estimates are updated every 5 minutes.

winner article) was then used to serve users. We extracted the serving policy from the "serving bucket", i.e., the best article at every 5 minutes from Nov. 1 2009 to Nov. 10 2009. Note that it is in the same period of time of the offline evaluation data set, ensuring that the sets of available arms are the same in both the serving and random buckets. Then, we used Algorithm 1 to evaluate the serving policy on the events from the random bucket for the offline metric.

It should be noted that the outcome of our experiments are not a foregone conclusion of the mathematics presented, because the setting differs in some ways from the i.i.d. assumption made in our theorems as is typical in real-world applications. In particular, events are not exchangeable since old articles leave the system and new ones enter, there are sometimes unlogged business rule constraints on the serving policy, and users of course do not behave independently when they repeatedly visit the same site. We finesse away this last issue, but the first two are still valid.

In the serving bucket, a winner article usually remains the best for a while. During its winning time, the user repeatedly sees the same article. At the same time, the users in the random bucket are very likely to see different articles at user visit events, due to the random serving policy. It is conceivable that the more a user views the same article, the less likely the user clicks on the article. This conditional effect violates the i.i.d. assumption in Theorem 1. Fortunately, the discrepancy can be removed by considering CTR on *distinct views*. For each user, consecutive events of viewing the same article are counted as one user visit only. The CTR on distinct views in the serving bucket measures user interactions to the winner articles across the whole session. Regarding the offline evaluation metric as in Algorithm 2, the subset of events sampled in the random bucket also measures user interactions with the winner articles across the whole session.

We first compared online and offline per-article CTRs. Only winner articles that were viewed more than $20,000$ times in the serving bucket are used in the plot so that their online CTRs are accurate enough to be treated as ground truth. Figure 2 shows that the CTR metric evaluated offline are very close to the CTR estimated online.

We next compared online and offline CTRs at the policy level. These CTRs are the overall CTR of the serving policy aggregated over all articles. Figure 3 shows the two CTRs are very close on each individual day.

Both sets of results corroborate the unbiasedness guarantee of Theorem 1, a property of particular importance in practice that is almost impossible with simulator-based evaluation methods. Therefore, our evaluation method provides a solution that is accurate (like bucket tests) without the cost and risk of running the policy in the real system.

4.3 Convergence Rate Analysis

We now study how the difference between offline and online CTRs decreases with more data (namely, the quantity T in the evaluation methods). To show the convergence rate, we present the estimated error versus the number of samples used in offline evaluation. Formally, we define the estimated error by $e = |c - \hat{c}|$, where c and $\hat{c}$ are the true CTR and estimated CTR, respectively.

Figures 4 and 5 present convergence rate of the CTR estimate error for various articles and the online serving policy, respectively, and the red curve is $1/\sqrt{T}$—the functional
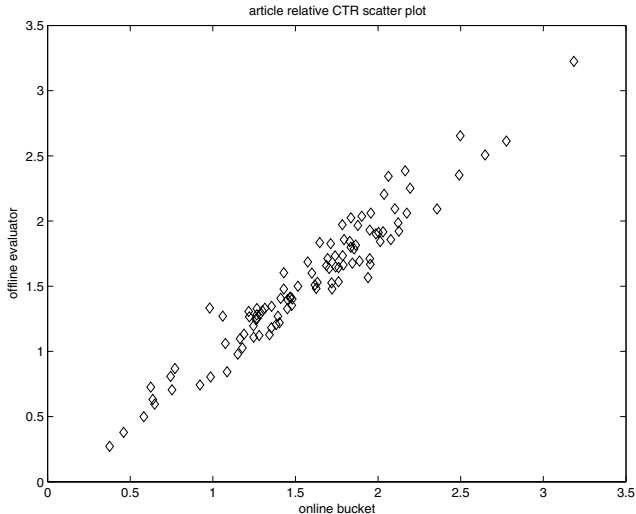
Figure 2: Articles' CTRs in the online bucket versus offline estimates.

Figure 3: Daily overall CTRs in the online bucket versus offline estimates.

algorithm	mean	std	max	min
ϵ-greedy	1.2664	0.0308	1.3079	1.1671
UCB	1.3278	0.0192	1.3661	1.2812
LinUCB	1.3867	0.0157	1.4268	1.3491

Table 1: Statistics of CTR estimates for three representative algorithms using Algorithm 2.

form of the upper confidence bound. These results suggest that, in practice, we can observe the error decay rate predicted by Theorem 2 for reasonably stable algorithms such as those evaluated.

