Non-Stationary Off-Policy Optimization

Joey Hong Branislav Kveton Manzil Zaheer Yinlam Chow Amr Ahmed
Google Research

Abstract

Off-policy learning is a framework for evaluating and optimizing policies without deploying them, from data collected by another policy. Real-world environments are typically non-stationary and the offline learned policies should adapt to these changes. To address this challenge, we study the novel problem of off-policy optimization in piecewise-stationary contextual bandits. Our proposed solution has two phases. In the offline learning phase, we partition logged data into categorical latent states and learn a near-optimal sub-policy for each state. In the online deployment phase, we adaptively switch between the learned sub-policies based on their performance. This approach is practical and analyzable, and we provide guarantees on both the quality of off-policy optimization and the regret during online deployment. To show the effectiveness of our approach, we compare it to state-of-the-art baselines on both synthetic and real-world datasets. Our approach outperforms methods that act only on observed context.

1 Introduction

When users interact with online platforms, such as search engines or recommender systems, their behavior is often guided by certain contexts that the system cannot directly observe. Examples of these contexts include *user preferences*, or in shorter term, *user intent*. As the user interacts with the system, these contexts are slowly revealed based on the actions and responses of the user. A good recommender system should be able to utilize these contexts to update the recommendation actions accordingly.

One popular framework to learn recommendation actions conditioned on contexts is using contextual bandits (Lattimore and Szepesvári, 2019). In contextual bandits, an agent (or policy) chooses an action based on current contexts and the feedback observed in previous rounds. Contextual bandits have been applied to many core machine learning systems, including search engines, recommender systems, and ad placement (Li et al., 2010; Bottou et al., 2013).

Contextual bandit algorithms are either *on-policy*, where the agent learns using online, real-world interactions (Langford and Zhang, 2008; Abbasi-yadkori et al., 2011), or *off-policy*, where the learning process uses offline logged data collected from previous policies (Strehl et al., 2010; Li et al., 2010). While the former is more straightforward, the latter is more suitable for applications where sub-optimal interactions are costly and may lead to costly outcomes.

Most existing contextual bandit algorithms assume that rewards are sampled from a stationary conditional distribution. While this is a valid assumption in simpler problems, where the user intents remain static during interactions, in general the environment should be non-stationary, where user preferences may change during the interactions due to some unexpected events. These shifts in the environment can either be smooth (Beshes et al., 2014) or abrupt at certain points in time (Hartland et al., 2007b). Here we mainly focus on the latter case, known as the *piecewise-stationary* environment (Hartland et al., 2007b; Garivier and Moulines, 2008), which is applicable to many event-sensitive decision-making problems.

Both non-stationary bandits (Auer et al., 2002; Luo et al., 2018) and, more specifically, piecewise-stationary bandits (Hartland et al., 2007a; Garivier and Moulines, 2008; Yu and Mannor, 2009) have been studied extensively in prior work, but in the on-policy setting. Prior work in non-stationary off-policy learning has only considered policy evaluation via learning the evolution of contexts via time series forecasting (Thomas et al., 2017) or weighting past observations (Jagerman et al., 2019). Neither of these methods consider policy optimization.

In this work, we develop a principled off-policy method to learn a piecewise-stationary contextual bandit policy with performance guarantees. Our algorithm consists of both the offline and online learning phases. In the offline phase, the piecewise-stationarity is modeled with a categorical latent state, whose evolution is either modeled by a change-point detector (Liu et al., 2018; Cao et al., 2019) or a hidden

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

Markov model (HMM) (Baum and Petrie, 1966). At each latent state, a corresponding policy is learned from a subset of offline data associated with that state. With the set of policies learned offline, the online phase then selects which policy to deploy based on a mixture-of-experts (Auer et al., 2002; Luo et al., 2018) online learning approach. We derive high-probability bounds on the off-policy performance of the learned policies and also analyze the regret of the online policy deployment. Finally, the effectiveness of our approach is demonstrated in both synthetic and real-world experiments, where we outperform existing off-policy contextual bandit baselines. The novel challenges we tackle are two-fold. First, we are the first to consider the bias in off-policy estimation due to not knowing the latent state. Second, deploying a non-stationary policy learned offline is nontrivial; we are first to propose a framework of learning the components of a switching policy offline, then augmenting the components with an adaptive switching algorithm online.

2 Background

Let $\mathcal{X}$ be a set of contexts and $\mathcal{A} = [K]$ be a set of actions. A typical contextual bandit setting consists of an agent interacting with a stationary environment over T rounds. In round $t \in [T]$, context $x_t \in \mathcal{X}$ is sampled from an unknown distribution P^{x}. Then, conditioned on x_t, the agent chooses an action $a_t \in \mathcal{A}$. Finally, conditioned on x_t and a_t, a reward $r_t \in [0, 1]$ is sampled from an unknown distribution $P^{\mathrm{r}}(\cdot \mid x_t, a_t)$.

Now, let $\mathcal{H}$ be the set of *stochastic stationary policies* $\mathcal{H} = \{\pi : \mathcal{X} \to \Delta^{K-1}\}$, where Δ^{K-1} is the $(K-1)$-dimensional simplex. We use shorthand $x, a, r \sim P, \pi$ to denote a triplet sampled as $x \sim P^{\mathrm{x}}, a \sim \pi(\cdot \mid x)$, and $r \sim P^{\mathrm{r}}(\cdot \mid x, a)$. We define

$$\mathbb{E}_{x,a,r\sim P,\pi}\left[r\right] = \mathbb{E}_{x\sim P^{\mathrm{x}}}\mathbb{E}_{a\sim\pi(\cdot|x)}\mathbb{E}_{r\sim P^{\mathrm{r}}(\cdot|x,a)}\left[r\right] .$$

With this notation, the expected reward of policy $\pi \in \mathcal{H}$ in round t can be written

$$V_t(\pi) = \mathbb{E}_{x_t,a_t,r_t\sim P,\pi}\left[r\right] .$$

Traditionally, $V_t(\pi)$ is the same for all rounds t.

In off-policy learning, actions are drawn by a known, stationary logging policy $\pi_0 \in \mathcal{H}$. Logged data is collected in the form of tuples,

$$\mathcal{D} = \{(x_1, a_1, r_1, p_1), \dots, (x_T, a_T, r_T, p_T)\} ,$$

where $x_t, a_t, r_t \sim P, \pi_0$ and $p_t = \pi_0(a_t \mid x_t)$ is the probability that the logging policy takes action a_t under context x_t. For simplicity, we assume that π_0 is known. Note that if the logging policy is not known, a stationary π_0 can be estimated from logged data to approximate the true logging policy (Strehl et al., 2010; Xie et al., 2019; Chen et al., 2019a). Off-policy learning focuses on two tasks: evaluation and optimization.

