On Adaptive Estimation for Dynamic Bernoulli Bandits

Xue Lu
Department of Mathematics
Imperial College London
London, SW7 2AZ, UK

Niall Adams
Department of Mathematics
Imperial College London
London, SW7 2AZ, UK

Nikolas Kantas
Department of Mathematics
Imperial College London
London, SW7 2AZ, UK

Abstract

The multi-armed bandit (MAB) problem is a classic example of the exploration-exploitation dilemma. It is concerned with maximising the total rewards for a gambler by sequentially pulling an arm from a multi-armed slot machine where each arm is associated with a reward distribution. In static MABs, the reward distributions do not change over time, while in dynamic MABs, each arm’s reward distribution can change, and the optimal arm can switch over time. Motivated by many real applications where rewards are binary counts, we focus on dynamic Bernoulli bandits. Standard methods like $\epsilon$-Greedy and Upper Confidence Bound (UCB), which rely on the sample mean estimator, often fail to track the changes in underlying reward for dynamic problems. In this paper, we overcome the shortcoming of slow response to change by deploying adaptive estimation in the standard methods and propose a new family of algorithms, which are adaptive versions of $\epsilon$-Greedy, UCB, and Thompson sampling. These new methods are simple and easy to implement. Moreover, they do not require any prior knowledge about the data, which is important for real applications. We examine the new algorithms numerically in different scenarios and find out that the results show solid improvements of our algorithms in dynamic environments.

Keywords: Dynamic Multi-armed Bandit, Bernoulli Bandits, Adaptive Estimation, UCB, Thompson Sampling

1. Introduction

The multi-armed bandit (MAB) problem is a classic decision problem where one needs to balance acquiring new knowledge with optimising the choices based on current knowledge, a dilemma commonly referred to as the exploration-exploitation trade-off. The problem originally proposed by Robbins (1952) aims to sequentially make selections among a (finite) set of arms, $A$, and maximise the total reward obtained through selections during a (possibly infinite) time horizon $T$. The MAB framework is natural to model many real-world
problems. It was originally motivated by the design of clinical trials (Thompson, 1933; see also Press, 2009, and Villar et al., 2015, for some recent developments). Other applications include online advertising (Li et al., 2010; Scott, 2015), adaptive routing (Awerbuch and Kleinberg, 2008), and financial portfolio design (Brochu et al., 2011; Shen et al., 2015). In stochastic MABs, each arm $a \in \mathcal{A}$ is characterised by an unknown reward distribution. The Bernoulli distribution is a very natural choice that appears very often in the literature, because in many real applications, the rewards can be represented by binary counts. For example, in clinical trials, we obtain a reward 1 for a successful treatment, and a reward 0 otherwise (Villar et al., 2015); in online advertising, counts of clicks are often used to measure success (Scott, 2010).

Formally, the MAB problem may be stated as follows: for discrete times $t = 1, \ldots, T$, the decision maker selects one arm $a_t$ from $\mathcal{A}$ and receives a reward $Y_t(a_t)$. The goal is to optimise the arm selection sequence and maximise the total expected reward $\sum_{t=1}^{T} \mathbb{E}[Y_t(a_t)]$, or equivalently, minimise the total regret:

$$R_T = \sum_{t=1}^{T} \mathbb{E}[Y_t(a_t^*)] - \sum_{t=1}^{T} \mathbb{E}[Y_t(a_t)],$$

(1)

where $a_t^*$ is the optimal arm at time $t$. The total regret can be interpreted as the difference between the total expected reward obtained by playing an optimal strategy (selecting the optimal arm at every step) and that obtained by the algorithm. For notational convenience, we let $\mu_t(a), a \in \mathcal{A}$, denote the expected reward of arm $a$ at time $t$, i.e., $\mu_t(a) = \mathbb{E}[Y_t(a)]$. In rest of this paper, we will also use the notations like $Y_t$ and $\mu_t$ when we introduce the methods/models that can be applied separately to different arms.

The classic MAB problem assumes the reward distribution structure does not change over time. That is to say, in this case, the optimal arm is the same for all $t$. A MAB problem with static reward distributions is also known as the stationary, or static MAB problem in the literature (e.g., Garivier and Moulines, 2011; Slivkins and Upfal, 2008). A dynamic MAB, where changes are allowed in the underlying reward distributions, is more realistic in real-world applications such as online advertising. An agent always seeks the best web position (that is, the placement of the advertisement on a webpage), and/or advertisement content, to maximise the probability of obtaining clicks. However, due to inherent changes in marketplace, the optimal choice may change over time, and thus the assumption of static reward distributions is not adequate in this example.

Two main types of change have been studied in the literature of dynamic MAB: abrupt changes (Garivier and Moulines, 2011; Yu and Mannor, 2009), and drifting (Granmo and Berg, 2010; Gupta et al., 2011; Slivkins and Upfal, 2008). For abrupt changes, the expected reward of an arm remains constant for a some period and changes suddenly at possibly unknown time instants (Garivier and Moulines, 2011). The study of drifting dynamic bandits follows the seminal work of Whittle (1988), in which restless bandit was introduced. In Whittle’s study, the state of an arm can change according to a Markov transition function over time whether it is selected or not. Restless bandits are regarded as intractable, i.e., it is not possible to derive an optimal strategy even if the transitions are deterministic (Papadimitriou and Tsitsiklis, 1999). In recent studies of drifting dynamic bandits, the
expected reward of an arm is often modelled by a random walk (e.g., Granmo and Berg, 2010; Gupta et al., 2011; Slivkins and Upfal, 2008).

In this work, we look at the problem of dynamic bandits where the expectation of the reward distribution changes over time, focusing on the Bernoulli reward distribution because of its wide relevance in real applications. In addition, we will emphasise on cases where the changes of the reward distribution can really have an effect on the decision making. As an example, for a two-armed Bernoulli bandit, the expected reward of Arm 1 oscillates in \([0, 0.3]\) over time, and the expected reward of Arm 2 oscillates in \([0.8, 0.9]\). The reward distributions for both arms change, but the optimal arm remains the same. We will not regard this example as a dynamic case.

Many algorithms have been proposed in the literature to perform arm selection for MAB. Some of the most popular ones include \(\epsilon\)-Greedy (Watkins, 1989), Upper Confidence Bound (UCB; Auer et al., 2002), and Thompson Sampling (TS; Thompson, 1933). These methods have been extended in various ways to improve performance. For example, Garivier and Cappe (2011) proposed the Kullback-Leibler UCB (KL-UCB) method which satisfies a uniformly better regret bound than UCB. May et al. (2012) introduced the Optimistic Thompson Sampling (OTS) method to boost exploration in TS. Some more extensions will be described in Section 3. Even in their basic forms, all the fore-mentioned approaches can perform well in practice in many situations (e.g., Chapelle and Li, 2011; Kuleshov and Precup, 2014; Vermorel and Mohri, 2005). One thing that these methods have in common is that, they treat all the observations \(Y_1, \cdots, Y_t\) equally when estimating or making inference of \(\mu_t\). Specifically, \(\epsilon\)-Greedy and UCB use sample averages to estimate \(\mu_t\). In static cases, given that \(Y_1, \cdots, Y_t\) are i.i.d, this choice is a sensible one from a theoretical perspective, and one could invoke various asymptotic results in here as justification (e.g., law of large numbers, central limited theorem, Berry Essen inequality etc.). From a practical point of view, when \(\mu_t\) changes significantly with time, it could become a bottleneck in performance. The problem is that, a sample average does not put more weight on more recent data \(Y_t\), which is a direct observation of \(\mu_t\). In this paper we will consider using a different estimator for \(\mu_t\) that is inspired from adaptive estimation (Haykin, 2002) and propose novel modifications of popular MAB algorithms.