4.4 Low Variance of Evaluation Results

In this subsection, we chose three representative algorithms (c.f., Appendix) to illustrate the low variance of the offline evaluation technique:

- ϵ-greedy, a stochastic, context-free algorithm;
- UCB, a deterministic, context-free variant of UCB1 [5];
- LinUCB [14], a deterministic, contextual bandit algorithm that uses ridge regression to estimate arm payoffs based on contexts.

Each of the algorithms above has one parameter: ϵ for ϵ-greedy and α for UCB and LinUCB (see [14] for details). We fixed the parameters to reasonable values: $\epsilon = 0.4$ and $\alpha = 1$. We collected over $4,000,000$ user visits from a random bucket on May 1, 2009. To evaluate variance, we subsampled this data so that each event is used with probability 0.5. We ran each algorithm 100 times on independently subsampled events and measure the returned CTR using Algorithm 2.

Table 1 summarizes statistics of CTR estimates for the three algorithms.[3] It shows that the evaluation results are highly consistent across different random runs. Specifically, the ratio between standard deviation and the mean CTR is about 2.4% for ϵ-greedy, and below 1.5% for the UCB and LinUCB which have known algorithm-specific deviation bounds.

This experiment demonstrates empirically that our evaluation method can give results that have small variance for a few natural algorithms, despite the artificial counterexample in Section 3, suggesting that with large datasets the result obtained from only one run of our evaluation method are already quite reliable.

---

[3] In the terminology of [14], the CTR estimates reported in Table 1 are for the "learning bucket". Similar standard deviations are found for the so-called "deployment bucket".
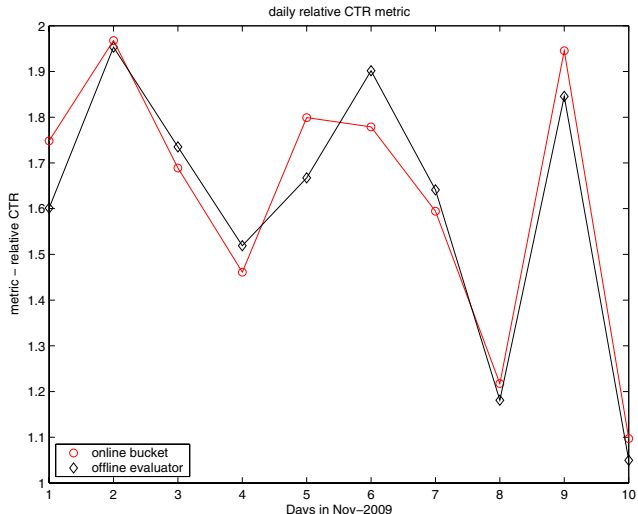
4.5 Consistency with Online Performance

Sections 4.2 and 4.3 give evidence for the accuracy when the offline evaluation method is applied to a static policy that is fixed over time. In this section, we show complimentary accuracy results for *learning algorithms* that may be viewed as history-dependent, non-fixed policies. In particular, we show the consistency between our offline evaluation and online evaluation of three ϵ-greedy bandit models:

- Estimated Most Popular (EMP): we estimate CTR of available articles over all users via a random exploration bucket, and then serve users in the EMP bucket by the article of the highest CTR;
- Segmented Most Popular (SEMP): we segment users into 18 clusters based on their age/gender information. We estimate CTR of available articles within each cluster, and for each user cluster serve the article with the highest CTR. Note that users' feedback may change serving policy for the cluster in future trials;
- Contextual Bandit Model (CEMP): this is a fine-grained personalized model. In this model, we define a separate context for each user based on her age, gender, etc. For each available article, we maintain a logistic regression model to predict its CTR given the user context. When a user comes, in the CEMP bucket, we estimate the CTRs of all articles for the user and select the article with highest estimated CTR to display. Users with different contexts may be served by different articles in this bucket, while each user's feedback will affect other users' click probability estimation on this article in future trials.
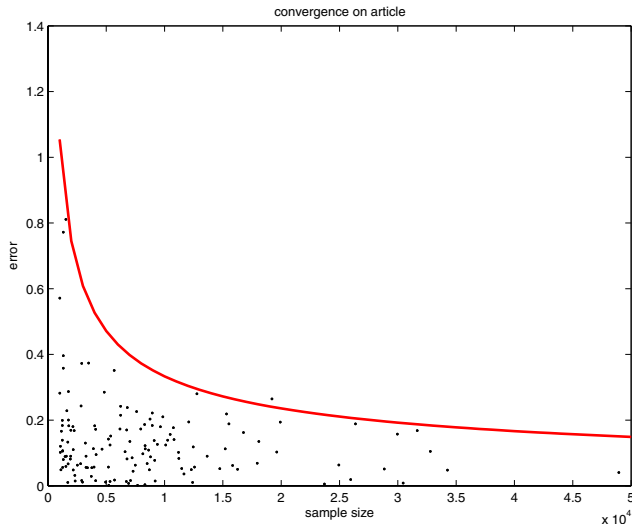
Figure 4: Decay rate of error in articles' CTR estimates with increasing data size. The red curve plots the function $1/\sqrt{x}$.