2.1 Off-Policy Evaluation

The goal is to estimate the expected reward of a target policy $\pi \in \mathcal{H}$, $V(\pi) = \sum_{t=1}^{T} V_t(\pi)$, from logged data $\mathcal{D}$. One popular approach is *inverse propensity scoring (IPS)* (Horvitz and Thompson, 1952), which reweighs observations with importance weights as

$$\hat{V}(\pi) = \sum_{t=1}^{T} \min\left\{M, \frac{\pi(a_t \mid x_t)}{p_t}\right\} r_t .$$

When the clipping parameter $M = \infty$, the IPS estimator is unbiased, that is $\mathbb{E}_{x,a,r\sim P,\pi_0}[\hat{V}(\pi)] = V(\pi)$. But its variance could be unbounded if the target and logging policies differ substantially. The clipping parameter M trades off variance due to differences in target and logging policies for bias from underestimating the reward (Ionides, 2008; Bottou et al., 2013), and there are methods to design the clipping weight to optimize such trade-off (Dudik et al., 2011; Wang et al., 2017). While we focus on the IPS estimator, our work can be incorporated into other estimators, such as the Direct Method (DM) and Doubly Robust (DR) estimator (Dudik et al., 2011), which leverage a reward model $\hat{r}(x, a) \simeq \mathbb{E}_{r\sim P^{\mathrm{r}}(\cdot|x,a)}\left[r\right]$, where $\simeq$ denotes approximation by fitting on $\mathcal{D}$.

2.2 Off-Policy Optimization

The goal is to find a policy with the maximum reward, $\pi^* = \arg\max_{\pi\in\mathcal{H}} V(\pi)$. One popular solution is to directly maximize the off-policy IPS estimate, $\hat{\pi} = \arg\max_{\pi\in\mathcal{H}} \hat{V}(\pi)$ (Chen et al., 2019b). For stochastic policies, one often optimizes an entropy-regularized estimate (Chen et al., 2019b),

$$\hat{\pi} = \arg\max_{\pi\in\mathcal{H}} \hat{V}(\pi) - \tau \sum_{t=1}^{T}\sum_{a\in\mathcal{A}} \pi(a \mid x_t) \log \pi(a \mid x_t) ,$$

where $\tau \geq 0$ is the *temperature* parameter that controls the determinism of the learned policy. That is, as $\tau \to 0$, the policy chooses the maximum. Following prior work (Swaminathan and Joachims, 2015b,a), one class of policies that solves this entropy-regularized objective is the linear soft categorical policy $\pi(a \mid x; \theta) \propto \exp(\theta^T f(x, a))$, where $\theta \in \mathbb{R}^d$ is the weight of the linear function approximation w.r.t. the joint feature maps of context and action $f(x, a) \in \mathbb{R}^d$. In the special case that $\mathcal{X}$ is finite, $f(x, a)$ becomes an indicator vector for each pair (x, a), and solving $\hat{\pi}$ reduces to an LP (Li et al., 2018).

3 Setting

In non-stationary bandits, the context and reward distributions change with round t. To model this, we consider an extended contextual bandit setting where the context and reward distributions also depend on a discrete latent variable $z \in \mathcal{Z}$, where $\mathcal{Z} = [L]$ is the set of L latent states.

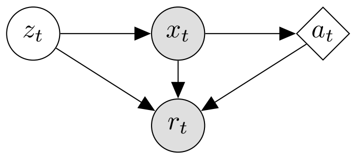


Figure 1: Graphical model for latent state z_t, context x_t, action a_t, and reward r_t.

We denote by $z_t \in \mathcal{Z}$ the latent state in round t, and by $z_{1:T} = (z_t)_{t=1}^T \in \mathcal{Z}^T$ its sequence over the logged data. We consider $z_{1:T}$ to be fixed but unknown. For analysis, we assume that L is known, but relax this assumption and tune L in the experiments. We also assume that the latent state is unaffected by the actions of the agent, a key difference from reinforcement learning (RL). In search engines, for instance, latent states could be different user intents that change over time, such as $\mathcal{Z} = \{\text{news}, \text{shopping}, \dots\}$.

We can modify our earlier notation to account for the latent state. Let P_z^{x} and P_z^{r} be the corresponding context and reward distributions conditioned on z. Then the expected reward of policy π at round t is $V_t(\pi) = \mathbb{E}_{x,a,r \sim P_{z_t}, \pi}[r]$. The relation between all variables can be summarized in a graphical model in Figure 1. Revisiting our search engine example, if a system knew that the user shops, it would likely recommend products to buy. Hence, instead of policies that only act on observed context, we should consider policies that also act according to the latent state. We define a new class $\mathcal{H}^{\mathcal{Z}}$ that consists of members $\Pi = (\pi_z)_{z \in \mathcal{Z}}$, where individual $\pi_z \in \mathcal{H}$ are stationary policies. We define the value of Π as $V(\Pi) = \sum_{z \in \mathcal{Z}} V_z(\pi_z)$, where $V_z(\pi_z) = \sum_{t=1}^T \mathbb{1}[z_t = z] \, V_t(\pi_z)$ is the value of π_z under the subset of logged data whose latent state is z.

Prior works on non-stationary bandits either studied environments with *smooth changes* (Beshes et al., 2014), or *piecewise-stationary* environments, where the changes are abrupt at a fixed number of unknown *change-points* (Hartland et al., 2007b; Garivier and Moulines, 2008). In this work we focus on the latter environment. In a piecewise-stationary environment, we additionally denote S as the number of stationary segments in $z_{1:T}$, where the latent state is constant over a segment. Here we assume $S \geq L$, as multiple segments can map to the same latent state, is small. We denote the change-points as $\tau_1 < \ldots < \tau_{S-1}$, and let $\tau_0 = 1$ and $\tau_S = T$ to simplify exposition.

4 Off-Policy Evaluation

To extend off-policy learning to the piecewise-stationary latent setting, we consider IPS estimator for $\Pi \in \mathcal{H}^Z$

$$\hat{V}(\Pi) = \sum_{z \in \mathcal{Z}} \hat{V}_z(\pi_z)\,, \tag{1}$$

$$\hat{V}_z(\pi_z) = \sum_{t=1}^T \mathbb{1}[\hat{z}_t = z] \cdot \min\left\{M, \frac{\pi_z(a_t \mid x_t)}{p_t}\right\} r_t$$

where $\hat{V}_z(\pi_z)$ is the IPS estimator that corresponds to the part of the logged data whose latent state is z, and $\hat{z}_{1:T}$ is a sequence of latent states predicted by some *oracle* O. This estimator partitions the logged data by latent state.

