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 the current context 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).

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).

Contextual bandit algorithms are either *on-policy*, where the agent learns online from real-world interactions (Langford and Zhang, 2008; Abbasi-yadkori et al., 2011), or *off-policy*, where the learning process uses offline logged data collected by other 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 catastrophic outcomes.

Most existing contextual bandit algorithms assume that rewards are generated by a stationary distribution. While this is a valid assumption in simpler problems, where the user intent is 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., 2007). We focus on the latter case, known as the *piecewise-stationary* environment (Hartland et al., 2007; Garivier and Moulines, 2008), which is applicable to many event-sensitive decision-making problems.

Non-stationary bandits (Auer et al., 2002; Luo et al., 2018), and more specifically piecewise-stationary bandits (Hartland et al., 2007; Garivier and Moulines, 2008; Yu and Mannor, 2009), have been studied extensively in the on-policy setting. The prior work in non-stationary off-policy learning only considered policy evaluation, where the evolution of contexts is modeled using time series (Thomas et al., 2017) or by weighting past observations (Jagerman et al., 2019). Neither of these works considered 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 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. We address two novel challenges. First, we are the first to consider the bias in off-policy estimation due to an unknown latent state. Second, it is nontrivial to deploy a non-stationary policy learned offline. We are the first to propose a framework for learning the components of a switching policy offline, and then augment them 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 $P^{\mathsf{r}}(\cdot \mid x_t, a_t)$.

Let $\mathcal{H} = \{\pi : \mathcal{X} \to \Delta^{K-1}\}$ be the set of *stochastic stationary policies*, 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 $x \sim P^{\mathsf{x}}, a \sim \pi(\cdot \mid x)$, and $r \sim P^{\mathsf{r}}(\cdot \mid x, a)$. We define

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

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

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

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

In off-policy learning, actions are chosen by a known stationary logging policy $\pi_0 \in \mathcal{H}$. Logged data are 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

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

where M is a tunable *clipping parameter*. When $M = \infty$, the IPS estimator is unbiased, that is $\mathbb{E}[\hat{V}(\pi)] = V(\pi)$. 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). There are methods to design the clipping weight to optimize such trade-offs (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 $\hat{r}(x, a) \simeq \mathbb{E}_{r\sim P^{\mathsf{r}}(\cdot|x,a)}\left[r\right]$, where $\simeq$ denotes an approximation by fitting on $\mathcal{D}$.

2.2 Off-Policy Optimization

Our goal is to learn a policy with the highest expected reward, $\pi^* = \arg\max_{\pi\in\mathcal{H}} V(\pi)$. One popular solution is to maximize the 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 of $\mathcal{X}$ being finite, $f(x, a)$ can be an indicator vector for each pair (x, a), and learning of $\hat{\pi}$ 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 state* $z \in \mathcal{Z}$, where $\mathcal{Z} = [L]$ is the set of L latent states.

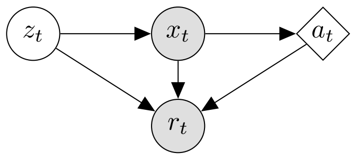


Figure 1: Graphical model for latent state z_t, context x_t, action a_t, and reward r_t.

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)*. (Barto and Sutton, 2018). 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. So, instead of policies that only act on observed context, we should consider policies that also act according to the latent state. Therefore, we define a new class of policies $\mathcal{H}^{\mathcal{Z}}$, whose members are tuples of policies $\Pi = (\pi_z)_{z \in \mathcal{Z}}$, with one stationary policy $\pi_z \in \mathcal{H}$ for each latent state z. The value of Π is

$$\begin{aligned} V(\Pi) &= \sum_{z \in \mathcal{Z}} V_z(\pi_z)\,, \\ V_z(\pi_z) &= \sum_{t=1}^T \mathbb{1}[z_t = z]\, V_t(\pi_z)\,, \end{aligned} \tag{1}$$

where the latter is the value of π_z on the subset of logged data with latent state z. Note that calculating $V(\Pi)$ requires knowing $z_{1:T}$. Therefore, $V(\Pi)$ is hard to compute in practice, but can still be used to reason about performance.

