Learning Contextual Bandits in a Non-stationary Environment

Qingyun Wu, Naveen Iyer, Hongning Wang
Department of Computer Science, University of Virginia
Charlottesville, VA, USA
{qw2ky,nki2kd,hw5x}@virginia.edu

ABSTRACT

Multi-armed bandit algorithms have become a reference solution for handling the explore/exploit dilemma in recommender systems, and many other important real-world problems, such as display advertisement. However, such algorithms usually assume a stationary reward distribution, which hardly holds in practice as users' preferences are dynamic. This inevitably costs a recommender system consistent suboptimal performance. In this paper, we consider the situation where the underlying distribution of reward remains unchanged over (possibly short) epochs and shifts at unknown time instants. In accordance, we propose a contextual bandit algorithm that detects possible changes of environment based on its reward estimation confidence and updates its arm selection strategy respectively. Rigorous upper regret bound analysis of the proposed algorithm demonstrates its learning effectiveness in such a non-trivial environment. Extensive empirical evaluations on both synthetic and real-world datasets for recommendation confirm its practical utility in a changing environment.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Theory of computation → Online learning algorithms**; **Regret bounds**; • **Computing methodologies → Sequential decision making**;

KEYWORDS

Non-stationary Bandit; Recommender Systems; Regret Analysis

ACM Reference Format:
Qingyun Wu, Naveen Iyer, Hongning Wang. 2018. Learning Contextual Bandits in a Non-stationary Environment. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210051

1 INTRODUCTION

Multi-armed bandit algorithms provide a principled solution to the explore/exploit dilemma [2, 3, 14], which exists in many important real-world applications such as display advertisement [21], recommender systems [18], and online learning to rank [27]. Intuitively, bandit algorithms adaptively designate a small amount of traffic to collect user feedback in each round while improving their model estimation quality on the fly. In recent years, contextual bandit algorithms [9, 17, 18] have gained increasing attention due to their capability of leveraging contextual information to deliver better personalized online services. They assume the expected reward of each action is determined by a conjecture of unknown bandit parameters and given context, which give them advantages when the space of recommendation is large but the rewards are interrelated.

Most existing stochastic contextual bandit algorithms assume a fixed yet unknown reward mapping function [9, 11, 18, 20, 25]. In practice, this translates to the assumption that users' preferences remain static over time. However, this assumption rarely holds in reality as users' preferences can be influenced by various internal or external factors [7]. For example, when a sports season ends after a championship, seasonal fans might jump over to following a different sport and not have much interest in the off-season. More importantly, such changes are often not observable to the learners. If a learning algorithm fails to model or recognize the possible changes of the environment, it would constantly make suboptimal choices, e.g., keep making out-of-date recommendations to users.

In this work, moving beyond a restrictive stationary environment assumption, we study a more sophisticate but realistic environment setting where the reward mapping function becomes stochastic over time. More specifically, we focus on the setting where there are abrupt changes in terms of user preferences (e.g., user interest in a recommender system) and those changes are not observable to the learner beforehand. Between consecutive change points, the reward distribution remains stationary yet unknown, i.e., piecewise stationary. Under such a non-stationary environment assumption, we propose a two-level hierarchical bandit algorithm, which automatically detects and adapts to changes in the environment by maintaining a suite of contextual bandit models during identified stationary periods based on its interactions with the environment.

At the lower level of our hierarchical bandit algorithm, a set of contextual bandit models, referred to as slave bandits, are maintained to estimate the reward distribution in the current environment (i.e., a particular user) since the last detected change point. At the upper level, a master bandit model monitors the 'badness' of each slave bandit by examining whether its reward prediction error exceeds its confidence bound. If the environment has not changed, i.e., being stationary since the last change, the probability of observing a large residual from a bandit model learned from that environment is bounded [1, 9]. Thus the 'badness' of slave bandit models reflects possible changes of the environment. Once a change is detected with high confidence, the master bandit discards the *out-of-date* slave bandits and creates new ones to fit the new environment. Consequentially, the active slave bandit models form an admissible arm set for the master bandit to choose from. At each time, the master bandit algorithm chooses one of the active slave bandits to interact with the user, based on its estimated 'badness', and distributes user feedback to all active slave bandit models attached with this user for model updating. The master bandit model

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210051

maintains its estimation confidence of the 'badness' of those slave bandits so as to recognize the out-of-date ones as early as possible.

We rigorously prove the upper regret bound of our non-stationary contextual bandit algorithm is $O(\Gamma_T \sqrt{S_{\max}} \log S_{\max})$, in which Γ_T is the total number of ground-truth environment changes up to time T and $S_{\max}$ is the longest stationary period till time T. This arguably is the *lowest* upper regret bound any bandit algorithm can achieve in such a non-stationary environment without further assumptions. Specifically, the best one can do in such an environment is to discard the old model and estimate a new one at each true change point, as no assumption about the change should be made. This leads to the same upper regret bound achieved in our algorithm. However, as the change points are *unknown* to the algorithm ahead of time, any early or late detection of the changes can only result in an increased regret. More importantly, we prove that if an algorithm fails to model the changes a linear regret is inevitable. Extensive empirical evaluations on both a synthetic dataset and three real-world datasets for content recommendation confirmed the improved utility of the proposed algorithm, compared with both state-of-the-art stationary and non-stationary bandit algorithms.