For simplicity, we restrict our performance analysis to a set of policies where the clipping condition is always satisfied,

$$\mathcal{H} = \left\{\pi : \frac{\pi(a \mid x)}{\pi_0(a \mid x)} \leq M \text{ for all } a \in \mathcal{A}, x \in \mathcal{X}\right\}\,, \tag{2}$$

so that the propensity score does not needed to be clipped. Extending this analysis to a general policy class is relatively straightforward, and it only adds an extra bias term to the performance bound (Ionides, 2008; Li et al., 2018). We omit this for the sake of brevity.

If the oracle accurately predicts all the ground-truth latent states, i.e., $\hat{z}_{1:T} = z_{1:T}$, and if $M = \infty$, then the following lemma shows that the IPS estimator $\hat{V}(\pi)$ is unbiased.

Lemma 1. *For any $\Pi \in \mathcal{H}^Z$, the IPS estimator $\hat{V}(\Pi)$ in (1) is unbiased when $\hat{z}_{1:T} = z_{1:T}$.*

Proof. From definition of $\hat{V}(\Pi)$ in (1), we have

$$\begin{aligned} V(\Pi) &= \sum_{t=1}^T V_t(\pi_{z_t}) = \sum_{t=1}^T \mathbb{E}_{x_t, a_t, r_t \sim P_{z_t}, \pi_0}\left[\frac{\pi_{z_t}(a_t \mid x_t)}{p_t} r_t\right] \\ &= \mathbb{E}\left[\sum_{t=1}^T \frac{\pi_{z_t}(a_t \mid x_t)}{p_t} r_t\right] = \mathbb{E}\left[\sum_{t=1}^T \hat{V}(\pi_{z_t})\right] \\ &= \mathbb{E}\left[\hat{V}(\Pi)\right]\,, \end{aligned}$$

where the last expectation is over all $x_t, a_t, r_t \sim P_{z_t}, \pi_0$, for any $t \in [T]$. $\square$

While the above technical result justifies our choice of the IPS estimator for piecewise-stationary environments, in reality there is no practical way to ensure a perfect latent state estimation because the latent states $z_{1:T}$ are not observed in logged data $\mathcal{D}$. To tackle this challenge, in the following we instead assume the latent state oracle O has a low prediction error with high probability and show how this error propagates into off-policy value estimation.

Assumption 1. *For any $z_{1:T}$ and $\delta \in (0, 1]$, oracle O estimates $\hat{z}_{1:T}$ such that $\sum_{t=1}^T \mathbb{1}[\hat{z}_t \neq z_t] \leq \varepsilon(T, \delta)$ holds with probability at least $1 - \delta$, where $\varepsilon(T, \delta) = o(T)$ is some function of T and δ.*

Utilizing this assumption, we now provide an upper bound on the estimation error (whose proof is in Appendix A) of the IPS estimator in (1) where the latent state prediction is generated by an oracle O that satisfies Assumption 1.

Lemma 2. *For any policy $\Pi \in \mathcal{H}^Z$, its IPS estimate $\hat{V}(\Pi)$ in (1), and true value $V(\Pi)$, we have that*

$$|V(\Pi) - \hat{V}(\Pi)| \leq M\varepsilon(T, \delta_1/2) + M\sqrt{2T\log(4/\delta_2)}$$

holds with probability at least $1 - \delta_1 - \delta_2$.

This technical lemma shows that in a piecewise-stationary environment, the error of an IPS off-policy value estimator can be decomposed into the latent oracle prediction error and a statistical error term that is sublinear in T. For the remaining part of this section, we introduce two latent prediction oracles. The first one is based on change-point detection, and we will show that it satisfies Assumption 1. The second one is based on a *hidden Markov model* (HMM), for which we do not prove an estimation error bound but get better empirical performance.

4.1 Change-Point Detector

In this section, we propose and analyze a change-point detector oracle that satisfies Assumption 1. First, we assume a one-to-one mapping between stationary segments and latent states, or $S = L$. We let $z_{1:T}$ form a non-decreasing sequence of integers that satisfies $z_1 = 1$, $z_T = S$ with $|z_{t+1} - z_t| \leq 1, \forall t \in [T-1]$, and change-points $\tau_0 = 1 < \tau_1 < \ldots < \tau_{S-1} < T = \tau_S$. In practice, this could over-segment the offline data, if multiple stationary segments can be modeled by the same latent state, but this assumption is only used for analysis.

We also assume that changes are *detectable*. This means that the difference in performance of a stationary logging policy before and after the change-point exceeds some threshold.

Assumption 2. *For each segment $i \in [S]$ there exists a threshold $\Delta > 0$ such that the difference of values between two consecutive change points is greater than Δ, i.e.* $|V_{\tau_i}(\pi_0) - V_{\tau_i - 1}(\pi_0)| \geq \Delta$.

Similar assumptions are common in piecewise-stationary bandits, where the state-of-the-art algorithms (Liu et al., 2018; Cao et al., 2019) use an online change-point detector to detect change points and reset the parameters of the bandit algorithm upon a change. In this work, we utilize a similar idea but in an offline off-policy setting. We construct a change-point detector oracle O with window size w and detection threshold c (Algorithm 1).

At a high-level, O computes difference statistics for each round in the offline data and iteratively chooses the round with the highest statistic, declaring that a change-point, and removing any nearby rounds from consideration. This procedure continues until there is no statistic that lies above threshold c. In the following we state a latent prediction error bound for this oracle, which is derived in Appendix B.

Theorem 1. *Let $\tau_i - \tau_{i-1} > 4w$ for all $i \in [L]$. For any $\delta \in (0, 1]$, and c and w in Algorithm 1 such that*

$$\Delta/2 \geq c \geq \sqrt{2\log(8T/\delta)/w}\,,$$

then Algorithm 1 estimates $\hat{z}_{1:T}$ so $\sum_{t=1}^{T} \mathbb{1}[\hat{z}_t \neq z_t] \leq Sw$ holds with probability at least $1 - \delta$.

Algorithm 1: Change-point detector oracle

```
Input: window size $w \in \mathbb{N}$, detection threshold $c \in \mathbb{R}^+$,
       and logged data $\mathcal{D}$
for $t \leftarrow w$ to $T - w + 1$ do
    $\mu_t^- \leftarrow w^{-1} \sum_{i=t-w}^{t-1} r_i$
    $\mu_t^+ \leftarrow w^{-1} \sum_{i=t}^{t+w-1} r_i$
end
Initialize candidates $C \leftarrow \{t : |\mu_t^- - \mu_t^+| \geq c\}$
while $C \neq \emptyset$ do
    Find change-point $\hat{\tau} \leftarrow \arg\max_{t \in C}\{|\mu_t^- - \mu_t^+|\}$
    $C \leftarrow C \setminus [\hat{\tau} - 2w, \hat{\tau} + 2w]$
end
Order change-points $1 < \hat{\tau}_1 < \ldots < \hat{\tau}_{S'-1} < T = \hat{\tau}_{S'}$
for $t \leftarrow 1$ to $T$ do
    $\hat{z}_t \leftarrow i$ such that $t \in [\hat{\tau}_{i-1}, \hat{\tau}_i - 1]$
end
```

Theorem 1 implies that the oracle O can correctly (without false positives) detect change-points within a window w with high probability. Note that both w and c in Theorem 1 depend on Δ, which may not be exactly known. A lower bound on Δ, which we denote by $\tilde{\Delta}$, is sufficient and more likely to be known.