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., 2007; Garivier and Moulines, 2008). In this work, we focus on the latter environment. We denote by S the number of stationary segments in $z_{1:T}$, where the latent state is constant over a segment. We assume that $S \geq L$, as multiple segments can map to the same latent state, and that S is small. We denote the change-points by

$$\tau_0 = 1 < \tau_1 < \ldots < \tau_{S-1} < T = \tau_S\,, \tag{2}$$

where we let $\tau_0 = 1, \tau_S = T$ to simplify exposition.

4 Off-Policy Evaluation

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

$$\begin{aligned} \hat{V}(\Pi) &= \sum_{z \in \mathcal{Z}} \hat{V}_z(\pi_z)\,, \\ \hat{V}_z(\pi_z) &= \sum_{t=1}^T \mathbb{1}[\hat{z}_t = z] \cdot \min\left\{M, \frac{\pi_z(a_t \mid x_t)}{p_t}\right\} r_t\,, \end{aligned} \tag{3}$$

where $\hat{V}_z(\pi_z)$ is the IPS estimator for the logged data with latent state z and $\hat{z}_{1:T}$ is a sequence of latent states predicted by an *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 satisfied,

$$\mathcal{H} = \left\{\pi : \frac{\pi(a \mid x)}{\pi_0(a \mid x)} \leq M \text{ for all } a \in \mathcal{A}, x \in \mathcal{X}\right\}\,, \tag{4}$$

so that the propensity score does not needed to be clipped. The analysis can be straightforwardly extended to a general policy class, and this only adds an extra bias term to the error bound (Ionides, 2008; Li et al., 2018). We omit this to simplify exposition.

If the oracle accurately predicts all the ground-truth latent states, that is $\hat{z}_{1:T} = z_{1:T}$, and $M = \infty$, the IPS estimator $\hat{V}(\pi)$ can be shown to be unbiased.

Lemma 1. *For any $\Pi \in \mathcal{H}^{\mathcal{Z}}$, the IPS estimator $\hat{V}(\Pi)$ in (3) is unbiased when $\hat{z}_{1:T} = z_{1:T}$.*

Proof. From definition of $\hat{V}(\Pi)$ in (3), we have

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

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 address this challenge, in the following we assume that 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.*

Now consider latent state predictions generated by an oracle O that satisfies Assumption 1. Using this oracle, we can provide an upper bound on the estimation error (proof is in Appendix A) of the IPS estimator in (3).

Lemma 2. *For any policy $\Pi \in \mathcal{H}^{\mathcal{Z}}$, its IPS estimate $\hat{V}(\Pi)$ in (3), 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 the IPS estimator can be decomposed into the latent oracle prediction error and a statistical error term that is sublinear in T. In the rest of this section, we introduce two latent prediction oracles. The first one is based on change-point detection, and we show that it satisfies Assumption 1. The second one is based on a *hidden Markov model (HMM)*, for which we do not prove an 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 in (2). In practice, this could over-segment the offline data, if multiple stationary segments can be modeled by the same latent state. However, this assumption is only used in the 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, that is $|V_{\tau_i}(\pi_0) - V_{\tau_i - 1}(\pi_0)| \geq \Delta$.*

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. Then it iteratively selects the round with the highest statistic, declares it a change-point, and removes any nearby rounds from consideration. This 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.