2 RELATED WORK

Multi-armed bandit algorithms [3, 4, 9, 11, 18, 20] have been extensively studied in literature. However, most of the stochastic bandit algorithms assume the reward pertaining to an arm is determined by an unknown but *fixed* reward distribution or a context mapping function. This limits the algorithms to a stationary environment assumption, which is restrictive considering the non-stationary nature of many real-world applications of bandit algorithms.

There are some existing works studying the non-stationary bandit problems. A typical non-stationary environment setting is the abruptly changing environment, or piecewise stationary environment, in which the environment undergoes abrupt changes at unknown time points but remains stationary between two consecutive change points. To deal with such an environment, Hartland et al. [16] proposed the $\gamma-$Restart algorithm, in which a discount factor γ is introduced to exponentially decay the effect of past observations. Garivier and Moulines [10] proposed a discounted-UCB algorithm, which is similar to the $\gamma-$Restart algorithm in discounting the historical observations. They also proposed a sliding window UCB algorithm, where only observations inside a sliding window are used to update the bandit model. Yu and Mannor [26] proposed a windowed mean-shift detection algorithm to detect the potential abrupt changes in the environment. An upper regret bound of $O(\Gamma_T \log(T))$ is proved for the proposed algorithm, in which Γ_T is the number of ground-truth changes up to time T. However, they assume that at each iteration, the agent can query a subset of arms for additional observations. Slivkins and Upfal [23] considered a continuously changing environment, in which the expected reward of each arm follows Brownian motion. They proposed a UCB-like algorithm, which considers the volatility of each arm in such an environment. The algorithm restarts in a predefined schedule to account for the change of reward distribution.

Most existing solutions for non-stationary bandit problems focus on context-free scenarios, which cannot utilize the available contextual information for reward modeling. Ghosh et al. proposed an algorithm in [13] to deal with environment misspecification in contextual bandit problems. Their algorithm comprises a hypothesis test for linearity followed by a decision to use either the learnt linear contextual bandit model or a context-free bandit model. But this algorithm still assumes a stationary environment, i.e., neither the ground-truth linear model nor unknown models are changing over time. Liu et al. [8] proposed to use cumulative sum and Page-Hinkley Test to detect sudden changes in the environment. An upper regret bound of $O(\sqrt{\Gamma_T T} \log T)$ is proved for one of their proposed algorithms. However, this work is limited to a simplified Bernoulli bandit environment. Recently, Luo et al [22] studied the non-stationary bandit problem and proposed several bandit algorithms with statistical tests to adapt to changes in the environment. They analyzed various notions of regret including interval regret, switching regret, and dynamic regret. Hariri et al. [15] proposed a contextual Thompson sampling algorithm with a change detection module, which involves iteratively applying a combination of cumulative sum charts and bootstrapping to capture potential changes of user preference in interactive recommendation. But no theoretical analysis is provided about this proposed algorithm.

3 METHODOLOGY

We develop a contextual bandit algorithm for a non-stationary environment, where the algorithm automatically detects the changes in the environment and maintains a suite of contextual bandit models for each detected stationary period. In the following discussions, we will first describe the notations and our assumptions about the non-stationary environment, then carefully illustrate our developed algorithm and corresponding regret analysis.

3.1 Problem Setting and Formulation

In a multi-armed bandit problem, a learner takes turns to interact with the environment, such as a user or a group of users in a recommender system, with a goal of maximizing its accumulated reward collected from the environment over time T. At round t, the learner makes a choice a_t among a finite, but possibly large, number of arms, i.e., $a_t \in \mathcal{A} = \{a_1, a_2, \ldots, a_K\}$, and gets the corresponding reward r_{a_t}, such as a user clicks on a recommended item. In a contextual bandit setting, each arm a is associated with a feature vector $\mathbf{x}_a \in \mathbb{R}^d$ ($\|\mathbf{x}_a\|_2 \leq 1$ without loss of generality) summarizing the side-information about it at a particular time point. The reward of each arm is assumed to be governed by a conjecture of unknown bandit parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ ($\|\boldsymbol{\theta}\|_2 \leq 1$ without loss of generality), which characterizes the environment. This can be specified by a reward mapping function $f_{\boldsymbol{\theta}}$: $r_{a_t} = f_{\boldsymbol{\theta}}(\mathbf{x}_{a_t})$. In a stationary environment, $\boldsymbol{\theta}$ is constant over time.