1.1 Contributions and Organisation

We propose algorithms that use adaptive forgetting factors (Bodenham and Adams, 2016) in conjunction with the standard MAB methods. This results to a new family of algorithms for dynamic Bernoulli bandits. These algorithms overcome the shortcomings related to using sample averages for estimation of dynamically changing rewards. These algorithms are easy to implement and require very little tuning effort; they are quite robust to tuning parameters and their initialisation does not require assumptions or knowledge on the model structure in advance.

The remainder of this paper is structured as follows: Section 2 briefly summarises some adaptive estimation techniques, focusing on Adaptive Forgetting Factors (AFFs). Section 3 introduces the methodology for arm selection. Section 4 presents a variety of numerical results for different dynamic models and MAB algorithms. We summarise our findings in Section 5.
2. Adaptive Estimation Using Forgetting Factors

Solving the MAB problem involves two main steps: learning the reward distribution of each arm (estimation step), and selecting one arm to play (selection step). The foundation of making a good selection is to correctly and efficiently track the expected reward of the arms, especially in the context of time-evolving reward distributions. Adaptive estimation approaches are useful for this task, as they provide an estimator that follows closer a moving target, here the target is the expected reward (Anagnostopoulos et al., 2012; Bodenham and Adams, 2016). In this section, we introduce how to use an Adaptive Forgetting Factor (AFF) estimator for monitoring a single arm. For the sake of simplicity, when clear we have dropped dependence on arms in the notation.

Assume now that we select one arm all the time until $t$ and receive rewards $Y_1, \ldots, Y_t$. If the reward distribution is static, $Y_1, \ldots, Y_t$ are i.i.d. Therefore, it is natural to estimate the expected reward via the sample mean: $\bar{Y}_t = \frac{1}{t}\sum_{i=1}^{t} Y_i$. This sample mean estimator was widely used in the algorithms designed for the static MAB problem such as $\epsilon$-Greedy and UCB. One problem with this estimator is that it often fails in the case that the reward distribution changes over time. The adaptive filtering literature (Haykin, 2002) provides a generic and practical tool to track a time-evolving data stream, and it has been recently adapted to a variety of streaming machine learning problems (Anagnostopoulos et al., 2012; Bodenham and Adams, 2016). The key idea behind adaptive estimation is to gradually reduce the weight on older data as new data arrives (Haykin, 2002). For example, a fixed forgetting factor estimator employs a discount factor $\lambda$, $\lambda \in [0, 1]$, and takes the form $\hat{Y}_{\lambda,t} = \frac{1}{w_{\lambda,t}} \sum_{i=1}^{t} \lambda^{t-i} Y_i$, where $w_{\lambda,t}$ is a some normalising constant. Bodenham and Adams (2016) illustrated that the fixed forgetting factor estimator has some similarities with the Exponentially Weighted Moving Average (EWMA) scheme (Roberts, 1959) which is a basic approach in the change detection literature (Tsung and Wang, 2010).

In this paper, we will use an adaptive forgetting factor where the magnitude of the forgetting factor $\lambda$ can be adjusted at each time step for better adaptation. One main advantage of an AFF estimator is that it can respond quickly to the changes of a target without requiring any prior knowledge about the process. In addition, by using data-adaptive tuning of $\lambda$, we side-step the problem of setting a key control parameter. Therefore, it is very useful when applied to dynamic MABs where we do not have any knowledge about the dynamics of the reward distribution.

Our AFF formulation follows Bodenham and Adams (2016). We present here only the main methodology. For a data stream $Y_1, \ldots, Y_t$, the adaptive forgetting factor mean (denoted by $\hat{Y}_t$) is defined as follows:

$$\hat{Y}_t = \frac{1}{w_t} \sum_{i=1}^{t} \left( \prod_{p=i}^{t-1} \lambda_p \right) Y_i,$$

where the normalising constant $w_t = \sum_{i=1}^{t} \left( \prod_{p=i}^{t-1} \lambda_p \right)$ is selected to give unbiased estimation when the data $Y_1, \ldots, Y_t$ are i.i.d. For convenience, we set $\prod_{p=i}^{t-1} \lambda_p = 1$. We can update $\hat{Y}_t$
via the following recursive updating equations:

\[
\hat{Y}_t = \frac{m_t}{w_t},
\]

\[
m_t = \lambda_{t-1} m_{t-1} + Y_t,
\]

\[
w_t = \lambda_{t-1} w_{t-1} + 1.
\]

The adaptive forgetting factor \( \lambda_t = (\lambda_1, \lambda_2, \cdots, \lambda_t) \) is a expanding sequence over time, and the forgetting factor \( \lambda_t \) is computed via a single gradient descent step, which is

\[
\lambda_t = \lambda_{t-1} - \eta \Delta(L_t, \lambda_{t-2}),
\]

where \( \eta (\eta \ll 1) \) is the step size, and \( L_t \) is a user determined cost function of the estimator \( \hat{Y}_t \). Here, we choose \( L_t = (\hat{Y}_{t-1} - Y_t)^2 \) for good mean tracking performance, which can be interpreted as the one-step-ahead squared prediction error. Other choices are possible, such as the one-step-ahead negative log-likelihood (Anagnostopoulos et al., 2012), but this will not be pursued here. In addition, \( \Delta(L_t, \lambda_{t-2}) \) is a derivative-like function of \( L_t \) with respect to \( \lambda_{t-2} \) (see Bodenham and Adams, 2016, sect. 4.2.1 for details). Note, the index of \( \lambda \) is \( t-2 \) as only \( \lambda_1, \cdots, \lambda_{t-2} \) are involved in \( L_t \). We require the following recursions to sequentially compute \( \Delta(L_t, \lambda_{t-2}) \):

\[
\dot{m}_t = \lambda_{t-1} \dot{m}_{t-1} + m_{t-1},
\]

\[
\dot{w}_t = \lambda_{t-1} \dot{w}_{t-1} + w_{t-1},
\]

\[
\Delta(L_t, \lambda_{t-2}) = 2(\hat{Y}_{t-1} - Y_t) \left( \frac{\dot{m}_{t-1} - \dot{w}_{t-1} \hat{Y}_{t-1}}{w_{t-1}} \right).
\]

In addition to the mean, we may make use of an adaptive estimate of the variance. The adaptive forgetting factor variance is defined as:

\[
s_t^2 = \frac{1}{v_t} \sum_{i=1}^{t} \left( \prod_{p=1}^{t-1} \lambda_p \right) (Y_t - \hat{Y}_t)^2.
\]

Note here we choose the same adaptive forgetting factor for mean and variance for convenience, though other formulations are possible. One can use a separate adaptive forgetting factor for the variance if needed. Again, \( s_t^2 \) can be computed recursively via the following equations:

\[
s_t^2 = \frac{1}{v_t} \left[ (\lambda_{t-1} v_{t-1}) s_{t-1}^2 + \left( \frac{w_t - 1}{w_t} \right) (\hat{Y}_{t-1} - Y_t)^2 \right],
\]

\[
u_t = \left( \frac{w_t - 1}{w_t} \right)^2 u_{t-1} + \left( \frac{1}{w_t} \right)^2,
\]

\[
v_t = w_t (1 - u_t).
\]