4.2 Graphical Model

Another natural way of partitioning the data is via a latent variable model. In this work, we specifically model the temporal evolution of $z_{1:T}$ with a HMM over $\mathcal{Z}$ (Baum and Petrie, 1966). Let $\Phi = [\Phi_{i,j}]_{i,j=1}^{L}$ be the *transition matrix* with $\Phi_{i,j} = P(z_t = j | z_{t-1} = i)$, and P_0 be the *initial distribution* over $\mathcal{Z}$ with $P_{0,i} = P(z_1 = i)$. The latent states evolve according to $z_1 \sim P_0$, and $z_{t+1} \sim \Phi_{z_t,:}$. Recall from Section 2 that we have joint feature maps of context and action $f(x,a) \in \mathbb{R}^d$. We assume the rewards are sampled according to the conditional distribution $P(\cdot|x,a,z) = \mathcal{N}(\beta_z^T f(x,a), 1)$, where $\beta = (\beta_z)_{z \in \mathcal{Z}}$ are regression weights; though we use Gaussian, any choice of distributions can be incorporated. Let $\mathcal{M} = \{P_0, \Phi, \beta\}$ be the HMM parameters. The HMM can be estimated through expectation-maximization (EM) (Baum and Petrie, 1966).

Oracle O can use the estimated HMM $\hat{\mathcal{M}}$ to predict $\hat{z}_{1:T}$ from Algorithm 2. At each round t, the oracle estimates the latent posterior $Q_t(z) = P(z_t = z \mid x_{1:T}, a_{1:T}, r_{1:T}; \hat{\mathcal{M}})$. using forward-backward recursion (Baum and Petrie, 1966). Then, O predicts $\hat{z}_t = \max_{z \in \mathcal{Z}} Q_t(z)$ at each round t. Though the described HMM oracle is practical, currently no guarantees similar to Assumption 1 can be derived. An analysis similar to Theorem 1 would require parameter recovery guarantees on the HMM, which to our knowledge, do not exist for EM or spectral methods[1] (Hsu et al., 2008).

[1]Guarantees exist only on the marginal probability of data.

Nevertheless, the HMM oracle has several appealing properties. First, unlike the change-point detector, the HMM can map multiple stationary segments into a single latent state, which potentially reduces the size of the latent space. Second, the learned reward model $\hat{r}_z(x, a) = \hat{\beta}_z^T f(x, a) \simeq \mathbb{E}_{r \sim P_z^r(\cdot|x,a)}[r]$ can be incorporated into more advanced off-policy estimators, e.g., DR, instead of IPS as in (1), which reduces the variance.

```
Algorithm 2: HMM oracle
Input: estimated HMM parameters $\hat{\mathcal{M}} = \{\hat{P}_0, \hat{\Phi}, \hat{\beta}\}$, and
       logged data $\mathcal{D}$
Initialize $A_0(z) \leftarrow \hat{P}_{0,z}, B_T(z) \leftarrow 1$.
for $z \in \mathcal{Z}$ do
    Compute $A_t(z), B_t(z)$ for all $t = 1, \ldots, T$ by
      forward-backward recursion
      $A_t(z) \leftarrow \sum_{z' \in Z} A_{t-1}(z') P(z \mid z'; \hat{\Phi}) P(r_t \mid x_t, a_t, z; \hat{\beta})$
      $B_t(z) \leftarrow$
        $\sum_{z' \in Z} P(z' \mid z; \hat{\Phi}) P(r_{t+1} \mid x_{t+1}, a_{t+1}, z'; \hat{\beta}) B_{t+1}(z')$
end
for $t \leftarrow 1, 2, \ldots, T$ do
    Compute $Q_t(z) \propto A_t(z) B_t(z)$ for all $z \in \mathcal{Z}$ and
    $\hat{z}_t \leftarrow \arg\max_{z \in \mathcal{Z}} Q_t(z)$
end
```

5 Optimization and Deployment

In this section, we introduce a piecewise-stationary off-policy optimization algorithm, which consists of two parts: (i) an offline policy optimization that solves for the latent-space policy $\hat{\Pi} = (\hat{\pi}_z)_{z \in \mathcal{Z}}$, $\hat{\pi}_z = \pi(\cdot|\cdot; \hat{\theta}_z) \in \mathcal{H}$; and (ii) an online sub-policy selection procedure. We also provide both a lower bound on reward of the policy derived from offline optimization, as well as an upper bound on regret of its online deployment.

```
Algorithm 3: Piecewise off-policy learning
Input: number of latent states $L \in \mathbb{N}$, logged data $\mathcal{D}$, and
       oracle $O$
Run $O$ on $\mathcal{D}$ to get latent state estimates $\hat{z}_{1:T} \in \mathcal{Z}^T$
for $z \leftarrow 1$ to $L$ do
    Solve for $\hat{\theta}_z$ in (3)
    Create sub-policy $\hat{\pi}_z$ from $\hat{\theta}_z$
end
```

```
Algorithm 4: Piecewise policy deployment
Input: learned policy $\hat{\Pi} \in \mathcal{H}^{\mathcal{Z}}$ and mixture-of-experts
       algorithm $\mathcal{E}$
Initialize algorithm $\mathcal{E}_1$.
for $t \leftarrow 1$ to $T$ do
    Given $x_t$, choose action $a_t \sim \mathcal{E}_t(x_t, \hat{\Pi})$
    Update $\mathcal{E}_{t+1}$ from $\mathcal{E}_t$ with reward $r_t \sim P_{z_t}^{\mathrm{r}}(\cdot \mid x_t, a_t)$
end
```

5.1 Off-Policy Optimization

Leveraging the fact that logged data are partitioned into L sub-datasets, each corresponds to a particular latent state, and the separable structure of the IPS estimator $\hat{V}(\pi)$, the policy optimization problem can also be broken down into learning the best policy at each individual latent state z, i.e., each component of $\hat{\Pi}$ is learned by solving the optimization $\hat{\pi}_z = \arg\max_{\pi \in \mathcal{H}} \hat{V}_z(\pi)$. If each sub-policy $\hat{\pi}_z = \pi(\cdot|\cdot; \hat{\theta}_z)$ is parameterized by some $\hat{\theta}_z \in \Theta$, where Θ denotes the space of model parameters, then we solve the following for each latent state z:

$$\hat{\theta}_z = \arg\max_{\theta \in \Theta} \sum_{t=1}^{T} \mathbb{1}[\hat{z}_t = z] \cdot \min\left\{M, \frac{\pi(a_t \mid x_t; \theta)}{p_t}\right\} r_t \,. \tag{3}$$

If we assume $\hat{\pi}_z$ is a linear soft categorical policy, its parameters $\hat{\theta}_z$ can be solved for as discussed in Section 2. Otherwise, following prior work (Swaminathan and Joachims, 2015b), we can iteratively solve for each sub-policy using standard off-the-shelf gradient ascent algorithms. Algorithm 3 summarizes the procedures for learning $\hat{\Pi} \in \mathcal{H}^{\mathcal{Z}}$.