In a non-stationary environment, the reward distribution over arms varies over time because of the changes in the environment's bandit parameter $\boldsymbol{\theta}$. In this paper, we consider abrupt changes in the environment [10, 15, 16], i.e., the ground-truth parameter $\boldsymbol{\theta}$ changes arbitrarily at arbitrary time, but remains constant between any two consecutive change points:

$$\underbrace{r_0, r_1, \cdots, r_{t_{c_1}-1}}_{\text{distribute by } f_{\boldsymbol{\theta}_{c_0}}}, \underbrace{r_{t_{c_1}}, r_{t_{c_1}+1}, \cdots, r_{t_{c_2}-1}}_{\text{distribute by } f_{\boldsymbol{\theta}_{c_1}}}, \cdots \underbrace{, r_{t_{c_\Gamma}}, r_{t_{c_\Gamma}+1}, \cdots, r_T}_{\text{distribute by } f_{\boldsymbol{\theta}_{c_{\Gamma-1}}}}$$

where the change points $\{t_{c_j}\}_{j=1}^{\Gamma_T-1}$ of the underlying reward distribution and the corresponding bandit parameters $\{\boldsymbol{\theta}_{c_j}\}_{j=0}^{\Gamma_T-1}$ are

unknown to the learner. We only assume there are at most $\Gamma_T - 1$ change points in the environment up to time T, with $\Gamma_T \ll T$.

To simplify the discussion, linear structure in $f_{\theta_u}(\mathbf{x}_{a_t})$ is postulated, but it can be readily extended to more complicated dependency structures, such as generalized linear models [9], without changing the design of our algorithm. Specifically, we have,

$$r_t = f_{\theta_t}(\mathbf{x}_{a_t}) = \mathbf{x}_{a_t}^{\mathsf{T}} \boldsymbol{\theta}_t^* + \eta_t \tag{1}$$

in which η_t is Gaussian noise drawn from $N(0, \sigma^2)$, and the superscript $*$ in $\boldsymbol{\theta}_t^*$ means it is the ground-truth bandit parameter in the environment. In addition, we impose the following assumption about the non-stationary environment, which guarantees the detectability of the changes, and reflects our insight how to detect them on the fly,

Assumption 1. *For any two consecutive change points t_{c_j} and $t_{c_{j+1}}$ in the environment, there exists $\Delta_j > 0$, such that when $t \geq t_{c_{j+1}}$ at least ρ $(0 < \rho \leq 1)$ portion of all the arms satisfy,*

$$|\mathbf{x}_{a_t}^{\mathsf{T}} \boldsymbol{\theta}_{t_{c_{j+1}}}^* - \mathbf{x}_{a_t}^{\mathsf{T}} \boldsymbol{\theta}_{t_{c_j}}^*| > \Delta_j \tag{2}$$

Remark 1. *The above assumption is general and mild to satisfy in many practical scenarios, since it only requires a portion of the arms to have recognizable change in their expected rewards. For example, a user may change his/her preference in sports news but not in political news. The arms that do not satisfy Eq* (2) *can be considered as having small deviations in the generic reward assumption made in Eq* (1)*. We will later prove our bandit solution remains its regret scaling in the presence of such small deviation.*

3.2 Dynamic Linear UCB

Based on the above assumption about a non-stationary environment, in any stationary period between two consecutive change points, the reward estimation error of a contextual bandit model trained on the observations collected from that period should be bounded with a high probability [1, 6]. Otherwise, the model's consistent wrong predictions can only come from the change of environment. Based on this insight, we can evaluate whether the stationary assumption holds by monitoring a bandit model's reward prediction quality over time. To reduce variance in the prediction error from one bandit model, we ensemble a set of models, by creating and abandoning them on the fly.

Specifically, we propose a hierarchical bandit algorithm, in which a master multi-armed bandit model operates over a set of slave contextual bandit models to interact with the changing environment. The master model monitors the slave models' reward estimation error over time, which is referred to as 'badness' in this paper, to evaluate whether a slave model is admissible for the current environment. Based on the estimated 'badness' of each slave model, the master model dynamically discards *out-of-date* slave models or creates new ones. At each round t, the master model selects a slave model with the smallest lower confidence bound (LCB) of 'badness' to interact with the environment, i.e., the most promising slave model. The obtained observation $(\mathbf{x}_{a_t}, r_{a_t})$ is shared across all admissible slave models to update their model parameters. The process is illustrated in Figure 1.

Any contextual bandit algorithm [9, 18, 20, 25] can serve as our slave model. Due to the simplified linear reward assumption made in Eq (1), we choose LinUCB [18] for the purpose in this paper; but

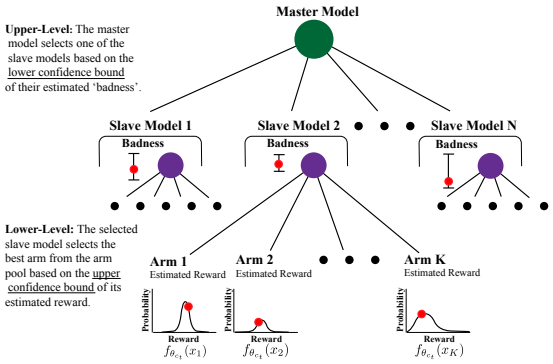


Figure 1: Illustration of dLinUCB. The master bandit model maintains the 'badness' estimation of slave models over time to detect changes in the environment. At each round, the most promising slave model is chosen to interact with the environment; and the acquired feedback is shared across all admissible slave models for model update.

our proposed algorithm can be readily adapted to any other choices of the slave model. This claim is also supported by our later regret analysis. As a result, we name our algorithm as Dynamic Linear Bandit with Upper Confidence Bound, or dLinUCB in short.