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 $t \leftarrow w + 1$ 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 $C \leftarrow \{t \in [T] : |\mu_t^- - \mu_t^+| \geq c\}$ and
  change-points $\Gamma = \emptyset$
while $C \neq \emptyset$ do
    Find change-point $\hat{\tau} \leftarrow \arg\max_{t \in C}\{|\mu_t^- - \mu_t^+|\}$
    $C \leftarrow C \setminus [\hat{\tau} - 2w, \hat{\tau} + 2w]$
    $\Gamma \leftarrow \Gamma \cup \{\hat{\tau}\}$
end
Order all elements in $\Gamma$ as
  $1 < \hat{\tau}_1 < \ldots < \hat{\tau}_{S'-1} < T = \hat{\tau}_{S'}$, where $S' = |\Gamma| + 1$.
for $t \leftarrow 1$ to $T$ do
    $\hat{z}_t \leftarrow \min\{i \in [S'] : t < \hat{\tau}_i\}$
end
```

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

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

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

Theorem 1 says that the oracle O can correctly detect, without false positives, all 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 known. A lower bound on Δ, which we denote by $\tilde{\Delta}$, would suffice and may be known. We do this to choose c in the experiments in Section 6.

4.2 Hidden Markov Model

Another natural way of partitioning the data is using 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 $\Phi_{i,j} = P(z_t = j \mid z_{t-1} = i)$, and P_0 be the *initial distribution* over $\mathcal{Z}$ with $P_{0,i} = P(z_1 = i)$. The latent states evolve as $z_1 \sim P_0$ and $z_{t+1} \sim \Phi_{z_t,\cdot}$ after that. 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 $P(\cdot \mid x, a, z) = \mathcal{N}(\beta_z^T f(x, a), \sigma^2)$, where $\beta = (\beta_z)_{z \in \mathcal{Z}}$

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 $t = 1, \dots, T$ by
     forward-backward recursion
     $A_t(z) \leftarrow \sum_{z' \in \mathcal{Z}} A_{t-1}(z') P(z \mid z'; \hat{\Phi}) P(r_t \mid x_t, a_t, z; \hat{\beta})$
     $B_t(z) \leftarrow$
       $\sum_{z' \in \mathcal{Z}} P(z' \mid z; \hat{\Phi}) P(r_{t+1} \mid x_{t+1}, a_{t+1}, z'; \hat{\beta}) B_{t+1}(z')$
end
for $t \leftarrow 1, 2, \dots, T$ 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
```

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 $\hat{z}_{1:T} \in [L]^T$
for $z \leftarrow 1$ to $L$ do
    Solve for
    $\hat{\pi}_z = \arg\max_{\pi \in \mathcal{H}} \sum_{t=1}^{T} \mathbb{1}[\hat{z}_t = z] \hat{V}_t(\pi)$
end
Output $\hat{\Pi} = (\hat{\pi}_z)_{z \in \mathcal{Z}}$.
```

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 $r_t \sim P^r_{z_t}(\cdot \mid x_t, a_t)$
end
```

are regression weights. Though we use Gaussians, any distribution could 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 forward and backward probabilities

$$A_t(z) = P(x_{1:t}, a_{1:t}, r_{1:t}, z_t = z; \hat{\mathcal{M}}),$$
$$B_t(z) = P(x_{t+1:T}, a_{t+1:T}, r_{t+1:T} \mid z_t = z; \hat{\mathcal{M}}),$$

and 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 for the HMM, which to our knowledge, do not exist for EM or spectral methods[1] (Hsu et al., 2008). 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 $\hat{r}_z(x, a) = \hat{\beta}_z^T f(x, a) \simeq \mathbb{E}_{r \sim P^r_z(\cdot \mid x, a)}[r]$ can be incorporated into more advanced off-policy estimators, such as the DR estimator in Section 2, and further reduce variance.

5 Optimization and Deployment

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

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

5.1 Off-Policy Optimization

For optimization, we leverage the fact that logged data are partitioned into L sub-datasets, each corresponding to a particular latent state, which gives the IPS estimator $\hat{V}(\pi)$ in (3) a separable structure. In this way, policy optimization can be broken down into learning the best policy at each individual latent state z. Formally, 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 $\hat{\pi}_z = \pi(\cdot \mid \cdot; \hat{\theta}_z)$ 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

$$\hat{\theta}_z = \arg\max_{\theta \in \Theta} \sum_{t=1}^{T} \mathbb{1}[\hat{z}_t = z] \cdot \min\left\{M, \frac{\pi(a_t \mid x_t; \theta)}{p_t}\right\} r_t . \tag{5}$$

If $\hat{\pi}_z$ was a linear soft categorical policy, its parameters $\hat{\theta}_z$ could be found as discussed in Section 2. Otherwise, following prior work (Swaminathan and Joachims, 2015b), we can iteratively solve for each sub-policy using off-the-shelf gradient ascent algorithms. Algorithm 3 summarizes our approach to learning $\hat{\Pi}$.

For $\hat{\Pi} = \arg\max_{\Pi \in \mathcal{H}^{\mathcal{Z}}} \hat{V}(\Pi)$, we now bound from below the expected reward of $\hat{\Pi}$, 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 Π^* decomposes into the error due to oracle O and randomness of logged data $\mathcal{D}$. It is important to note that we assume that the true latent sequence $z_{1:T}$ is known when measuring the performance of a policy. This is evident in (1), where the sub-policy in round t is chosen by z_t. We relax this assumption in Section 5.2, where the latent state is estimated only from past interactions.