For $\hat{\Pi} = \arg\max_{\Pi \in \mathcal{H}^{\mathcal{Z}}} \hat{V}(\Pi)$, we now bound from below the expected reward of $\hat{\Pi}$. The following technical result provides a lower bound on expected reward of the learned policy in terms of any oracle O that satisfies Assumption 1. We merely state the result here and defer its derivation to Appendix A.

Theorem 2. *Let*

$$\hat{\Pi} = \arg\max_{\Pi \in \mathcal{H}^{\mathcal{Z}}} \hat{V}(\Pi), \quad \Pi^* = \arg\max_{\Pi \in \mathcal{H}^{\mathcal{Z}}} V(\Pi)$$

be the optimal latent policies w.r.t. the off-policy estimated value and the true value respectively. Then for any $\delta_1, \delta_2 \in (0, 1]$, we have that

$$V(\hat{\Pi}) \geq V(\Pi^*) - 2M\varepsilon(T, \delta_1/2) - 2M\sqrt{2T \log(4/\delta_2)}$$

holds with probability at least $1 - \delta_1 - \delta_2$.

Theorem 2 states that the reward gap of the learned policy $\hat{\Pi}$ from optimal Π^* can be decomposed into error due to oracle O and randomness of logged data $\mathcal{D}$. It is important to note that the value of a policy V is computed assuming the true latent sequence was known. We relax this assumption in Section 5.2, where the latent state is estimated using only the history of interactions thus far.

In the next result, we derive a lower bound on expected reward of policy $\hat{\Pi}$ learned via Algorithm 3 using change-point detector oracle O described in Algorithm 1.

Corollary 1. *Fix any $\tilde{\Delta} \leq \Delta$ and $\delta_1, \delta_2 \in (0, 1]$. Let oracle O be Algorithm 1 with*

$$w = 8 \log(16T/\delta_1)/\tilde{\Delta}^2, \quad c = \tilde{\Delta}/2\,,$$

and Π^*, $\hat{\Pi}$ be defined as in Theorem 2. Then

$$V(\hat{\Pi}) \geq$$
$$V(\Pi^*) - 16M\left(S \log(16T/\delta_1)/\tilde{\Delta}^2\right) - 2M\sqrt{2T \log(4/\delta_2)}$$

holds with probability at least $1 - \delta_1 - \delta_2$.

Corollary 1 follows from applying Theorem 1 to Theorem 2. This implies that if the estimated latent states $\hat{z}_{1:T}$ are generated by Algorithm 1, and the policy $\hat{\Pi}$ is learned via Algorithm 3, then the difference in the expected rewards of $\hat{\Pi}$ from Π^* is $\tilde{O}(\sqrt{T})$.

5.2 Online Deployment

Recall that our offline optimizer learns a vector of sub-policies $\hat{\Pi} = (\hat{\pi}_z)_{z \in \mathcal{Z}}$, one for each latent state. During online deployment, however, the latent state is still unobserved, and we cannot query an oracle as we did offline. We need an online algorithm that switches between the L learned sub-policies based on past rewards.

Our solution is to treat each sub-policy as an "expert", and select which sub-policy to execute each round via a mixture-of-experts algorithm $\mathcal{E}$. This is because we can treat how well each sub-policy performs on the online data as a surrogate predictor of the unknown latent state. The online deployment algorithm is detailed in Algorithm 4, which takes as input a mixture-of-experts algorithm $\mathcal{E}$. At each round t, actions are sampled as $a_t \sim \mathcal{E}_t(x_t, \hat{\Pi})$, where $\mathcal{E}_t$ depends on the history of rewards so far and context x_t.

To simplify exposition, we introduce shorthand $\mathbb{E}_{z,\pi}[\cdot] = \mathbb{E}_{x,a,r \sim P_z,\pi}[\cdot]$. We also assume initially that the latent sequence in T rounds online is the same $z_{1:T}$ in the offline data; we later give a high-level argument on how to relax this assumption. Let the T-round regret be defined as

$$\mathcal{R}(T; \mathcal{E}, \hat{\Pi}) = \sum_{t=1}^{T} \mathbb{E}_{z_t, \pi^*_{z_t}}[r_t] - \sum_{t=1}^{T} \mathbb{E}_{z_t, \mathcal{E}_t}[r_t]\,.$$

The first term is the optimal policy Π^* acting according to the true latent state, and the second term is our offline-learned policies $\hat{\Pi}$ acting according to $\mathcal{E}$. In this section, we give a brief outline of how to bound the online regret, and defer details to Appendix C.

Recall that S is the number of stationary segments, and $\tau_0 = 1 < \tau_1 < \ldots < \tau_{S-1} < T = \tau_S$ are the change-points. Assuming the latent state is constant over a stationary segment, we first have the following lemma that decomposes the regret $\mathcal{R}(T; \mathcal{E}, \hat{\Pi})$.

Lemma 3. *The regret $\mathcal{R}(T; \mathcal{E}, \hat{\Pi})$ is bounded from above as*

$$\begin{aligned} \mathcal{R}(T; \mathcal{E}, \hat{\Pi}) \leq & \left[\sum_{t=1}^{T} \mathbb{E}_{z_t, \pi^*_{z_t}}[r_t] - \sum_{t=1}^{T} \mathbb{E}_{z_t, \hat{\pi}_{z_t}}[r_t]\right] \\ & + \left[\sum_{s=1}^{S} \max_{z \in \mathcal{Z}} \sum_{t=\tau_{s-1}}^{\tau_s - 1} \mathbb{E}_{z_t, \hat{\pi}_z}[r_t] - \sum_{t=1}^{T} \mathbb{E}_{z_t, \mathcal{E}_t}[r_t]\right]. \end{aligned} \tag{4}$$

The first-term is exactly $(V(\Pi^*) - V(\hat{\Pi}))$ and is bounded by Theorem 2 in our offline analysis, which shows near-optimality of $\hat{\Pi}$. The second term is bounded by the regret of mixture-of-experts algorithm $\mathcal{E}$ over $S - 1$ switches.

Prior work showed an optimal T-round switching regret with $S - 1$ switches of $O(\sqrt{SKT})$ (Luo et al., 2018). One such algorithm that is optimal up to log factors is Exp4.S (Luo et al., 2018). We adapt Exp4.S to stochastic experts in Algorithm 6 of Appendix C. Using this algorithm for $\mathcal{E}$ gives us the following bound on online regret.