In the following, we first briefly describe our chosen slave model LinUCB. Then we formally define the concept of 'badness', based on which we design the strategy for creating and discarding slave bandit models. Lastly, we explain how dLinUCB selects the most promising slave model from the admissible model set. The detailed description of dLinUCB is provided in Algorithm 1.

Slave bandit model: LinUCB. Each slave LinUCB model maintains all historical observations that the master model has assigned to it. Based on the assigned observations, a slave model m gets an estimate of user preference $\hat{\boldsymbol{\theta}}_t(m) = \mathbf{A}_t^{-1}(m)\mathbf{b}_t(m)$ [18], in which $\mathbf{A}_t(m) = \lambda\mathbf{I} + \sum_{i \in \mathcal{I}_{m,t}} \mathbf{x}_{a_i}\mathbf{x}_{a_i}^{\mathsf{T}}$, $\mathbf{I}$ is a $d \times d$ identity matrix, λ is the coefficient for $L2$ regularization; $\mathbf{b}_t(m) = \sum_{i \in \mathcal{I}_{m,t}} \mathbf{x}_{a_i} r_{a_i}$, and $\mathcal{I}_{m,t}$ is an index set recording when the observations are assigned to the slave model m up to time t. According to [1], with a high probability $1 - \delta_1$ the expected reward estimation error of model m is upper bounded: $|\hat{r}_{a_t}(m) - \mathbb{E}[r_{a_t}]| \leq B_t(m, a)$, in which $B_t(m, a) = \left(\sigma^2\sqrt{d\ln(1 + \frac{|\mathcal{I}_{m,t}|}{\lambda\delta_1})} + \sqrt{\lambda}\right)\|\mathbf{x}_a\|_{\mathbf{A}_t^{-1}(m)}$. Based on the upper confidence bound principle [3], a slave model m takes an action using the following arm selection strategy (i.e., line 6 in Algorithm 1):

$$a_t(m) = \arg\max_{a \in \mathcal{A}} \left(\mathbf{x}_a^{\mathsf{T}}\hat{\boldsymbol{\theta}}_t(m) + B_t(m, a)\right) \tag{3}$$

Slave model creation and abandonment. For each slave bandit model m, we define a binary random variable $e_i(m)$ to indicate whether the slave model m's prediction error at time i exceeds its confidence bound,

$$e_i(m) := \mathbb{1}\left\{|\hat{r}_i(m) - r_i(m)| > B_i(m, a_i) + \epsilon\right\} \tag{4}$$

where $\epsilon = \sqrt{2}\sigma\mathrm{erf}^{-1}(\delta_1 - 1)$ and $\mathrm{erf}^{-1}(\cdot)$ is the inverse of Gauss error function. ϵ represents the high probability bound of Gaussian noise in the received feedback.

According to Eq (7) in Theorem 3.1, if the environment stays stationary since the slave model m has been created, we have $\mathbb{P}(e_i(m) = 1) \leq \delta_1$, where $\delta_1 \in (0, 1)$ is a hyper-parameter in

Algorithm 1 Dynamic Linear UCB (dLinUCB)