The only tuning parameter in AFF estimation is the step size \( \eta \) used in (6), and its choice may affect the performance of estimation. In Bodenham and Adams (2016), the authors
proved that $\mathbb{E}[\Delta(L_t, \lambda_{t-2})] \sim O(\sigma^2)$ when the data $Y_1, \cdots, Y_t$ are i.i.d with variance $\sigma^2$. That is to say, the forgetting factors, $\lambda_1, \cdots, \lambda_t$, computed via (6) will be forced to be either 0 or 1 if $\sigma^2$ is too large. Therefore, before examining the influence of $\eta$, they scaled $\Delta(L_t, \lambda_{t-2})$ to $\Delta(L_t, \lambda_{t-2})/\sigma^2$ ($\sigma^2$ can be estimated during a burn-in period). However, in this paper, we are only interested in Bernoulli rewards, which means that $\sigma^2$ is less than 1, so it is not essential to devise an elaborate scaling scheme. We apply the AFF estimation in standard MAB algorithms (see Section 3), and examine empirically the influence of $\eta$ on these algorithms in Section 4.2.1

2.1 Dealing with Missing Observations

In the MAB setting, we have at least two arms, and for each arm, we will construct an AFF estimator. However, we can only observe one arm at a time. This means that the estimations and intermediate quantities of an unobserved arm will retain their previous values, that is, if arm $a$ is not observed at time $t$,

$$\hat{Y}_t(a) = \hat{Y}_{t-1}(a),$$
$$s^2_t(a) = s^2_{t-1}(a),$$
$$\lambda_t(a) = \lambda_{t-1}(a),$$
$$m_t(a) = m_{t-1}(a),$$
$$w_t(a) = w_{t-1}(a).$$

Not being able to update estimators sets more challenges in dynamic cases. In static cases, the sample mean estimator will converge quickly to the expected reward with a few observations, and therefore it has little effect if the arm is not observed further. However, in dynamic cases, even if the estimator tracks the expected reward perfectly at a given moment, its precision may deteriorate quickly once it stops getting new observations. Therefore, it is more challenging to balance exploration and exploitation in dynamic cases.

3. Action Selection

Having discussed how to track the expected reward of arms in the previous section, we now move on to methods for the selection step. We will consider three of the most popular methods: $\epsilon$-Greedy (Watkins, 1989), UCB (Auer et al., 2002) and TS (Thompson, 1933). They are easy to implement and computationally efficient. Moreover, they have good performance in numerical evaluations (Chapelle and Li, 2011; Kuleshov and Precup, 2014; Vermorel and Mohri, 2005). Each of these methods uses a different mechanism to balance the exploration-exploitation trade-off. Deploying AFF in these methods, we propose a new family of MAB algorithms for dynamic Bernoulli bandits, and they are denoted with the prefix AFF- to emphasise the use of AFF in estimation. Driven by $\epsilon$-Greedy, UCB and TS, the new algorithms are AFF-$d$-Greedy, AFF-UCB, and AFF-TS/AFF-OTS respectively.

In the literature of dynamic bandits, many approaches attempted to improve the performance in standard methods by choosing an estimator that uses the reward history wisely. Kouliouriotis and Xanthopoulos (2008) applied exponentially-weighted average estimation in $\epsilon$-Greedy. Kocsis and Szepesvari (2006) introduced the discounted UCB method (it was
also called D-UCB in Garivier and Moulines, 2011) which used a fixed discounting factor in estimation. Garivier and Moulines (2011) proposed the Sliding Window UCB (SW-UCB) algorithm where the reward used for estimation is restricted by a window. The Dynamic Thompson Sampling (DTS) algorithm applied a bound on the reward history used for updating the hyperparameters in posterior distribution of $\mu_t$ (Gupta et al., 2011). These sophisticated algorithms require accurate tuning of some input parameters, which relies on knowledge of the model/behaviour of $\mu_t$. For example, computing the window size of SW-UCB, or the discounting factor of D-UCB (Garivier and Moulines, 2011) requires knowing the number of switch points (i.e., time instants that the optimal arm switches). While the idea behind our AFF MAB algorithms is similar, our approaches automate the tuning of the key parameters (i.e., the forgetting factors), and only require little effort to tune the higher level parameter $\eta$ in (6). Moreover, we use the AFF technique to guide the tuning of the key parameter in the DTS algorithm, which will be discussed later in this section. Other approaches for dynamic bandits include UCB$_f$ (Slivkins and Upfal, 2008). DaCosta et al. (2008) used the Page-Hinkley test to restart the UCB algorithm in the application of adaptive operator selection.

In what follows, we discuss each AFF-deployed method separately. We review briefly the basics of each method and refer the reader to the references for more details. In addition, we will continue to use notations like $Y_t$ instead of $Y_t(a)$ when clear. In all the AFF MAB algorithms we propose below, we will use a very short initialisation (or burn-in) period for the initial estimations. Normally, the length of the burn-in period is $|\mathcal{A}|$, that is, selecting each arm once; for the algorithms that requires estimates of variance, we use a longer burn-in period by selecting each arm $M$ times.

### 3.1 $\epsilon$-Greedy

$\epsilon$-Greedy (Watkins, 1989) is the simplest method for the static MAB problem. The expected reward of an arm is estimated by its sample mean, and a fixed parameter $\epsilon \in (0, 1)$ is used for selection. At each time step, with probability $\epsilon$, the algorithm selects an arm uniformly to explore, and with probability $1 - \epsilon$, the arm with the highest estimated reward is picked. $\epsilon$-Greedy is simple and easy to implement, which makes it appealing for dynamic bandits. However, it can have two main issues: first, the sample average is not ideal for tracking the moving reward; second, the parameter $\epsilon$ is the key to balancing the exploration-exploitation dilemma, but it is challenging to tune as an optimal strategy in dynamic environments may require varying $\epsilon$ over time.

In Algorithm 1, we propose the AFF-$d$-Greedy algorithm to overcome the above weaknesses. In the algorithm, we use the AFF mean $\hat{Y}_t$ from (2) to estimate the expected reward. This estimator can respond quickly to changes, that is, for an arm that is frequently observed, it can closely follow the underlying reward; for an arm that is not observed for a long time, the estimator can capture $\mu_t$ quickly once the arm is selected again. At each time step, we first identify the arm with the highest AFF mean; if the absolute difference between this arm’s last two forgetting factors is smaller than $d$, we select it; otherwise, we select an arm from $\mathcal{A}$ uniformly. A threshold $d \in (0, 1)$ is used to balance exploration and exploitation. Tuning $d$ is easier than $\epsilon$ as it is related to the step size $\eta$ used in (6). This
was confirmed in a large number of simulations. For Bernoulli dynamic bandits, we suggest to set $d \approx \eta$.

We use the forgetting factors $\lambda_t$ ($t = 1, 2, \cdots$) in the decision rule as their magnitudes indicate the variability of the data stream. For example, if $\lambda_t$ is close to zero, it can be interpreted as a sudden change occurring at time $t$, and if close to 1, it indicates that the data stream is stable at time $t$. To understand the decision rule better, we illustrate it using two examples.