Next we derive a lower bound on expected reward of policy $\hat{\Pi}$ learned by Algorithm 3 with change-point detector oracle O 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 combining Theorems 1 and 2. It says that if the estimated latent states $\hat{z}_{1:T}$ are generated by Algorithm 1, and the policy $\hat{\Pi}$ is learned by Algorithm 3, then the difference in the expected rewards of $\hat{\Pi}$ from Π^* is $O(\log T\sqrt{T})$.

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 learned sub-policies based on past rewards.

Our solution is to treat each sub-policy as an "expert", and select which one to execute in each round by a mixture-of-experts algorithm $\mathcal{E}$. This is because the online performance of sub-policies can be treated as a surrogate predictor for the unknown latent state. Our online algorithm is presented in Algorithm 4, and takes a mixture-of-experts algorithm $\mathcal{E}$ as an input. 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 thus 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 online latent sequence is the same as $z_{1:T}$ in 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. The second term is our offline-learned policy $\hat{\Pi}$ 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 change-points are defined as in (2). 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*

$$\begin{aligned}\mathcal{R}(T; \mathcal{E}, \hat{\Pi}) \leq& \left[\sum_{t=1}^{T} \mathbb{E}_{z_t,\pi^*_{z_t}}[r_t] - \sum_{t=1}^{T} \mathbb{E}_{z_t,\hat{\pi}_{z_t}}[r_t]\right] \\ &+ \left[\sum_{s=1}^{S} \max_{z\in\mathcal{Z}} \sum_{t=\tau_{s-1}}^{\tau_s - 1} \mathbb{E}_{z_t,\hat{\pi}_z}[r_t] - \sum_{t=1}^{T} \mathbb{E}_{z_t,\mathcal{E}_t}[r_t]\right]\,.\end{aligned} \tag{6}$$

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 $\hat{\Pi}$ when $z_{1:T}$ are known. The second term is bounded by the regret of mixture-of-experts algorithm $\mathcal{E}$ over $S - 1$ change-points.

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 in Appendix C. Using this algorithm for $\mathcal{E}$ gives us the following regret bound.

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 $1 - \delta_1 - \delta_2$.

Algorithm 5: HMM posterior sampling

```
Input: learned policy $\hat{\Pi} \in \mathcal{H}^{\mathcal{Z}}$ and estimated HMM
       parameters $\hat{\mathcal{M}} = \{\hat{P}_0, \hat{\Phi}, \hat{\beta}\}$
Initialize $w_1 = \hat{P}_0$
for $t \leftarrow 1, 2, \ldots, T$ 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-state posterior 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
```

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

5.3 Policy Selection by Posterior Sampling

In Section 4.2, we learn an HMM offline to identify the latent states. The same HMM can be used to sample a latent state from its posterior, and act according to the corresponding expert, similarly 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. Our posterior sampling algorithm is in Algorithm 5, and works by computing a latent state posterior

$$Q_t(z) = P(z_t = z \mid x_{1:t-1}, a_{1:t-1}, r_{1:t-1}; \hat{\mathcal{M}})\,.$$

Note that this is different from Q_t in Section 4.2, because we only condition on the history. Algorithm 5 can be used as $\mathcal{E}$ in Algorithm 4 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.

5.4 Extension to Different Latent Sequences

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

Now we consider online data with a different latent state sequence $z'_{1:T}$. For stationary policy $\pi \in \mathcal{H}$, we denote its value in round t 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 by $V'(\Pi) = \sum_{z \in \mathcal{Z}} V'_z(\pi_z)$ 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 and T'_z be the number of occurrences of z in $z_{1:T}$ and $z'_{1:T}$, respectively. Note that

$$V_z(\pi_z) = \sum_{t=1}^T \mathbb{1}[z_t = z]\, V_t(\pi_z) = T_z \mathbb{E}_{x,a,r \sim P_z, \pi_z}[r]\,,$$

and similarly for V'_z, as the value of any policy under latent state z is constant. We can bound the difference in reward gap of $\hat{\Pi}$ between the two latent sequences as

$$\begin{aligned}
&\left(V'(\Pi^*) - V'(\hat{\Pi})\right) - \left(V(\Pi^*) - V(\hat{\Pi})\right) \\
&= \sum_{z \in \mathcal{Z}} (V'_z(\pi^*_z) - V'_z(\hat{\pi}_z)) - (V_z(\pi^*_z) - V_z(\hat{\pi}_z)) \\
&\leq \sum_{z \in \mathcal{Z}} |T'_z - T_z|\,.
\end{aligned}$$