Figure 5: Decay rate of error in overall CTR estimates with increasing data size. The red curve plots the function $1/\sqrt{x}$.

For all these three bandit models, we set up three online bcookie-based buckets to deploy the three bandit models respectively. We also set up another bucket to collect random exploration data. This random data is used to update the states of three online bandit models and also used for our offline evaluation. For a given period, we obtain the per-trial payoffs $g_\mathsf{A}^\text{online}$ for $\mathsf{A} \in \{\mathsf{EMP}, \mathsf{SEMP}, \mathsf{CEMP}\}$. Using data in the random exploration bucket, we run our offline evaluation for these three models in the same period and get the per-trial payoffs $g_\mathsf{A}^\text{offline}$.

It is important to note that there were unlogged business-rule constraints in all online serving buckets of Today Module; for instance, an article may be forced to shown in a given time window. Fortunately, our data analysis (not reported here) suggested that such business rules have roughly the same multiplicative impact on an algorithm's online CTR, although this multiplicative factor may vary across different days. To remove effects caused by business rules, we report the ratio of offline CTR estimate and online CTR for each model: $\rho_\mathsf{A} = g_\mathsf{A}^\text{offline}/g_\mathsf{A}^\text{online}$. If our offline evaluation metric is truthful of an algorithm's online metric in the absence of business rules, then it is expected that, for a given period of time like one day, ρ_A should remain constant ideally and does not depend on the algorithm A.

In Figure 6, we present a scatter plot of ρ_EMP vs. ρ_SEMP of 16 days, from May 03, 2009 to May 18, 2009. In each day, we have about $2,000,000$ views (i.e., user visits) in each of the four online buckets. The scatter plot indicates a strong linear correlation. The slope in least squares linear regression is 1.019 and the standard deviation in residue vector is 0.0563. We observed that business rules give almost the same impact on CTR in buckets for the two serving policies.

SEMP is a relatively simple bandit algorithm similar to EMP. In the next experiment, we study the online/offline correlation of a more complicated contextual bandit serving policy CEMP, in which CTRs are estimated using logistic regression on user features and a separate logistic regression model is maintained for each article. Figure 7 shows the scatter plot
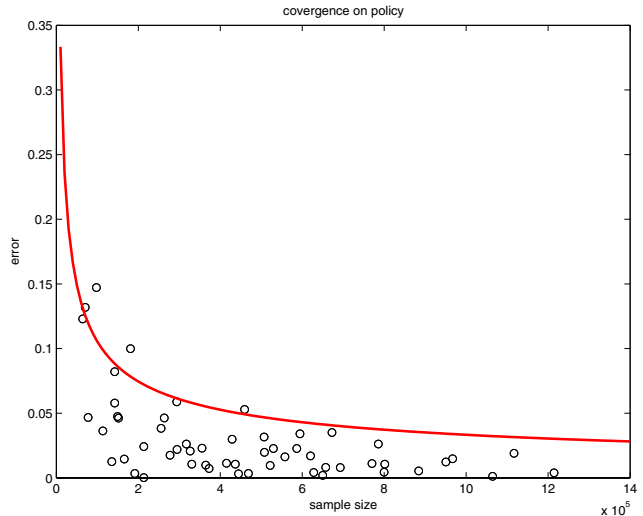
of ρ_EMP vs ρ_CEMP in 18-day data from May 22, 2010 to June 8, 2010. In each day, we have $2,000,000 \sim 3,000,000$ views in each online bucket. The scatter plot again indicates a strong linear correlation. In this comparison, the slope and standard deviation in residue vector is 1.113 and 0.075 respectively. It shows that the difference between our offline and online evaluation, caused by business rules and other systemic factors, e.g. time-out in user feature retrieval and delays in model update, is comparable across bandit models. Although the daily factor is unpredictable, the relative performance of bandit models in offline evaluation is reserved in online buckets. Thus, our offline evaluator can provide reliable comparison of different models on historical data, even in the presence of business rules.