1. Variable arm example: let us say arm $\hat{a}$ was selected at time $t - 1$, and at time $t$, arm $\hat{a}$ has the highest estimated reward and $|\lambda_{t-1}(\hat{a}) - \lambda_{t-2}(\hat{a})| < d$. By the decision rule, the algorithm will select this arm again. We are interested in two cases: first, both $\lambda_{t-1}(\hat{a})$ and $\lambda_{t-2}(\hat{a})$ are close to 1; second, both $\lambda_{t-1}(\hat{a})$ and $\lambda_{t-2}(\hat{a})$ are close to 0. It is easy to understand why the algorithm selects it in the first case, as the arm is currently stable and it has the highest estimated reward. In the second case, $\mu_t(\hat{a})$ seems variable in the past two steps. Even if $\mu_t(\hat{a})$ had kept moving down (that is, the worst possibility), the estimated reward would have fallen as well, since arm $\hat{a}$ still has the highest estimated reward, Algorithm 1 will select it.

2. Idle arm example: let us say arm $\hat{a}$ has the highest estimated reward at time $t$, and it was not selected at $t - 1$. By the decision rule, Algorithm 1 will select this arm for sure since $|\lambda_{t-1}(\hat{a}) - \lambda_{t-2}(\hat{a})| = 0$.

From these examples, we can see that exploration and exploitation are balanced in a way that takes into account the variability in the estimation procedure rather than by simply flipping a coin. It boosts gaining knowledge for active but variable arms and idle arms.

Algorithm 1 AFF-$d$-Greedy

**Require:** $d \in (0,1)$; $\eta \in (0,1)$.

**Initialisation:** play each arm once.

**for** $t = |A| + 1, \cdots, T$ **do**

- find $a_t = \arg \max_{a' \in A} \hat{Y}_{t-1}(a')$;
- if $|\lambda_{t-1}(a_t) - \lambda_{t-2}(a_t)| \geq d$ **then**
  - choose $a_t$ uniformly from $A$;
- **end if**
- select arm $a_t$ and observe reward $Y_t(a_t)$;
- update $\hat{Y}_t(a_t)$.

**end for**

3.2 Upper Confidence Bound

Another type of algorithms uses upper confidence bounds for selection. The idea is that, instead of the plain sample average, an exploration bonus $B_t$ is added to account for the uncertainty in the estimation, and the arm with highest potential of being optimal will be selected. This exploration bonus is typically derived using concentration inequalities (e.g., Hoeffding, 1963). The UCB1 algorithm introduced by Auer et al. (2002) is a classic method. In latter works, UCB1 was often called simply UCB. For any reward distribution that is
bound in $[0,1]$, the UCB algorithm picks the arm which maximise the quantity $\bar{Y}_t + \sqrt{\frac{2 \log t}{N_t}}$, where $\bar{Y}_t$ is the sample average and $N_t$ is the number of times this arm was played up to time $t$. The exploration bonus $B_t = \sqrt{\frac{2 \log t}{N_t}}$ was derived using the Chernoff-Hoeffding bound. It is proved that the UCB algorithm achieves logarithmic regret uniformly over time (Auer et al., 2002).

For better adaptation in dynamic environments, we replace $\bar{Y}_t$ with $\hat{Y}_t$, and modify the upper bound accordingly. This results to the AFF-UCB algorithm in Algorithm 2. The upper bound for selection at time $t+1$ takes the form $\hat{Y}_t + B_t$. We set $B_t$ to:

$$B_t = \sqrt{-\log(0.05)/(2w_t)} \mathbb{I}(t - t_{last} = 0) + \sqrt{\frac{2}{w_t} (t - t_{last})^{1/|A|}}, \quad (14)$$

where $t_{last}$ is the last time instant that the arm was observed; $w_t$, $k_t$, and $s^2_t$ are quantities related to the AFF estimation (see Section 2; $k_t = \sum_{i=1}^t (\prod_{p=1}^{t-1} \lambda^2_p)$ and its recursive updating is $k_t = \lambda^2_{t-1} k_{t-1} + 1$).

From (14), $B_t$ is a combination of two components. It can be interpreted by considering two cases:

1. if an arm was observed at the previous time step, $t$ (i.e., $t - t_{last} = 0$), $B_t = \sqrt{-\log(0.05)/(2w_t)}$;

2. if an arm was not observed at the previous time step, $B_t = \sqrt{\frac{s^2_t}{w_t} (t - t_{last})^{1/|A|}}$.

In the former case, $B_t = \sqrt{-\log(0.05)/(2w_t)}$ is derived via the Chernoff-Hoeffding bound in a similar way to the derivation of UCB (see Appendix A for details). However, for an unselected arm, if we use the same $B_t$ expression, its upper bound will be static since $\hat{Y}_t$, $w_t$ and $k_t$ do not change. As a consequence, it will only be selected if the arm with current highest upper bound drops below it. This is not desirable since in a changing environment, any sub-optimal arm can become optimal at any time. This motivates us to deliberately add some inflation to the upper bound of unselected arms to impose exploration, which leads to $B_t = \sqrt{\frac{s^2_t}{w_t} (t - t_{last})^{1/|A|}}$. Note here $B_t$ decreases with the number of arms, $|A|$. This makes use of the fact that as $|A|$ increases, the population of arms will “fill” more the reward space and more opportunities will arise for picking high reward arms.

**Algorithm 2 AFF-UCB**

**Require:** $\eta \in (0, 1)$.

**Initialisation:** play each arm $M$ times.

**for** $t = M|A| + 1, \cdots , T$ **do**

- for all $a \in A$, compute $B_{t-1}(a)$ according to (14);
- find $a_t = \arg \max_{a' \in A} \left( \hat{Y}_{t-1}(a') + B_{t-1}(a') \right)$;
- select arm $a_t$ and observe reward $Y_t(a_t)$;
- update $\hat{Y}_t(a_t)$, $w_t(a_t)$, $k_t(a_t)$, $s^2_t(a_t)$, and $t_{last}(a_t)$.

**end for**

Note here, we use a longer burn-in period since we need to initialise the estimation of data variance. In the simulation study in Section 4, we choose $M = 10$. 

9
3.3 Thompson Sampling

Algorithm 3 AFF-TS for Dynamic Bernoulli Bandits

Require: \( Beta(\alpha_0(a), \beta_0(a)) \) for all \( a \in \mathcal{A}; \eta \in (0, 1) \).

Initialisation: play each arm once.

for \( t = |\mathcal{A}| + 1, \cdots, T \) do

for all \( a \in \mathcal{A} \), draw a sample \( x(a) \) from \( Beta(\alpha_{t-1}(a), \beta_{t-1}(a)) \);

find \( a_t = \arg \max_{a' \in \mathcal{A}} x(a') \);

select arm \( a_t \) and observe reward \( Y_t(a_t) \);

update \( \alpha_t(a_t) \) and \( \beta_t(a_t) \) according to (19)-(20).

end for

Recently, researchers (e.g., Scott, 2015) have given more attention to the Thompson Sampling (TS) method which can be dated back to Thompson (1933). It is an approach based on Bayesian principles. A (usually conjugate) prior is assigned to the expected reward of each arm at the beginning, and the posterior distribution of the expected reward is sequentially updated through successive arm selection. A decision rule is constructed using this posterior distribution. At each round, a random sample is drawn from the posterior distribution of each arm, and the arm with the highest sample value is selected.