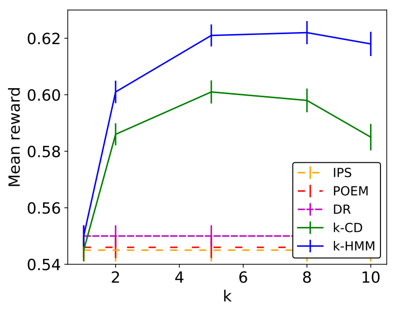
where the first inequality is due to naively bounding from above $\mathbb{E}_{x,a,r \sim P_z, \pi^*_z}[r] - \mathbb{E}_{x,a,r \sim P_z, \hat{\pi}_z}[r] \leq 1$, and the second bounds the ℓ_1 with ℓ_2-norm. This additional error can be added to the regret bound in Theorem 3.

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**: a single policy trained on the IPS objective; (ii) **DR**: a single policy trained on the DR objective, with reward model $\hat{r}(x, a) = \hat{\beta}^T f(x, a)$ fit using least squares; (iii) **POEM**: a single policy trained on the 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 with a change-point detector (Algorithm 1), deployed using Exp4.S (Algorithm 6 of Appendix C); (v) k-**HMM**: k sub-policies trained using our method with an HMM (Algorithm 2), deployed using posterior sampling (Algorithm 5). The first three methods 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. In k-CD, we control the number of latent states by k-means clustering on detected stationary segments. Specifically, we compute the value of the logging policy across each stationary segment, and cluster segments by their value into k latent states.

6.1 Synthetic Dataset

The first problem is a synthetic non-stationary bandit without context, with $\mathcal{A} = [5]$ and $\mathcal{Z} = [5]$, i.e. $L = 5$. The



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

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

mean rewards of actions are sampled uniformly at random as $\mu(a, z) \sim \mathsf{Uniform}(0, 1)$ for each $a \in \mathcal{A}, z \in \mathcal{Z}$. The rewards are drawn i.i.d. as $r \sim \mathcal{N}(\cdot \mid \mu(a, z), \sigma^2)$ with $\sigma = 0.5$. The horizon is $T = 100,000$ rounds. The latent state sequence $z_{1:T}$ is generated as follows. We set $z_1 = 1$. Then, every $10,000$ rounds, the latent state is incremented by one. After round $50,000$, the latent state is decremented by one every $10,000$ rounds. This is a piecewise-stationary environment with changes every $10,000$ rounds. Since this problem is non-contextual, the feature vector $f(x, a) \in \{0, 1\}^{|\mathcal{A}|}$ for action a is its indicator. The logging policy π_0 is designed to perform well on average over all latent states, which often happens in practice. We define it as $\pi_0(a) \propto \exp(\tilde{\mu}(a))$, where $\tilde{\mu}(a) = |\mathcal{Z}|^{-1} \sum_{z \in \mathcal{Z}} \mu(a, z) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, 0.1)$ is a perturbed mean reward for action a.

The learned policies are evaluated by a simulated online deployment, on the same latent state sequence $z_{1:T}$ as in logged data. This is the setting that we analyze. We relax it in the next experiment. In the change-point detector of k-CD, $w = 4,000$ and $c = \sqrt{2 \log(8T^2)/w}$. This satisfies the condition in Theorem 1 for $\delta = 1/T$. Figure 2 shows expected rewards of all learned policies. Both of our approaches, k-CD and k-HMM, significantly outperform 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. As the number of latent states L is not known in practice, it must be estimated, and we also do that in this experiment. 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, as both result in suboptimal performance compared to choosing $k = L$.