Theorem 3. *Let $\hat{\Pi}$ be defined as in Theorem 2 and $\mathcal{E}$ be Exp4.S Algorithm 6. Let $z_{1:T}$ be the same latent states as in offline data $\mathcal{D}$ and S be the number of stationary segments. Then for any $\delta_1, \delta_2 \in (0, 1]$, we have that*

$$\mathcal{R}(T; \mathcal{E}, \hat{\Pi}) \leq$$
$$2M\varepsilon(T, \delta_1/2) + 2M\sqrt{2T \log(4/\delta_2)} + 2\sqrt{STK \log L}$$

holds with probability at least $1 - \delta_1 - \delta_2$.

Deploying our offline-learned policy $\hat{\Pi}$ online yields regret that elegantly decomposes into the expected reward gap of $\hat{\Pi}$ from Π^* from off-policy optimization, and the regret of $\mathcal{E}$ used to switch between sub-policies of $\hat{\Pi}$.

5.3 Policy selection by posterior sampling.

In Section 4.2, we proposed but did not analyze using an HMM estimated on the offline data to learn the latent state partitioning. The same HMM can be used to stochastically sample a latent state from its posterior probability, and play according to the corresponding expert, similar to Bayesian policy reuse for adversarial environments (Rosman et al., 2016). Some guarantees exist for posterior sampling of stationary latent states (Hong et al., 2020), but not for ones that evolve according to an unknown HMM. The posterior sampling algorithm is shown in Algorithm 5, and can be incorporated in place of Exp4.S as $\mathcal{E}$ if an HMM was estimated offline. While regret guarantees do not exist as for Exp4.S, such posterior sampling algorithms typically have much better empirical performance.

Algorithm 5: HMM posterior sampling

```
Input: vector of experts $\hat{\Pi} = (\hat{\pi}_z)_{z \in \mathcal{Z}}$ with $|\mathcal{Z}| = L$, and
       estimated HMM parameters $\hat{\mathcal{M}} = \{\hat{P}_0, \hat{\Phi}, \hat{\beta}\}$
Initialize $w_1 = \hat{P}_0$.
for $t \leftarrow 1, 2, \dots, T$ do
    Observe $x_t \in \mathcal{X}$, and expert feedback
      $\hat{\pi}_z(\cdot \mid x_t)$, $\forall z \in \mathcal{Z}$
    Choose action $a_t \sim w_t$, where for each $a \in \mathcal{A}$,
      $w_t(a) = \sum_{z \in \mathcal{Z}} Q_t(z) \hat{\pi}_z(a \mid x_t)$
    Observe $r_t$. Update the latent distribution, $\forall z \in \mathcal{Z}$,
      $Q_{t+1}(z) \propto \sum_{z' \in \mathcal{Z}} Q_t(z') P(r_t \mid x_t, a_t, z'; \hat{\beta}) P(z \mid z'; \hat{\Phi})$
end
```

5.4 Extending to different latent sequences.

In Theorem 3, we bounded the online regret of our algorithm with respect to latent state sequence $z_{1:T}$ in Lemma 3. In particular, the first term of the regret is given by $(V(\Pi_*) - V(\hat{\Pi}))$, and is computed with respect to $z_{1:T}$.

Now, consider the case that the online data has a different latent sequence $z'_{1:T}$. For stationary policy $\pi \in \mathcal{H}$, its value for round t is given by $V'_t(\pi) = \mathbb{E}_{x,a,r \sim P_{z'_t}, \pi}[r]$. For policy $\Pi \in \mathcal{H}^{\mathcal{Z}}$, we define its value as $V'(\Pi) = \sum_{z \in \mathcal{Z}} V'_z(\pi_z)$, where $V'_z(\pi_z) = \sum_{t=1}^{T} \mathbb{1}[z'_t = z] V'_t(\pi_z)$. We want to characterize how the reward gap $(V'(\Pi_*) - V'(\hat{\Pi}))$ changes when computed with respect to $z'_{1:T}$.

For $z \in \mathcal{Z}$, let T_z be the number of occurrences of z in $z_{1:T}$ and T'_z in $z'_{1:T}$. We can bound the difference in reward gap of $\hat{\Pi}$ between the two latent sequences as,

$$\begin{aligned}
&\left(V'(\Pi^*) - V'(\hat{\Pi})\right) - \left(V(\Pi^*) - V(\hat{\Pi})\right) \\
&\quad \le \sum_{z \in Z} \left(V'_z(\Pi^*) - V'_z(\hat{\Pi})\right) - \left(V_z(\Pi^*) - V_z(\hat{\Pi})\right) \\
&\quad \le \sqrt{L \textstyle\sum_{z \in \mathcal{Z}} (T'_z - T_z)^2}\,.
\end{aligned}$$

This additional error can be naively added to the regret bound in Theorem 3.

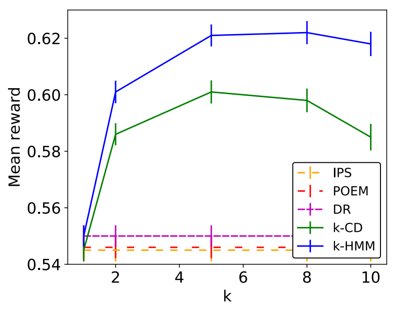
6 Experiments

In this section, we evaluate our approach on synthetic and real-world datasets, and show that it outperforms learning a single stationary policy. We compare the following methods: (i) **IPS**: single policy trained on IPS objective; (ii) **DR**: Single policy trained on DR objective, with reward model $\hat{r}(x, a) = \hat{\beta}^T f(x, a)$ fit using least squares; (iii) **POEM**: single policy trained on counterfactual risk minimization (CRM) objective, which adds an empirical covariance regularizer to the objective in Section 2 (Swaminathan and Joachims, 2015b); (iv) k**-CD**: k sub-policies trained using our method and change-point detector oracle (Algorithm 1), deployed via Exp4.S (Algorithm 6); (v) k**-HMM**: k sub-policies trained using our method and HMM oracle (Algorithm 2), deployed using posterior sampling (Algorithm 5). The first three are baselines in stationary off-policy optimization, and the last two are our approach. In our approach, k is a tunable parameter that estimates the unknown number of latent states L. Note that in k-CD, we controlled for the number of latent states by performing k-means clustering on detected stationary segments by the value of the logging policy over each segment.