For the static Bernoulli bandit, following the approach of Chapelle and Li (2011), it is convenient to choose the Beta distribution, \( Beta(\alpha_0, \beta_0) \), as a prior. The posterior distribution is then \( Beta(\alpha_t, \beta_t) \) at time \( t \), and the parameters \( \alpha_t \) and \( \beta_t \) can be updated recursively as follows: if an arm is selected at time \( t \),

\[
\alpha_t = \alpha_{t-1} + Y_t, \quad (15)
\]
\[
\beta_t = \beta_{t-1} + 1 - Y_t; \quad (16)
\]

otherwise,

\[
\alpha_t = \alpha_{t-1}, \quad (17)
\]
\[
\beta_t = \beta_{t-1}. \quad (18)
\]

The simplicity and effectiveness in real applications (Scott, 2015) make TS a good candidate for dynamic bandits. However, it has similar issues in tracking \( \mu_t \) as in \( \epsilon \)-Greedy and UCB. For illustration, assume an arm is observed all the time, and one can re-write the recursions in (15)-(16) as:

\[
\alpha_t = \alpha_0 + \sum_{i=1}^{t} Y_i,
\]
\[
\beta_t = \beta_0 + \sum_{i=1}^{t} 1 - \sum_{i=1}^{t} Y_i.
\]

As a result, the posterior distribution \( Beta(\alpha_t, \beta_t) \) keeps full memory of all the past observations, making posterior inference less responsive to observations near time \( t \).
To modify the above updating, we use the intermediate quantities $m_t$ and $w_t$ from (4)-(5). If an arm is selected at time $t$,

$$\alpha_t = \alpha_0 + m_t; \quad (19)$$
$$\beta_t = \beta_0 + w_t - m_t; \quad (20)$$

otherwise, $\alpha_t$ and $\beta_t$ are updated via (17)-(18). Using these updates, we propose in Algorithm 3 the AFF-TS algorithm for dynamic Bernoulli bandits.

3.3.1 Optimistic Thompson Sampling

We now look at some popular extensions of TS. May et al. (2012) introduced the optimistic version of Thompson sampling called Optimistic Thompson Sampling (OTS), where the drawn sample value is replaced by its posterior mean if the former is smaller. That is to say, for each arm, the score used for decision will never be smaller than the posterior mean. OTS boosts further the exploration of highly uncertain arms compared to TS, as OTS increases the probability of getting a high score for arms with high posterior variance.

However, OTS has the same problem as TS when applied to a dynamic problem, that it uses the full reward history to update the posterior distribution. We propose the AFF version of OTS in Algorithm 4.

Algorithm 4 AFF-OTS for Dynamic Bernoulli Bandits

Require: $Beta(\alpha_0(a), \beta_0(a))$ for all $a \in A$; $\eta \in (0, 1)$.

Initialisation: play each arm once.

for $t = |A| + 1, \cdots, T$ do

for all $a \in A$, draw a sample $x(a)$ from $Beta(\alpha_{t-1}(a), \beta_{t-1}(a))$,

and replace $x(a)$ with $\frac{\alpha_{t-1}(a)}{\alpha_{t-1}(a) + \beta_{t-1}(a)}$ if $x(a)$ is smaller;

find $a_t = \arg \max_{a' \in A} x(a')$;

select arm $a_t$ and observe reward $Y_t(a_t)$;

update $\alpha_t(a_t)$ and $\beta_t(a_t)$ according to (19)-(20).

end for

3.3.2 Tuning Parameter $C$ in Dynamic Thompson Sampling

The Dynamic Thompson Sampling (DTS) algorithm was introduced by Gupta et al. (2011) specifically for solving the dynamic Bernoulli bandit problem of interest here. The DTS algorithm uses a pre-determined threshold $C$ in updating the posterior parameters $\alpha_t$ and $\beta_t$ while using the standard Thompson sampling technique for arm selection. For the arm that is selected at time $t$, if $\alpha_{t-1} + \beta_{t-1} < C$, the posterior parameters are updated via (15)-(16); otherwise when $\alpha_{t-1} + \beta_{t-1} \geq C$,

$$\alpha_t = (\alpha_{t-1} + Y_t) \frac{C}{C + 1},$$
$$\beta_t = (\beta_{t-1} + 1 - Y_t) \frac{C}{C + 1}.$$

To understand, let $\hat{\mu}_t$ denote the posterior mean, and assume an arm is observed all the time. Say at time $s$ the arm achieves the threshold, i.e., $\alpha_t + \beta_t = C$ for $t = s$ and onwards.
Following (17)-(21) of Gupta et al. (2011),

\[ \hat{\mu}_t = (1 - \frac{1}{C+1}) \hat{\mu}_{t-1} + \frac{1}{C+1} Y_t, \]  

(21)

which is a weighed average of \( \hat{\mu}_{t-1} \) and the observation \( Y_t \). The recursion of \( \hat{\mu}_t \) is similar to the EWMA scheme (Roberts, 1959). Essentially, the DTS algorithm uses the threshold \( C \) to bound the total amount of reward history used for updating the posterior distribution. Once it comes to the threshold, the algorithm yields putting more weight on newer observations.

Although it was demonstrated in Gupta et al. (2011) that the DTS algorithm has the ability to track the changes in the expected reward, the performance of the algorithm is very sensitive to the choice of \( C \). In our numerical simulations (see Section 4.2.2), we found that the performance of the DTS algorithm varies a lot with different \( C \) values. However, in Gupta et al. (2011), the authors did not provide tuning methods for \( C \). To address this issue, we propose below two different ways to tune \( C \) adaptively at each time step using AFF estimations (AFF-DTS1 & 2 resp.).

**AFF-DTS1** From the numerical results in Gupta et al. (2011, sect. IV.C), the optimal \( C \) is related to the the speed of change of \( \mu_t \). This motivates us to tune \( C \) according to the variance of the data stream. We can use the AFF variance, \( s_t^2 \), defined in (10) as an estimation of the data variance. One option is to use \( C_t \propto \frac{1}{s_t^2} \); since high \( s_t^2 \) indicates more dynamics in \( \mu_t \), a shorter reward history is required. For example in the numerical examples in Section 4.2.2, we will use \( C_t = \frac{4}{s_t^2} - 1 \).

**AFF-DTS2** Another way to set \( C_t \) could be based on the similarity of the posterior mean in DTS and the AFF mean introduced in Section 2. In particular, in (21) the posterior mean is given by:

\[ \hat{\mu}_t = (1 - \frac{1}{C+1}) \hat{\mu}_{t-1} + \frac{1}{C+1} Y_t. \]

Using (2)-(5), one can re-write the the AFF mean as:

\[ \hat{Y}_t = \left( 1 - \frac{1}{w_t} \right) \hat{Y}_{t-1} + \left( \frac{1}{w_t} \right) Y_t. \]

Therefore, at each time step \( t \), we can set \( C_t = w_t - 1 \).

### 4. Numerical Results

In this section, we illustrate the performance improvements on \( \epsilon \)-Greedy, UCB, and TS using AFFs. We consider two different dynamic scenarios for the expected reward \( \mu_t \): abruptly changing and drifting. For the abruptly changing scenario, instead of manually setting up change points in \( \mu_t \) as in Yu and Mannor (2009) and Garivier and Moulines (2011), we set up change-point instants for an arm by an exponential clock (see Section 4.1.1). In the drifting scenario, the evolution of the expected reward \( \mu_t \) is driven by a random walk in the interval \((0,1)\). For the random walk case we use two different models: the first model is inspired by Slivkins and Upfal (2008) where \( \mu_t \) is modelled by a random walk with reflecting bounds; the second model is to use a transformation function on a random walk. For each
scenario, we test the performance with 2, 50 and 100 arms; the two-armed examples are used for the purpose of illustration, and the latter examples (50 and 100 arms) are used to evaluate the performance with a large number of arms. We also demonstrate the robustness of the AFF MAB algorithms to tuning, specifically, sensitivity to the step size, $\eta$. Finally, we use a two-armed example to show that the modified DTS algorithms, i.e., AFF-DTS1 and AFF-DTS2, can reduce the performance sensitivity of DTS to the input parameter $C$.