1: **Inputs:** $\lambda > 0$, $\tau > 0$, $\delta_1, \delta_2 \in (0,1)$, $\tilde{\delta}_1 \in [0, \delta_1]$
2: **Initialize:** Maintain a set of slave models $\mathcal{M}_t$ with $\mathcal{M}_1 = \{m_1\}$, initialize m_1: $\mathbf{A}_1(m_1) = \lambda\mathbf{I}$, $\mathbf{b}_1(m_1) = \mathbf{0}$, $\hat{\boldsymbol{\theta}}_1(m_1) = \mathbf{0}$; and initialize the 'badness' statistics of it: $\hat{e}_1(m_1) = 0$, $d_1(m_1) = 0$
3: **for** $t = 1$ to T **do**
4: Choose a slave model from the active slave model set $\tilde{m}_t = \arg\min_{m \in \mathcal{M}_t} \left(\hat{e}_t(m) - \sqrt{\ln \tau} \times d_t(m)\right)$
5: Observe candidate arm pool $\mathcal{A}_t$, with $\mathbf{x}_a \in \mathbb{R}^d$ for $\forall a \in \mathcal{A}_t$
6: Take action $a_t = \arg\max_{a \in \mathcal{A}_t} \left(\mathbf{x}_a^\top \hat{\boldsymbol{\theta}}_t(\tilde{m}_t) + B_t(\tilde{m}_t, a)\right)$, in which $B_t(\tilde{m}_t, a)$ is defined in Eq (3)
7: Observe payoff r_{a_t}
8: Set CreatNewFlag = True
9: **for** $m \in \mathcal{M}_t$ **do**
10: $e_t(m) = \mathbb{1}\{|\hat{r}_t(m) - r_t| > B_t(m, a) + \epsilon\}$, where $\hat{r}_t(m) = \mathbf{x}_{a_t}^\top \hat{\boldsymbol{\theta}}_t(m)$ and $\epsilon = \sqrt{2}\sigma \text{erf}^{-1}(\delta_1 - 1)$
11: **if** $e_t(m) = 0$ **then**
12: Update slave model: $\mathbf{A}_{t+1}(m) = \mathbf{A}_t(m) + \mathbf{x}_{a_t}\mathbf{x}_{a_t}^\top$, $\mathbf{b}_{t+1}(m) = \mathbf{b}_t(m) + \mathbf{x}_{a_t} r_t$, $\hat{\boldsymbol{\theta}}_{t+1} = \mathbf{A}_{t+1}^{-1}(m)\mathbf{b}_{t+1}(m)$
13: **end if**
14: $\tilde{\tau}(m) = \min\{t - t_m, \tau\}$, where t_m is when m was created
15: Update 'badness' $\hat{e}_t(m) = \frac{\sum_{i=t-\tilde{\tau}}^{t} e_i(m)}{\tilde{\tau}(m)}$, $d_t(m) = \sqrt{\frac{\ln 1/\delta_2}{2\tilde{\tau}(m)}}$
16: **if** $\hat{e}_t(m) < \tilde{\delta}_1 + d_t(m)$ **then**
17: Set CreatNewFlag = False
18: **else if** $\hat{e}_t(m) \geq \delta_1 + d_t(m)$ **then**
19: Discard slave model m: $M_{t+1} = M_t - m$
20: **end if**
21: **end for**
22: **if** CreateNewFlag or $\mathcal{M}_t = \emptyset$ **then**
23: Create a new slave model m_t: $\mathcal{M}_{t+1} = \mathcal{M}_t + m_t$
24: Initialize m_t: $\mathbf{A}_t(m_t) = \lambda\mathbf{I}$, $\mathbf{b}_t(m_t) = \mathbf{0}$, $\hat{\boldsymbol{\theta}}_t(m_t) = \mathbf{0}$
25: Initialize 'badness' statistics of m_t: $\hat{e}_t(m_t) = 0, d_t(m_t) = 0$
26: **end if**
27: **end for**

$B_i(m, a)$. Therefore, if we observe a sequence of consistent prediction errors from the slave model m, it strongly suggests a change of environment, so that this slave model should be abandoned from the admissible set. Moreover, we introduce a size-τ sliding window to only accumulate the most recent observations when estimating the expected error in slave model m. The benefit of sliding window design will be discussed with more details later in Section 3.3.

We define $\hat{e}_t(m) := \frac{\sum_{i=t-\tilde{\tau}(m)}^{t} e_i(m)}{\tilde{\tau}(m)}$, which estimates the 'badness' of slave model m within the most recent period $\tilde{\tau}$ to time t, i.e., $\tilde{\tau}(m) = \min\{t - t_m, \tau\}$, in which t_m is when model m was created. Combining the concentration inequality in Theorem 7.2 (provided in the appendix), we have the assertion that if in the period $[t - \tilde{\tau}(m), t]$ the stationary hypothesis is true, for any given $\delta_1 \in (0,1)$ and $\delta_2 \in (0,1)$, with a probability at least $1 - \delta_2$, the expected 'badness' of slave model m satisfies,

$$\hat{e}_t(m) \leq \mathbb{E}[e_t(m)] + \sqrt{\frac{\ln(1/\delta_2)}{2\tilde{\tau}(m)}} \leq \delta_1 + \sqrt{\frac{\ln(1/\delta_2)}{2\tilde{\tau}(m)}} \tag{5}$$

Eq (5) provides a tight bound to detect changes in the environment. If the environment is unchanged, within a sliding window the estimation error made by an up-to-date slave model should not exceed the right-hand side of Eq (5) with a high probability. Otherwise, the stationary hypothesis has to be rejected and thus the slave model m should be discarded. Accordingly, if none of the slave models in the admissible bandit set satisfy this condition, a new slave bandit model should be created for this new environment. Specifically, the master bandit model controls the slave model creation and abandonment in the following way.

• *Model abandonment*: when the slave model m's estimated 'badness' exceeds its upper confidence bound defined in Eq (5), i.e., $\hat{e}_t(m) > \delta_1 + \sqrt{\frac{\ln(1/\delta_2)}{2\tilde{\tau}(m)}}$, it will be discarded and removed from the admissible slave model set. This corresponds to line 18-20 in Algorithm 1.

• *Model creation*: When no slave model's estimated 'badness' is within its expected confidence bound, i.e., no slave model satisfies $\hat{e}_t(m) \leq \tilde{\delta}_1 + \sqrt{\frac{\ln(1/\delta_2)}{2\tilde{\tau}(m)}}$, a new slave model will be created. $\tilde{\delta}_1 \in [0, \delta_1]$ is a parameter to control the sensitivity of dLinUCB, which affects the number of maintained slave models. When $\tilde{\delta}_1 = \delta_1$, the threshold of creating and abandoning a slave model matches and the algorithm only maintains one admissible slave model. When $\tilde{\delta}_1 < \delta_1$ multiple slave models will be maintained. The intuition is that an environment change is very likely to happen when all active slave models face a high risk of being out-of-date (although they have not been abandoned yet). This corresponds to line 8, 16-17, and 22-26 in Algorithm 1.