6.1 Synthetic Dataset

First, we create a synthetic non-stationary multi-armed bandit without context, with $\mathcal{A} = [5]$ and $\mathcal{Z} = [5]$. In this case, the joint feature vector $f(x, a) \in \{0, 1\}^{|\mathcal{A}|}$ for action a is its indicator. Mean rewards are sampled uniformly at random $\mu(a, z) \sim \mathsf{Uniform}(0, 1)$ for each $a \in \mathcal{A}, z \in \mathcal{Z}$. Rewards are drawn i.i.d. as $r \sim \mathcal{N}(\cdot \mid \mu(a, z), \sigma^2)$ with $\sigma = 0.5$. In constructing $z_{1:T}$, we had $z_1 = 1$, then had each latent state last $10,000$ runs before being incremented to the next one. After round $50,000$ we decremented the latent state instead. So we construct a piecewise-stationary environment with $T = 100,000$ and changes every $10,000$ rounds. In collecting logged data, we want the logging policy π_0 to perform well on average over all latent states, which often happens in practice. We constructed π_0 to act as $\pi_0(a) \propto \exp(\tilde{\mu}(a))$, where $\tilde{\mu}(a) = |\mathcal{Z}|^{-1} \sum_{z \in \mathcal{Z}} \mu(a, z) + \epsilon, \epsilon \sim \mathcal{N}(0, 0.1)$ is a perturbed mean reward for action a.

To evaluate the methods listed, we performed online deployment of the policies learned on the logged data via each method on 10 independent runs of the same piecewise-stationary environment. Here the latent sequence $z_{1:T}$ is the same in logged data and evaluation, which is the case we analyzed. We relax this assumption in the next experiment. For the change-point detector of k-CD, we set $w = 4,000$ and solve for $c = \sqrt{2 \log(8T^2)/w}$ so that the inequality in Theorem 1 is satisfied. Figure 2 shows the expected reward of all learned policies. Both of our approaches, k-CD and k-HMM, significantly outperformed learning a stationary policy, with k-HMM performing better. This is likely because k-HMM acts stochastically according to the learned HMM, whereas k-CD, which uses adversarial Exp4.S, acts too conservatively. Note that since L is not known in practice, we must estimate the number of



Method	Reward
IPS	0.545
DR	0.550
POEM	0.546
Ours:	
k-CD	0.601
k-HMM	**0.621**

Figure 2: Mean reward and standard deviation in the synthetic dataset. The results are averaged over 10 runs. The table shows results for $k = 5$.

latent states. This results in a bias-variance trade-off, where underestimating $k < L$ leads to under-partitioned data and biased sub-policies, and overestimating $k > L$ results in over-partitioned data and sub-policies with high variance. This is evident in Figure 2.

6.2 Yahoo! Dataset

We also experiment with the Yahoo! clickstream dataset (Li et al., 2010). The dataset consists of offline interactions: in each interaction, a document was uniformly sampled from a pool of documents to show to a user, and whether the document was clicked by the user was logged. In prior work, the average clic$k$-through-rates (CTR) of documents across users was empirically verified to change over time (Cao et al., 2019; Wu et al., 2018)

We construct a logged dataset as follows. To reduce the size of the data, we chose a 6-day horizon, and randomly subsampled one interaction per second uniformly from the data. For each sampled interaction, we chose a random subset of 10 documents uniformly sampled without replacement from the pool to ensure that each round had the same number of actions. The context for each interaction consists of the concatenation of the 10 sampled document vectors, and the action and reward correspond to the randomly sampled document and whether it was clicked in the original dataset. Hence, we created a logged dataset with horizon $T = 86,400 \times 6 = 518,400$, and number of actions $K = 10$. It is important to note that the CTR for each document likely non-stationary according to a mixture of smooth and sudden changes. We use this experiment to show that our algorithms perform well even when piecewise-stationary may not hold.

Given this logged data, we can learn policies offline using the methods described in the beginning of Section 6. Because our switching strategies depend on past interactions, offline evaluation of such policies using the logged data requires rejection sampling, which can be sample inefficient to do (Li et al., 2011). We remedy this by instead constructing a semi-synthetic piecewise-stationary bandit environment. To sample the reward for choosing a document to recommend in a particular round, we compute the mean CTR for the chosen document over a half-day window around that round, and sampled Bernoulli rewards from the computed mean. The half-day window is to model that the reward for a document is piecewise-stationary.

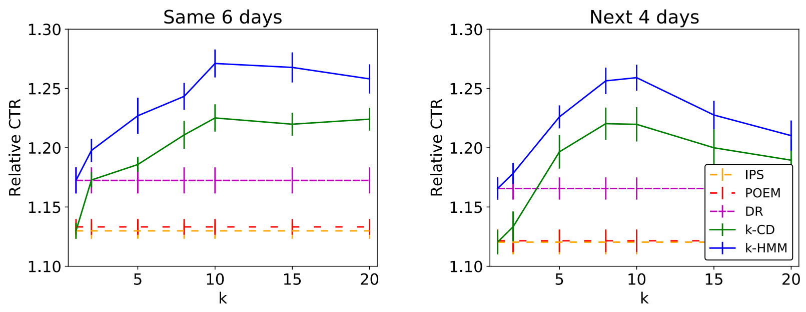
We evaluate our learned policies from the logged data in online deployment in two different bandit experiments. In the first one, we sub-sampled interactions from the same 6-day horizon, one per second, to make the rounds in the episode. This approximately ensures that the underlying latent sequence in the episode is the same as that in the logged data, which is the special case that we analyze. In the second experiment, the rounds were sub-sampled from the next 4 days of data, which potentially have a dramatically different latent sequence. In Figure 3, we reported relative CTR for all the methods over 10 runs. We also plot the relative CTR of k-CD and k-HMM methods as a function of the estimated number of latent states, k. Both of our approaches performed the best, with k-HMM better due to learning a full environment model. Our methods outperformed stationary baselines by up to 10%. The results show that even in situations with non-obvious latent state structure, our approach still improves on methods that ignore latent states.

7 Related Work

Our work considers off-policy learning in a non-stationary bandit setting. Both domains are individually well-explored in prior literature.

Off-policy Learning. A plethora of works is devoted to building counterfactual estimators for evaluating policies. The unbiased IPS estimator has optimal theoretical guarantees when the logging policy is known or estimated well (Strehl et al., 2010; Xie et al., 2019). Various techniques have been employed to reduce the variance of IPS estimators as importance weight clipping (Ionides, 2008; Bottou et al., 2013), or learning a model of reward feedback, to improve the MSE of the estimator (Dudik et al., 2011; Farajtabar et al., 2018; Wang et al., 2017; Chen et al., 2019b). Off-policy estimators can be directly applied to learning policies by optimizing the estimated value. Recent work in off-policy optimization additionally regularizes the estimated value with its empirical standard deviation (Swaminathan and Joachims, 2015b), or uses self-normalization as control variates (Swaminathan and Joachims, 2015a). There is also orthogonal work in handling combinatorial actions (Swaminathan et al., 2016; Li et al., 2018; Chen et al., 2019a).