5. CONCLUSIONS AND FUTURE WORK

This paper studies an offline evaluation method of bandit algorithms that relies on log data directly rather than on a simulator. The only requirement of this method is that the log data is generated i.i.d. with arms chosen by an (ideally uniformly) random policy. We show that the evaluation method gives unbiased estimates of quantities like total payoffs, and also provide a sample complexity bound for the estimated error when the algorithm is a fixed policy. The evaluation method is empirically validated using real-world data collected from Yahoo! Front Page for the challenging application of online news article recommendation. Empirical results verify our theoretical guarantees, and demonstrate both accuracy and stability of our method using real online bucket results. These encouraging results suggest the usefulness of our evaluation method, which can be easily applied to other related applications such as online refinement of ranking results [16] and ads display.

Our evaluation method, however, ignores $(K-1)/K$ fraction of logged data. Therefore, it does not make use of *all* data, which can be a problem when K is large or when data is expensive to obtain. Furthermore, in some risk-sensitive applications, while we can inject some randomness
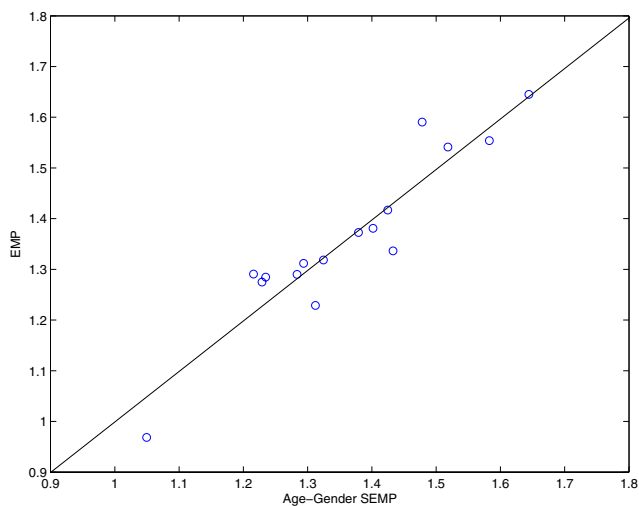
Figure 6: Scatter plot of ratios of offline metric and online bucket performance of 16 days in 2009.
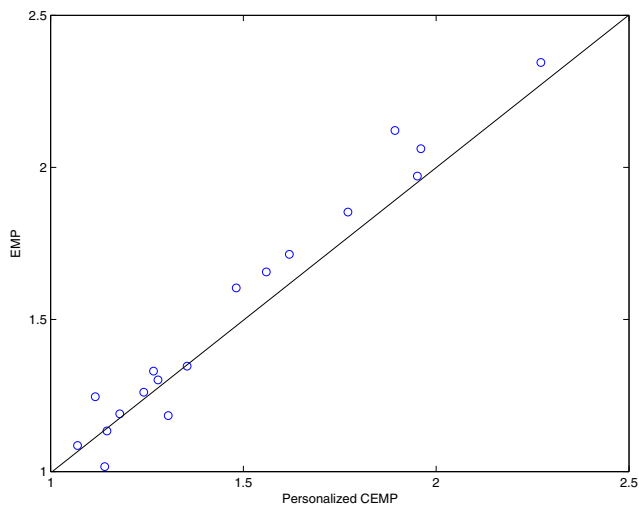

Figure 7: Scatter plot of ratios of offline metric and online bucket performance in 18 days in 2010.

during data collection, a uniformly random policy might be too much to hope for due to practical constraints (such as user satisfaction). As we mentioned earlier, our evaluation method may be extended to work for data collected by any random policy with rejection sampling, which enjoys similar unbiasedness guarantees, but reduces the data efficiency at the same time. An interesting future direction, therefore, is exploiting problem-specific structures to avoid exploration of the full arm space. A related question is how to make use of *non-random* data for reliable offline evaluation, for which a recent progress has been made [18].

Acknowledgements

6. REFERENCES

[1] Naoki Abe, Alan W. Biermann, and Philip M. Long. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37(4):263–293, 2003.

[2] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. Explore/exploit schemes for web content optimization. In *Proceedings of the Ninth International Conference on Data Mining*, 2009.

[3] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. Spatio-temporal models for estimating click-through rate. In *Proceedings of the Eighteenth International Conference on World Wide Web*, 2009.

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

[5] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

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

[7] Donald A. Berry and Bert Fristedt. *Bandit Problems: Sequential Allocation of Experiments*. Monographs on Statistics and Applied Probability. Chapman and Hall, 1985.

[8] J.C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41:148–177, 1979.

[9] Leslie Pack Kaelbling. Associative reinforcement learning: Functions in k-DNF. *Machine Learning*, 15(3):279–298, 1994.

[10] Michael J. Kearns, Yishay Mansour, and Andrew Y. Ng. Approximate planning in large POMDPs via reusable trajectories. In *Advances in Neural Information Processing Systems 12*, 2000.