Slave model selection and update. At each round, the master bandit model selects one active slave bandit model to interact with the environment, and updates all active slave models with the acquired feedback accordingly. As we mentioned before, with the model abandonment mechanism every active slave model is guaranteed to be admissible for taking acceptable actions; but they are associated with different levels of risk of being out of date. A well-designed model selection strategy can further reduce the overall regret, by minimizing this risk. Intuitively, when facing a changing environment, one should prefer a slave model with the lowest empirical error in the most recent period.

The uncertainty in assessing each slave model's 'badness' introduces another explore-exploit dilemma, when choosing the active slave models. Essentially, we prefer a slave model of lower 'badness' with a higher confidence. We realize this criterion by selecting a slave model according to its Lower Confidence Bound (LCB) of the estimated 'badness.' This corresponds to line 4 in Algorithm 1.

Once the feedback $(\mathbf{x}_{a_t}, r_t)$ is obtained from the environment on the selected arm a_t, the master algorithm can not only update the selected slave model but also all other active ones for both of their 'badness' estimation and model parameters (line 11-13 and line 15 in Algorithm 1 accordingly). This would reduce the sample complexity in each slave model's estimation. However, at this stage, it is important to differentiate those "about to be out-of-date" models from the "up-to-date" ones, as any unnecessary model update blurs the boundary between them. As a result, we only update the *perfect* slave models, i.e., those whose 'badness' is still zero at this round of interaction; and later we will prove this updating strategy is helpful to decrease the chance of late detection.

3.3 Regret Analysis

In this section, we provide a detailed regret analysis of our proposed dLinUCB algorithm. We focus on the accumulated pseudo regret, which is formally defined as,

$$\mathbf{R}(T) = \sum_{t=1}^{T} \left(\mathbb{E}[r_{a_t^*}] - \mathbb{E}[r_{a_t}]\right) \tag{6}$$

where a_t^* is the best arm to select according to the oracle of this problem, and a_t is the arm selected by the algorithm to be evaluated.

It is easy to prove that if a bandit algorithm does not model the change of environment, it would suffer from a linearly increasing regret: An optimal arm in the previous stationary period may become sub-optimal after the change; but the algorithm that does not model environment change will constantly choose this sub-optimal arm until its estimated reward falls behind the other arms'. This leads to a linearly increasing regret in each new stationary period.

Next, we first characterize the confidence bound of reward estimation in a linear bandit model in Theorem 3.1. Then we prove the upper regret bound of two variants of our dLinUCB algorithm in Theorem 3.2 and Theorem 3.5. More detailed proofs are provided in the appendix.

Theorem 3.1. *For a linear bandit model m specified in Algorithm 1, if the underlying environment is stationary, for any $\delta_1 \in (0,1)$ we have the following inequality with probability at least $1-\delta_1$,*

$$|\hat{r}_t(m) - r_t| \leq B_t(m,a) + \epsilon \tag{7}$$

where $B_t(m,a) = \alpha_t \|\mathbf{x}_{a_t}\|_{\mathbf{A}_{t-1}^{-1}}$ with $\alpha_t = \left(\sigma^2\sqrt{d\ln(1+\frac{|\mathcal{I}_t(m)|}{\lambda\delta_1}}\right) + \sqrt{\lambda}\right)$, $\epsilon = \sqrt{2}\sigma erf^{-1}(\delta_1 - 1)$, σ is the standard deviation of the Gaussian noise in reward feedback, and erf$(\cdot)$ is the Gauss error function.

Denote $R_{\text{Lin}}(S)$ as the upper regret bound of a linear bandit model within a stationary period S. Based on Theorem 3.1, one can prove that $R_{\text{Lin}}(S) \leq \sqrt{dS\log(\lambda+\frac{S}{d})}\left(\sigma^2\sqrt{d\log(1+\frac{S}{\lambda\delta_1}}) + \sqrt{\lambda}\right)$ [1]. In the following, we provide an upper regret bound analysis for the basic version of dLinUCB, in which the size of the admissible slave models is restricted to one (i.e., by setting $\tilde{\delta_1} = \delta_1$).

Theorem 3.2. *When Assumption 1 is satisfied with $\Delta \geq 2\sqrt{\lambda}+2\epsilon$, if δ_1 and τ in Algorithm 1 are set according to Lemma 3.4, and δ_2 is set to $\delta_2 \leq \frac{1}{2S_{max}}$, with probability at least $(1-\delta_1)(1-\delta_2)(1-\frac{\delta_2}{1-\delta_2})$, the accumulated regret of dLinUCB satisfies,*

$$\boldsymbol{R}(T) \leq 2\Gamma_T R_{Lin}(S_{max}) + \Gamma_T(\tau + \frac{4}{1-\delta_2}) \tag{8}$$

where S_{max} is the length of the longest stationary period up to T.