Non-stationary Bandits. The problem of non-stationary rewards is well-studied in bandit literature (Beshes et al., 2014; Garivier and Moulines, 2008). First works adapted



Method	same 6 days	next 4 days
IPS	1.13 ± 0.006	1.12 ± 0.010
DR	1.16 ± 0.011	1.17 ± 0.009
POEM	1.13 ± 0.008	1.13 ± 0.009
Ours:		
k-CD	1.21 ± 0.012	1.21 ± 0.010
k-HMM	$\mathbf{1.25 \pm 0.011}$	$\mathbf{1.24 \pm 0.011}$

Figure 3: Mean relative CTR and standard deviation in the Yahoo! dataset. The results are averaged over 10 runs. The table shows results for $k = 10$.

to changes passively by weighting rewards, either by exponential discounting (Kocsis and Szepesvári, 2006) or by considering recent rewards in a sliding window (Garivier and Moulines, 2008). In the adversarial setting (Auer et al., 2002; Auer, 2002), adaptation can be achieved by bounding the weights of experts from below. These algorithms have state-of-the-art switching regret, which we leverage in the online component of our algorithm. Recent works in piecewise-stationary bandits explored the idea of monitoring reward changes with a change-point detector. The detector examines differences in their distributions (Liu et al., 2018) or empirical means (Cao et al., 2019). Such algorithms have state-of-the-art theoretical and empirical performance, and can be extended with similar guarantees to the contextual case (Luo et al., 2018; Wu et al., 2018). However, off-policy learning in non-stationary bandits focused solely on evaluating a fixed target policy. They utilize time-series forecasting of future values (Thomas et al., 2017) or passively reweigh past observations (Jagerman et al., 2019). There is also related work in offline evaluation of history-dependent policies in stationary environments (Li et al., 2011; Dudik et al., 2012). We are the first to provide a comprehensive method for both off-policy optimization and online policy selection in piecewise-stationary environments.

8 Conclusions

In this work, we take first steps for off-policy optimization in piecewise-stationary environments. We propose algorithms that partition the offline dataset by latent state, and optimize latent sub-policies conditioned on the partitions. We provide two techniques to partition the data: change-point detection and HMM. We prove high-probability bounds on both the quality of off-policy optimized sub-policies and regret during online deployment. Finally, we empirically validate our approach in synthetic and real-world data. We believe our work is a first-step in general off-policy optimization under non-stationarity. Our current approach uses simple oracles to model the logged data; we propose using a change-point detector or HMM, but do not provide guarantees on HMMs due to lack of existing guarantees in inference. Future work can involve leveraging much richer latent variable models. In addition, and additional avenue for research can involve handling smooth changes in the logged data.

References

Yasin Abbasi-yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *NeurIPS*, 2011.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 2002.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. In *SIAM journal on computing*, 2002.

Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 1966.

Omar Beshes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *NIPS*, 2014.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 2013.

Yang Cao, Zheng Wen, Branislav Kveton, and Yao Xie. Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit. *AISTATS*, 2019.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a REINFORCE recommender system. *WSDM*, 2019a.

Minmin Chen, Ramki Gummadi, Chris Harris, and Dale Schuurmans. Surrogate objectives for batch policy optimization in one-step decision making. *NIPS*, 2019b.

Miroslav Dudik, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *ICML*, 2011.

Miroslav Dudik, Dumitru Erhan, John Langford, and Lihong Li. Sample-efficient nonstationary policy evaluation for contextual bandits. *UAI*, 2012.

Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghamvamzadeh. More robust doubly robust off-policy evaluation. *ICML*, 2018.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *International Conference on Algorithmic Learning Theory*, 2008.

Cédric Hartland, Nicolas Baskiotis, Sylvain Gelly, Michèle Sebag, and Olivier Teytaud. Change point detection and meta-bandits for online learning in dynamic environments. In *CAp*, 2007a.

Cédric Hartland, Nicolas Baskiotis, Sylvain Gelly, Michèle Sebag, and Olivier Teytaud. Change point detection and meta-bandits for online learning in dynamic environments. *CAp*, 2007b.

Joey Hong, Branislav Kveton, Manzil Zaheer, Yinlam Chow, Amr Ahmed, and Craig Boutilier. Latent bandits revisited. In *NeurIPS*, 2020.

D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 1952.

Daniel J. Hsu, Sham M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *CoRR*, abs/0811.4413, 2008.

Edward L Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 2008.

Rolf Jagerman, Ilya Markov, and Maarten de Rijke. When people change their mind: Off-policy evaluation in non-stationary recommendation environments. *WSDM*, 2019.

Levente Kocsis and Csaba Szepesvári. Discounted ucb. *In 2nd PASCAL Challenges Workshop*, 2006.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. *NeurIPS*, 2008.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2019. doi: 10.1017/9781108571401.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *WWW*, 2010.

Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual bandit-based news article recommendation algorithms. *WSDM*, 2011.

Shuai Li, Yasin Abbasi-Yadkori, Branislav Kveton, S. Muthukrishnan, Vishwa Vinay, and Zheng Wen. Offline evaluation of ranking policies with click models. *KDD*, 2018.

Fang Liu, Joohyun Lee, and Ness B. Shroff. A change-detection based framework for piecewise-stationary multi-armed bandit problem. *AAAI*, 2018.

Haipeng Luo, Alekh Agarwal, and John Langford. Efficient contextual bandits in non-stationary worlds. *COLT*, 2018.

Benjamin Rosman, Majd Hawasly, and Subramanian Ramamoorthy. Bayesian policy reuse. *Machine Learning*, 2016.

Alexander L. Strehl, John Langford, Lihong Li, and Sham M. Kakade. Learning from logged implicit exploration data. *NIPS*, 2010.

Adith Swaminathan and Thorsten Joachims. The self-normalized estimator for counterfactual learning. *NIPS*, 2015a.

Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. *ICML*, 2015b.

Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudik, John Langford, Damien Jose, and Imed Zitouni. Off-policy evaluation for slate recommendation. *NIPS*, 2016.

Philip S. Thomas, Georgios Theocharous, Mohammad Ghavamzadeh, Ishan Durugkar, and Emma Brunskill. Predictive off-policy policy evaluation for nonstationary decision problems, with applications to digital marketing. *AAAI*, 2017.

Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudik. Optimal and adaptive off-policy evaluation in contextual bandits. *ICML*, 2017.

Qingyun Wu, Naveen Iyer, and Hongning Wang. Learning contextual bandits in a non-stationary environment. *SIGIR*, 2018.

Yuan Xie, Boyi Liu, Qiang Liu, Zhaoran Wang, Yuan Zhou, and Jian Peng. Off-policy evaluation and learning from logged bandit feedback: Error reduction via surrogate policy. *ICLR*, 2019.

Jia Yuan Yu and Shie Mannor. Piecewise-stationary bandit problems with side observations. In *International Conference on Machine Learning*, 2009.