4.1 Performance for Different Dynamic Models

We first use two-armed examples to compare the performance of AFF-$d$-Greedy, AFF-UCB, and AFF-TS/AFF-OTS to the standard methods $\epsilon$-Greedy, UCB, and TS respectively. We consider four different cases: two cases for the abruptly changing scenario, and two for the drifting scenario; each case has 100 independent replications. The length of each simulated experiment is $T = 10,000$. For the $\epsilon$-Greedy method, we evaluate over a grid of choice of $\epsilon$, $\epsilon \in (0.1, \cdots, 0.9)$, and report performance for the best choice. We use step size $\eta = 0.001$ for all AFF MAB algorithms. For AFF-$d$-Greedy, we set the threshold $d = \eta$. For all Thompson sampling based algorithms, we use $Beta(2, 2)$ as the prior.

4.1.1 Abruptly Changing Expected Reward

The expected reward $\mu_t$ is simulated by the following exponential clock model:

\[
\begin{align*}
J(t) & \sim \text{homogeneous Poisson process with rate } \theta, \\
\mu_t & = \begin{cases} 
\text{sample from } U(r_l, r_u), & \text{if } J(t) > J(t-1), \\
\mu_{t-1}, & \text{otherwise,}
\end{cases} \\
\mu_0 & = \text{sample from } U(r_l, r_u).
\end{align*}
\]

(22)

The parameter $\theta$ determines the frequency at which change point occurs. At each change point, the new expected reward is sampled from a uniform distribution $U(r_l, r_u)$. We generate two different cases, Case 1 and 2. Parameters used for generating these cases can be

![Figure 1: Abruptly changing scenario: example $\mu_t$ sampled from the model in (22) with parameters displayed in Table 1. Arm 1 is black and Arm 2 is red. Left/right are for Case 1/2 resp.](image)

(a) Case 1: a simulated $\mu_t$ against $t$. (b) Case 2: a simulated $\mu_t$ against $t$.
found in Table 1. For visualisation purposes, we display in Figure 1 a single simulated path for \( \mu_t \) against \( t \). For Case 1, we distinguish the two arms by varying their frequency of change, but in the long run, for high \( T \), \( \bar{\mu}_T = \mathbb{E}\left[\frac{1}{T} \sum_{i=1}^{T} \mu_i\right] \) are the same. In Case 2, Arm 1 has a higher \( \bar{\mu}_T \).

Table 1: Parameters used in the exponential clock model shown in (22).

|          | Case 1   |          | Case 2   |
|----------|----------|----------|----------|
|          | \( \theta \) | \( r_l \) | \( r_u \) | \( \theta \) | \( r_l \) | \( r_u \) |
| Arm 1    | 0.001    | 0.0      | 1.0      | 0.001    | 0.3      | 1.0      |
| Arm 2    | 0.010    | 0.0      | 1.0      | 0.010    | 0.0      | 0.7      |

Figure 2: Results for the two-armed Bernoulli bandit with abruptly changing expected rewards. The top row displays the cumulative regret over time; results are averaged over 100 replications. The bottom row are boxplots of total regret at time \( t = 10,000 \). Trajectories are sampled from (22) with parameters displayed in Table 1.
In Figure 2, we present comparisons in each case. The bottom row of Figure 2 shows boxplots of the total regret $R_T$ as in (1). In addition, the top row of Figure 2 displays the cumulative regret over time; the results are averaged over 100 independent replications. The plots are good evidence that our algorithms yield improved performance over standard approaches. In particular, the improvement is distinguishable in Case 1, for which the two arms have the same $\bar{\mu}_T$. In the case that one arm’s mean dominates in the long run (Case 2), the AFF MAB algorithms perform similarly to the standard methods. However, the AFF MAB algorithms have smaller variance among replications. In both cases, AFF-OTS has the best performance in terms of total regret.

![Boxplot](image1)

(a) Case 3: a simulated $\mu_t$ against $t$.

![Boxplot](image2)

(b) Case 4: a simulated $\mu_t$ against $t$.

Figure 3: Drifting scenario: visualisation of single simulated $\mu_t$. The left plot shows a random example simulated via the model in (23) with $\sigma^2_\mu = 0.0001$, and the right plot is a random example simulated via the model in (24) with $\sigma^2_\mu = 0.001$.

### 4.1.2 Drifting Expected Reward

For the drifting scenario, we use two different models. The first is the random walk model with reflecting bounds introduced in Slivkins and Upfal (2008), which is:

$$
\mu_t = f(\mu_{t-1} + \omega_t), \quad \omega_t \sim N(0, \sigma^2_\mu),
$$

where $f(x) = \begin{cases} 
  x' & x' \leq 1 \\
  1 - (x' - 1) & x' > 1 
\end{cases}$, and $x' = |x|$ (mod 2). Slivkins and Upfal (2008) showed that $\mu_t$ generated by this model is stationary, that is in the long run, $\mu_t$ will be distributed according to a uniform distribution. The parameter $\sigma^2_\mu$ used in the model controls the rate of change in an arm. In the left panel of Figure 3, we illustrate a single sample from (23) with $\sigma^2_\mu = 0.0001$ (Case 3). Similar to Case 1, the two arms in Case 3 have the same $\bar{\mu}_T$.

The second model we use to simulate drifting arms is:

$$
\begin{align*}
  z_0 &= \text{sample from } U(0, 1), \\
  z_t &= z_{t-1} + \omega_t, \quad \omega_t \sim N(0, \sigma^2_\mu), \\
  \mu_t &= \frac{1}{1 + \exp(-z_t)},
\end{align*}
$$

15
where the expected reward $\mu_t$ is transformed from the random walk $z_t$. Since a random walk diverges in the long run, any trajectory will move closer and closer to one of the boundaries 0 or 1. Again, the parameter $\sigma^2_\mu$ controls the speed that $\mu_t$ evolves. In the right panel of Figure 3, we illustrate a single sample from (24) with $\sigma^2_\mu = 0.001$ (Case 4).

Figure 4: Results for the two-armed Bernoulli bandit with drifting expected rewards. The top row displays the cumulative regret over time; results are averaged over 100 independent replications. The bottom row are boxplots of total regret at time $t = 10,000$. Trajectories for Case 3 are sampled from the model in (23) with $\sigma^2_\mu = 0.0001$, and trajectories for Case 4 are sampled from (24) with $\sigma^2_\mu = 0.001$.

The results for the drifting scenario can be found in Figure 4. The top row of Figure 4 displays the cumulative regret averaged over 100 independent replications, and the bottom row shows boxplots of total regret. For Case 3 that is simulated from the model in (23), we can see that the AFF MAB algorithms outperform the standard approaches. For Case 4 that is simulated from the model in (24), there is a solid improvement in the performance of TS, while UCB and AFF-UCB perform similarly. Similar to the abruptly changing case, AFF-OTS performs very well in both drifting cases in terms of total regret. It was more
challenging to deploy adaptive estimation at UCB because it was harder to interpret the estimate from AFF estimator (it is more dynamic with less memory) and modify the upper bound.