Proof. **Step 1:** If change points can be perfectly detected, the regret of dLinUCB can be bounded by $\sum_{j=0}^{\Gamma_T-1} R_{\text{Lin}}(S_{c_j})$. However, additional regret may accumulate if early or late detection happens. In the following two steps, we will bound the possible additional regret from early detection, denoted as R_{early}, and that from late detection, denoted as R_{late}.

Step 2: Define k_{c_j} as the number of early detection within this stationary period $[t_{c_j}, t_{c_{j+1}}]$, with $S_{c_j} = t_{c_{j+1}} - t_{c_j}$. Define p_e as the probability of early detection in the stationary period, we have $\mathbb{P}(k_{c_j} = k) = \binom{S_{c_j}}{k} p_e^k (1-p_e)^{S_{c_j}-k}$. According to Lemma 3.3, we have $p_e \leq \delta_2$. Combining the property of binomial distribution $B(S_{c_j}, p_e)$ and Chebyshev's concentration inequality, we have $k_{c_j} \leq 2S_{c_j}\delta_2$ with probability at least $1-\frac{1-\delta_2}{2S_{c_j}\delta_2}$. Hence, with a probability $(1-\delta_3)\times(1-\delta_1)^{k_{S_{\max}}}$, we have $R_{\text{early}} \leq \sum_{j=1}^{\Gamma_T} k_{c_j} R_{\text{Lin}}(\frac{s_{c_j}}{k_{c_j}}) \leq \sum_{j=1}^{\Gamma_T} 2S_{c_j}\delta_2 R_{\text{Lin}}(\frac{1}{2\delta_2}) \leq 2\delta_2\Gamma_T S_T R_{\text{Lin}}(\frac{1}{2\delta_2})$. Considering the calculation of R_{Lin}, when $\delta_2 \leq \frac{1}{2S_{\max}}$ we have $R_{\text{early}} \leq \Gamma_T R_{\text{Lin}}(S_{\max})$ with a probability at least $(1-\delta_2)(1-\delta_1)$. This upper bounds the additional regret from any possible early detection, and maintains it in the same order as the slave model's.

Step 3. Define $\tilde{k}_{c_j}$ as the number of interactions where the environment has changed (comparing to $\boldsymbol{\theta}_{c_j}^*$) but the change is not detected by the algorithm. The additional regret from this late detection can be bounded by $2\tilde{k}_{c_j}$ (i.e., the maximum regret in each round of interaction). Define p_d as the probability of detection after the change happens, we have $\mathbb{P}(\tilde{k}_{c_j} = k) = (1-p_d)^{k-1}p_d$, i.e., a Geometric distribution. According to Lemma 3.4, $p_d \geq 1-\delta_2$. Based on the property of Geometric distribution $G(p_d)$ and Chebyshev's inequality, we have $\tilde{k}_{c_j} \leq \frac{2}{1-\delta_2}$ with probability $1-\frac{\delta_2}{1-\delta_2}$. If we consider the case where the change point locates inside the sliding window τ, we may have at most another τ delay after each change point. Therefore, the additional regret from late detection can be bounded by $R_{\text{late}} \leq \Gamma_T\left(\tau + \frac{4}{1-\delta_2}\right)$, which is not directly related to the length of any stationary period.

Combining the above three steps concludes the proof. □

Lemma 3.3 (Bound the probability of early detection). *For $\delta_2 \in (0,1)$ and any slave model in Algorithm 1,*

$$p_e = \mathbb{P}[\hat{\epsilon}_t(m) > \delta_1 + d_t(m) | \textit{stationary in past } \tilde{\tau}(m) \textit{ rounds}] \leq \delta_2.$$

The intuition behind Lemma 3.3 is that when the environment is stationary, the 'badness' of a slave model m should be small and bounded according to Eq (5).

Lemma 3.4 (Bound the probability of late detection). *When the magnitude of change in the environment in Assumption 1 satisfies $\Delta > 2\sqrt{\lambda}+2\epsilon$, and the shortest stationary period length S_{min} satisfies $S_{min} > \frac{\sqrt{\lambda}}{2\rho}(\Delta - 2\sqrt{\lambda} - 2\epsilon)$, for any $\delta_2 \in (0,1)$, if δ_1 and τ in Algorithm 1 are set to $\delta_1 \leq 1 - \frac{1}{\rho}\left(1 - \frac{\sqrt{\lambda}}{2S_{min}\rho}(\Delta - 2\sqrt{\lambda} - 2\epsilon)\right)$ and $\tau \geq \frac{2\ln\frac{2}{\delta_2}}{(\rho(1-\delta_1)-\delta_1)^2}$, for any slave model m in Algorithm 1, we have,*

$$p_d = \mathbb{P}\left(\hat{\epsilon}_t(m) > \delta_1 + d_t(m) | \textit{ changed within past } \tilde{\tau}(m)\right) \geq 1-\delta_2.$$

The intuition behind Lemma 3.4 is that when the environment has changed, with a high probability that Eq (7) will not be satisfied in an out-of-date slave model. It means that we will accumulate larger badness from this slave model. In both Lemma 3.3 and 3.4, δ_2 is a parameter controlling the confidence of the 'badenss' estimation in Chernoff Bound; and therefore an input to the algorithm.