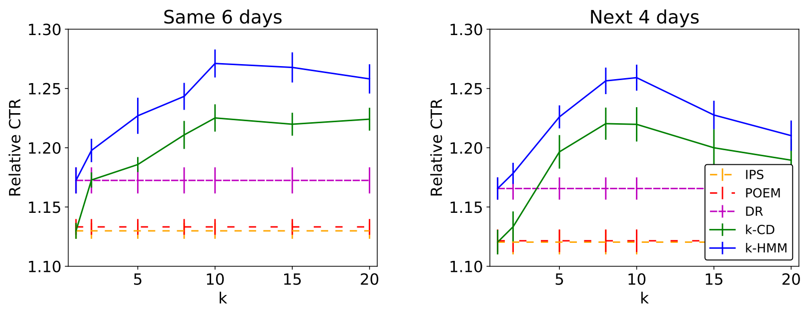
6.2 Yahoo! Dataset

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

We construct our logged dataset as follows. To reduce the size of the data, we choose a 6-day horizon and randomly subsample one interaction per second over that horizon. For each sampled interaction, we choose a random subset of 10 documents that could be shown to the user, to ensure the same number of actions in each round. The context for each interaction is a concatenation of the feature vectors of all 10 sampled documents. The actions are documents and their rewards are indicators of being clicked in the original dataset. The result of this preprocessing is a logged dataset with horizon $T = 6 \times 86,400 = 518,400$ and $K = 10$ actions. It is important to note that the CTR for each document is likely to be non-stationary and change smoothly. Hence, this experiment shows that our algorithms perform well even when our modeling assumptions may not hold.

We learn policies offline using the same methods as in Section 6.1. Because our switching strategies depend on past interactions, offline evaluation of such policies from logged data is challenging. One approach is rejection sampling (Li et al., 2011); but that can be sample inefficient. We remedy this by constructing a semi-synthetic piecewise-stationary bandit environment. In this environment, the CTR of a document in a given round is estimated from a half-day window around that round, and the click is sampled from a Bernoulli distribution with that mean. The half-day window is to model the non-stationarity of clicks.

We evaluate our learned policies in online deployment in two different bandit experiments. In the first experiment, we sub-sample interactions from the same 6-day horizon, one per second. This approximately ensures that the underlying latent sequence is the same as in the logged data, which is the special case that we analyze. In the second experiment, we sub-sample interactions from the next 4 days of data, which potentially have a dramatically different latent state sequence. In Figure 3, we report relative CTRs for all compared methods, averaged 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 perform the best, with k-HMM being better due to learning a full environment model. Our methods outperform stationary baselines by up to 10%. These results show that even in situations with a non-obvious latent state structure, our approach improves over methods that ignore latent states.



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 CTRs and their standard deviations in the Yahoo! dataset. The results are averaged over 10 runs. The table shows results for $k = 10$.

7 Related Work

We study off-policy learning in a non-stationary bandit setting. Both areas have been individually well-explored before.

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 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, and we use them as 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 reward 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 setting (Luo et al., 2018; Wu et al., 2018).

Off-policy learning. Many works in off-policy learning have been 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, such as importance weight clipping (Ionides, 2008; Bottou et al., 2013) or learning a reward model, 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 works in off-policy optimization additionally regularized the estimated value with its empirical standard deviation (Swaminathan and Joachims, 2015b) or used self-normalization as control variates (Swaminathan and Joachims, 2015a). Combinatorial actions, which are common in learning to rank, have been also explored (Swaminathan et al., 2016; Li et al., 2018; Chen et al., 2019a).

Prior work in off-policy learning in non-stationary bandits is sparse, and has focused solely on evaluating a fixed target policy. Such works utilized time-series forecasting of future values (Thomas et al., 2017) or passively reweighed past observations (Jagerman et al., 2019). There are also related works 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 non-stationary environments.

8 Conclusions

In this work, we take first steps towards off-policy optimization in non-stationary environments. Our algorithms partition the offline logged data by latent state, and optimize latent sub-policies conditioned on the partitions. We propose two techniques to partition the data: change-point detection and HMM. We prove high-probability bounds on the quality of off-policy optimized sub-policies and their regret during online deployment. Finally, we empirically validate our approach in synthetic and real-world data.

We believe that our work is the first step in general off-policy optimization under non-stationarity. Our current approach uses simple non-stationary models of 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. Good directions for future work are better models of non-stationarity, which could potentially handle 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.

Andrew Barto and Richard S. Sutton. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 2018.

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. *CAp*, 2007.

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.