4.1.3 LARGE NUMBER OF ARMS

Modern applications of bandits problem can involve a large number of arms. For example, in online advertising, we need to optimise among hundreds of websites. Therefore, we evaluate the performance of our AFF MAB algorithms with large number of arms. We repeat earlier experiments with 50 and 100 arms. Results can be seen from Figures 5-8. It can be seen that performance gains hold for a large number of arms, and are very pronounced for all methods including UCB (that was more challenging to improve). For Case 1 and 3, the results for fifty-armred and one-hundred-armred examples are very similar to the two-armred ones. For Case 2 and 4, unlike the two-armred examples where the improvement of adaptive estimation on UCB is marginal, with 50 and 100 arms, AFF-UCB performs better than UCB. In addition, AFF-OTS has good performance in all cases. In summary, with a large number of arms, our algorithms perform much better than the standard methods. Interestingly, in all cases, results for 50 and 100 arms are very similar. This could be attributed to both 50 and 100 arms being numbers large enough to fill the reward space [0,1] well enough so that the decision maker in both cases finds high value arms.

Figure 5: Large number of arms: abruptly changing environment (Case 1).

Figure 6: Large number of arms: abruptly changing environment (Case 2).
4.2 Robustness to Tuning

We have already seen the improvements the AFF MAB algorithms can offer in different dynamic scenarios. We now move on to examine the sensitivity of performance to the tuning parameters.

4.2.1 Initialisation in the AFF MAB Algorithms

In this section, we examine the influence of the step size $\eta$ on the AFF MAB algorithms. We present only for Case 3 (see Section 4.1.2) for the sake of brevity; results for other cases are very similar and hence omitted. For each AFF MAB algorithm, we do experiments with $\eta_1 = 0.0001, \eta_2 = 0.001, \eta_3 = 0.01$, and $\eta_4(t) = 0.0001/s_t^2$, where $s_t^2$ is the AFF variance defined in (10). Note here $\eta_1$, $\eta_2$, and $\eta_3$ are fixed, while $\eta_4$ can change over time. Figures 9-12 display the results for AFF-$d$-Greedy, AFF-UCB, AFF-TS, and AFF-OTS respectively. From the results, we can see that the algorithms are not particularly sensitive to the step size $\eta$. 
Figure 9: AFF-d-Greedy algorithm with different $\eta$ values. $\eta_1 = 0.0001$, $\eta_2 = 0.001$, $\eta_3 = 0.01$, and $\eta_4(t) = 0.0001/s_t^2$, where $s_t^2$ is as in (10).

Figure 10: AFF-UCB algorithm with different $\eta$ values. $\eta_1 = 0.0001$, $\eta_2 = 0.001$, $\eta_3 = 0.01$, and $\eta_4(t) = 0.0001/s_t^2$, where $s_t^2$ is as in (10).

Figure 11: AFF-TS algorithm with different $\eta$ values. $\eta_1 = 0.0001$, $\eta_2 = 0.001$, $\eta_3 = 0.01$, and $\eta_4(t) = 0.0001/s_t^2$, where $s_t^2$ is as in (10).
Figure 12: AFF-OTS algorithm with different $\eta$ values. $\eta_1 = 0.0001$, $\eta_2 = 0.001$, $\eta_3 = 0.01$, and $\eta_4(t) = 0.0001/s_t^2$, where $s_t^2$ is as in (10).

4.2.2 Using Adaptive Forgetting Factors to Tune Parameter $C$ in Dynamic Thompson Sampling

Figure 13: Boxplot of total regret for algorithms DTS, AFF-DTS1, and AFF-DTS2. Acronym like DTS-C5 represents the DTS algorithm with parameter $C = 5$. Similarly acronym like AFF-DTS1-C5 represents the AFF-DTS1 algorithm with initial value $C_0 = 5$. We also plot the result of AFF-OTS for benchmark.

In Section 3.3.2, we discussed that we can use adaptive estimation to tune the input parameter $C$ in the DTS algorithm proposed by Gupta et al. (2011), and we offered two self-tuning solutions, AFF-DTS1 and AFF-DTS2. We use the two-armed abruptly changing example (Case 1 in Section 4.1.1) to illustrate how the AFF version algorithms can reduce the sensitivity to $C$. 

20
We test $C = 5, 10, 100, \text{ and } 1000$ for DTS, AFF-DTS1, and AFF-DTS2. It (the $C$ value) works as the initial value of $C_t$ in AFF-DTS1 and AFF-DTS2. Step size $\eta = 0.001$ is used for AFF related algorithms. Figure 13 displays the boxplot of total regret. We also plotted the result of AFF-OTS for benchmark since it has good performance in all cases studied in the previous section. From Figure 13, the performance of AFF-DTS1 and AFF-DTS2 are very stable, while DTS is very sensitive to $C$. With a bad choice of $C$ (i.e., 100 and 1000 in this case), the total regret of DTS is much higher than AFF-DTS1 and AFF-DTS2.

5. Conclusion

We have seen that the performance of popular MAB algorithms can be improved significantly using AFFs. The improvements are substantial when the arms are not distinguishable in the long run, i.e., the arms have the same long-term averaged expected reward $\bar{\mu}_T$, $\bar{\mu}_T = \mathbb{E}\left[\frac{1}{T} \sum_{t=1}^{T} \mu_t\right]$. For the case that one arm has a higher $\bar{\mu}_T$ (e.g., the two-armed example in Case 2), gains for the AFF MAB algorithms seem marginal, but there is no loss in performance, so practitioners could be encouraged to implement our adaptive methods when they do not have knowledge of the behaviour of $\mu_t$ with time. In addition, the performance gains for a large number of arms are very pronounced for all methods. Finally, the AFF MAB algorithms we proposed are easy to implement; they do not require any prior knowledge about the dynamic environment, and seem to be more robust to tuning parameters.

Combining adaptive estimation with UCB was more challenging. The reason was that one needs to reinterpret the estimate of $\mu_t$ from a stable long run average to a “more dynamic” estimator (with less memory), and modify accordingly the upper bound. We should mention here that our algorithm AFF-UCB turns out to be similar to D-UCB (Garivier and Moulines, 2011). In D-UCB, a fixed forgetting factor approach is used for estimation. However, it requires knowing the number of switch points to tune the the forgetting factor, which we do not require.

We conclude by mentioning some interesting avenues for future work. One extension is to apply AFF-based methods for more challenging problems, e.g., rotting bandits (Levine et al., 2017), contextual bandits (Langford and Zhang, 2008; Li et al., 2010), and applications like online advertising. Another extension could involve a rigorous analysis of how the bias in AFF estimation varies with time and how can this affect the selection in MAB problems.
Appendix A. Derivations for Exploration Bonus in AFF-UCB

We present how we derive the following exploration bonus $B_t$ (used for selection at time step $t + 1$) in the AFF-UCB algorithm:

$$B_t = \sqrt{-\log(0.05)}\frac{1}{2(w_t)^2/k_t}(t-t_{last} = 0) + \sqrt{\frac{2}{w_t}(t-t_{last})^{1/|A|}}.$$

The construction of $B_t$ are the combination of two parts: $B_t^{(1)} = \sqrt{-\log(0.05)}\frac{1}{2(w_t)^2/k_t}$, and $B_t^{(2)} = \sqrt{\frac{2}{w_t}(t-t_{last})^{1/|A|}}$, hence we can separate $B_t$ into two cases:

1. if an arm was observed at time $t$, (i.e., $t-t_{last} = 0$), $B_t = B_t^{(1)}$;
2. if an arm was not observed at time $t$, $B_t = B_t^{(2)}(t-t_{last})^{1/|A|}$.