[11] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[12] John Langford, Alexander L. Strehl, and Jennifer Wortman. Exploration scavenging. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning*, pages 528–535, 2008.

[13] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Advances in Neural Information Processing Systems 20*, 2008.

[14] 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*, 2010.

[15] Colin McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, volume 141 of *London Mathematical Society Lecture Notes*, pages 148–188. Cambridge University Press, 1989.

[16] Taesup Moon, Lihong Li, Wei Chu, Ciya Liao, Zhaohui Zheng, and Yi Chang. Online learning for recency search ranking using real-time user feedback. In *Proceedings of the Nineteenth International Conference on Knowledge Management*, 2010.

[17] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 759–766, 2000.

[18] Alexander L. Strehl, John Langford, Lihong Li, and Sham M. Kakade. Learning from logged implicit exploration data. In *Advances in Neural Information Processing Systems 23*, 2011.

[19] Alexander L. Strehl, Chris Mesterharm, Michael L. Littman, and Haym Hirsh. Experience-efficient learning in associative bandit problems. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 889–896, 2006.

[20] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, March 1998.

[21] Chih-Chun Wang, Sanjeev R. Kulkarni, and H. Vincent Poor. Bandit problems with side observations. *IEEE Transactions on Automatic Control*, 50(3):338–355, 2005.

[22] Michael Woodroofe. A one-armed bandit problem with a concomitant variable. *Journal of the American Statistics Association*, 74(368):799–806, 1979.

APPENDIX

One of the simplest and most straightforward algorithms is ϵ-`greedy`. In each trial t, this algorithm first estimates the average payoff $\hat{\mu}_{t,a}$ of each arm a. Then, with probability $1 - \epsilon$, it chooses the *greedy* arm (e.g., the arm with highest payoff estimate): $a_t = \arg\max_a \hat{\mu}_{t,a}$; with probability ϵ, it chooses a random arm. In the limit, each arm will be tried infinitely often, and so the payoff estimate $\hat{\mu}_{t,a}$ converges to the true value μ_a with probability 1 as $t \to \infty$. Furthermore, by decaying ϵ appropriately, the per-step regret, $R_A(T)/T$, converges to 0 with probability 1.

In contrast to the *unguided* ϵ-`greedy` strategy, another class of algorithms generally known as upper confidence bound algorithms [11, 5] use a smarter way to balance exploration and exploitation. Specifically, in trial t, these algorithms estimate both the mean payoff $\hat{\mu}_{t,a}$ of each arm a as well as a corresponding confidence interval $c_{t,a}$, so that $|\hat{\mu}_{t,a} - \mu_a| < c_{t,a}$ holds with high probability. They then select the arm that achieves a highest upper confidence bound (UCB for short): $a_t = \arg\max_a (\hat{\mu}_{t,a} + \alpha c_{t,a})$, where α is a tunable parameter. With appropriately defined confidence intervals and parameter α, it can be shown that such algorithms have a small total T-trial regret that is only logarithmic in the total number of trials T [11].

While context-free K-armed bandits are extensively studied and well understood, the more general contextual bandit problem has remained challenging. The `EXP4` algorithm [6] uses the exponential weighting technique to achieve an $\tilde{O}(\sqrt{T})$ regret in expectation, where $\tilde{O}(x) \stackrel{\text{def}}{=} O(x \ln x)$, even if the sequence of contexts and payoffs are chosen by an adversarial world, but the computational complexity may be exponential in the number of features in general. Another general contextual bandit algorithm is the `epoch-greedy` algorithm [13] that is similar to ϵ-`greedy` with adaptively shrinking ϵ. Assuming the sequence of contexts, $\mathbf{x}_1, \ldots, \mathbf{x}_T$, is i.i.d., this algorithm is computationally efficient given an oracle empirical risk minimizer but has the weaker regret guarantee of $\tilde{O}(T^{2/3})$ in general, with stronger guarantees in various special cases.

Algorithms with stronger regret guarantees may be designed under various modeling assumptions about the contextual bandit. Assuming the expected payoff of an arm is linear in its features (namely, $\mathbf{E}_D[r_{t,a} \mid \mathbf{x}_{t,a}] = \mathbf{w}^\top \mathbf{x}_{t,a}$ for some coefficient vector $\mathbf{w}$), both `LinRel` [4] and `LinUCB` [14] are essentially UCB-type approaches generalized to linear payoff functions, and their variants have a regret of $\tilde{O}(\sqrt{T})$, a significant improvement over earlier algorithms [1] as well as the more general `epoch-greedy` algorithm.