We now present how to derive the first part $B_t^{(1)}$. According to (2)-(5), the AFF mean $\hat{Y}_t$ for independent data stream $Y_1, \cdots, Y_t$ is

$$\hat{Y}_t = \frac{m_t}{w_t}, \quad m_t = \sum_{i=1}^{t} \left( \prod_{p=1}^{i-1} \lambda_p \right) Y_i, \quad w_t = \sum_{i=1}^{t} \left( \prod_{p=1}^{i-1} \lambda_p \right).$$

According to Hoeffding’s inequality, we have

$$P(E[\hat{Y}_t] - \hat{Y}_t \geq B_t^{(1)}) = P\left( \frac{m_t}{w_t} - \frac{m_t}{w_t} \geq B_t^{(1)} \right) = P\left( \frac{m_t}{w_t} - m_t \geq B_t^{(1)}w_t \right) \leq \exp\left( - \frac{2(B_t^{(1)})^2(w_t)^2}{k_t} \right),$$

where $k_t = \sum_{i=1}^{t} \left( \prod_{p=1}^{t-1} \lambda_p^2 \right)$. We use the fact that $\left( \prod_{p=1}^{t-1} \lambda_p \right) Y_i$ is bounded in $[0, \left( \prod_{p=1}^{t-1} \lambda_p \right)]$ since $Y_i$ is bounded in $[0, 1]$. Whilst the use of Hoeffding’s inequality is typically for i.i.d variables, there are similar expressions for Markov chains (Glynn and Ormoneit, 2002), which fits to our framework.

Let $\xi$ denote $P(E[\hat{Y}_t] - \hat{Y}_t \geq B_t^{(1)})$, and set $\xi = \exp\left( - \frac{2(B_t^{(1)})^2(w_t)^2}{k_t} \right)$, we have

$$B_t^{(1)} = \sqrt{\frac{-\log(\xi)k_t}{2(w_t)^2}}.$$

$\xi$ is the the probability that the difference between $E[\hat{Y}_t]$ and $\hat{Y}_t$ exceeds $B_t^{(1)}$. The form of $B_t^{(1)}$ is similar to the exploration bonus in UCB (Auer et al., 2002), and in UCB, $\xi$ was set to $\xi = t^{-4}$ to obtain a tighter upper bound as the number of trials increases (that is, exploration is reduced over time). This is sensible in static cases as the estimates converge.
with $t$. However, as we are interested in dynamic cases, we are in favour of a bound that keeps a certain level of exploration over time, and hence we take a constant $\xi = 0.05$, and get the first part of the exploration bonus

$$B_t^{(1)} = \sqrt{-\log(0.05)}$2(wt)2/kt$.$

If we only use $B_t^{(1)}$ as the exploration bonus, for an unselected arm, its value will be static since $w_t$ and $k_t$ do not change. As a consequence, it will only be selected if the arm with current highest upper bound drops below it. This motivates us to add some inflation deliberately. A naive choice is to add the data variance, and this leads to the second part $B_t^{(2)} = \sqrt{\frac{s^2}{w_t}}$ where $s_t^2$ is the AFF variance defined in (10).
References

Christoforos Anagnostopoulos, Dimitris K. Tasoulis, Niall M. Adams, Nicos G. Pavlidis, and David J. Hand. Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(2):139–166, 2012.

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

Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, 2008.

Dean A. Bodenham and Niall M. Adams. Continuous monitoring for changepoints in data streams using adaptive estimation. *Statistics and Computing*, 27(5):1257–1270, 2016.

Eric Brochu, Matthew D. Hoffman, and Nando de Freitas. Portfolio allocation for Bayesian optimization. *arXiv:1009.5419v2*, 2011.

Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems 24*, pages 2249–2257. Curran Associates, Inc., 2011.

Luis DaCosta, Alvaro Fialho, Marc Schoenauer, and Michele Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 913–920, 2008.

Aurélien Garivier and Olivier Cappe. The KL-UCB algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th Annual Conference on Learning Theory*, volume 19, pages 359–376, 2011.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Algorithmic Learning Theory*, volume 6925 of *Lecture Notes in Artificial Intelligence*, pages 174–188. Springer-Verlag Berlin, 2011.

Peter W. Glynn and Dirk Ormoneit. Hoeffding’s inequality for uniformly ergodic Markov chains. *Statistics and Probability Letters*, 56(2):143–146, 2002.

Ole-Christoffer Granmo and Stian Berg. Solving non-stationary bandit problems by random sampling from sibling Kalman filters. In *Proceedings of Trends in Applied Intelligent Systems*, PT III, volume 6098 of *Lecture Notes in Artificial Intelligence*, pages 199–208. Springer-Verlag Berlin, 2010.

Neha Gupta, Ole-Christoffer Granmo, and Ashok Agrawala. Thompson sampling for dynamic multi-armed bandits. In *Proceedings of the 10th International Conference on Machine Learning and Applications (ICMLA)*, volume 1, pages 484–489, 2011.

Simon S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, Upper Saddle River, N.J., 4th edition, 2002.
Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Levente Kocsis and Csaba Szepesvari. Discounted UCB. In *2nd PASCAL Challenges Workshop*, Venice, 2006. URL https://www.lri.fr/~sebag/Slides/Venice/Kocsis.pdf.

D. E. Koulouriotis and A. Xanthopoulos. Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation*, 196(2):913–922, 2008.

Volodymyr Kuleshov and Doina Precup. Algorithms for the multi-armed bandit problem. *arXiv:1402.6028v1*, 2014.

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

Nir Levine, Koby Crammer, and Shie Mannor. Rotting bandits. *arXiv:1702.07274v3*, 2017.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670. ACM, 2010.

Benedict C. May, Nathan Korda, Anthony Lee, and David S. Leslie. Optimistic Bayesian sampling in contextual-bandit problems. *The Journal of Machine Learning Research*, 13(1):2069–2106, 2012.

Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of optimal queuing network control. *Mathematics of Operations Research*, 24(2):293–305, 1999.

William H. Press. Bandit solutions provide unified ethical models for randomized clinical trials and comparative effectiveness research. *Proceedings of the National Academy of Sciences of the United States of America*, 106(52):22387–22392, 2009.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–536, 1952.

S. W. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1(3):239–250, 1959.

Steven L. Scott. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

Steven L. Scott. Multi-armed bandit experiments in the online service economy. *Applied Stochastic Models in Business and Industry*, 31(1):37–45, 2015.

Weiwei Shen, Jun Wang, Yu-Gang Jiang, and Hongyuan Zha. Portfolio choices with orthogonal bandit learning. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI2015)*, pages 974–980, 2015.
Aleksandrs Slivkins and Eli Upfal. Adapting to a changing environment: The Brownian restless bandits. In 21st Conference on Learning Theory (COLT), pages 343–354, 2008.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika, 25(3/4):285–294, 1933.

Fujee Tsung and Kaibo Wang. Adaptive charting techniques: Literature review and extensions, pages 19–35. Physica-Verlag HD, Heidelberg, 2010.

Joannès Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In Proceedings of the 16th European Conference on Machine Learning, volume 3720, pages 437–448, 2005.

Sofia S. Villar, Jack Bowden, and James Wason. Multi-armed bandit models for the optimal design of clinical trials: Benefits and challenges. Statistical Science, 30(2):199–215, 2015.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. PhD thesis, Cambridge University, 1989.

Peter Whittle. Restless bandits: Activity allocation in a changing world. Journal of Applied Probability, 25(A):287–298, 1988.

Jia Yuan Yu and Shie Mannor. Piecewise-stationary bandit problems with side observations. In Proceedings of the 26th International Conference on Machine Learning, pages 1177–1184, 2009.