Remark 2 (How the environment assumption affects dLinUCB). *1. The magnitude of environment change Δ affects whether a change is detectable by our algorithm. However, we need to emphasize that when Δ is very small, the additional regret from re-using an out-of-date slave model is also small. In this case, a similar scale of regret bound can still be achieved, which will be briefly proved in Appendix and empirically studied in* Section 4.1. *2. We require the*

shortest stationary period length $S_{min} > \max\{\frac{\sqrt{\lambda}}{2\rho}(\Delta - 2\sqrt{\lambda} - 2\epsilon), \tau\}$, *which guarantees there are enough observations accumulated in a slave model to make an informed model selection. 3. The portion of changed arms* ρ *will affect the probability of achieving our derived regret bound, as we require* $\delta_1 \leq 1 - \frac{1}{\rho}\big(1 - \frac{\sqrt{\lambda}}{2S_{min}\rho}(\Delta - 2\sqrt{\lambda} - 2\epsilon)\big)$. ρ *also interacts with* S_{min} *and* τ*: when* ρ *is small, more observations are needed for a slave model to detect the changes. The effect of* ρ *and* S_{min} *will also be studied in our empirical evaluations.*

Theorem 3.2 indicates with our model update and abandonment mechanism, each slave model in dLinUCB is 'admissible' in terms of upper regret bound. In the following, we further prove that maintaining multiple slave models and selecting them according to their LCB of 'badness' can further improve the regret bound.

Theorem 3.5. *Under the same condition as specified in Theorem 3.2, with probability at least* $(1-\delta_1)(1-\delta_2)(1-\frac{\delta_2}{1-\delta_2})$, *the expected accumulated regret of dLinUCB up to time* T *can be bounded by,*

$$\boldsymbol{R}(T) \leq \Big(\sum_{j=0}^{\Gamma_T - 1} R_{Lin}(S_{c_j}) + 2\Gamma_T(\tau + \frac{4}{1-\delta_2}) \tag{9}$$
$$+ \sum_{j=0}^{\Gamma_T - 1} \Big(8 \sum_{m \in \mathcal{M}, m \neq m^*_{c_j}} \frac{\ln S_{c_j}}{g_{m, m^*_{c_j}}} + (1 + \frac{\pi^2}{3}) g_{m, m^*_{c_j}}\Big)$$

in which $m^*_{c_j}$ *is the best slave model among all the active ones in the stationary period* $[t_{c_j}, t_{c_{j+1}}]$ *according to the oracle, and* $g_{m, m^*_{c_j}}$ *is difference between the accumulated expected reward from the selected model* m *and that from* $m^*_{c_j}$ *in the period* $[t_{c_j}, t_{c_{j+1}}]$.

Proof. Define the optimal expected cumulative reward in the stationary period $[t_{c_j}, t_{c_j+1}]$ according to the oracle as $G^*(S_{c_j})$ and the expected accumulative reward in dLinUCB as $G(S_{c_j})$. $G_{m^*_{c_j}}(S_{c_j})$ is the expected cumulative reward from $m^*_{c_j}$. The accumulated regret of dLinUCB can be written as,

$$\mathbf{R}(T) = \sum_{j=0}^{\Gamma_T - 1} \big(G^*(S_{c_j}) - G_{m^*_{c_j}}(S_{c_j})\big) + \big(G_{m^*_{c_j}}(S_{c_j}) - G(S_{c_j})\big) \tag{10}$$

The first term of Eq(10) can be bounded based on Theorem 3.2. Define $\tilde{N}_{c_j}(m)$ as the number of times a slave model m is selected when it is not the best in $[t_{c_j}, t_{c_j+1}]$: $\tilde{N}_{c_j}(m) = \sum_{t=t_{c_j}}^{t_{c_{j+1}}} \mathbb{1}\{\tilde{m}_t = m, m^*_{c_j} \neq m\}$, we have $G_{m^*_{c_j}}(S_{c_j}) - G(S_{c_j}) \leq \sum_{m \in \mathcal{M}} g_{m, m^*_{c_j}} \mathbb{E}[\tilde{N}_{c_j}(m)]$. In Lemma 3.6, we provide the bound of $\mathbb{E}[\tilde{N}_{c_j}(m)]$. Substituting the above conclusions into Eq (10) finishes the proof. □

Lemma 3.6. *The model selection strategy in Algorithm 1 guarantees,*

$$\mathbb{E}[\tilde{N}_{c_j}(m)] \leq \frac{8 \ln S_{c_j}}{g^2_{m, m^*_{c_j}}} + 1 + \frac{\pi^2}{3} \tag{11}$$

Remark 3 (regret comparison of dLinUCB with one slave model and multiple slave models). *By maintaining multiple admissible slave models and selecting one according to the LCB of 'badness' when interacting with the environment, dLinUCB achieves a regret reduction in the first part of Eq* (9). *Although there is additional regret introduced by switching between the best model* m^* *and the chosen model* $\tilde{m}$*, this added regret increases much slower than that resulted from any slave model (i.e.,* $\ln T$ *v.s.,* $R_{Lin}(T)$*); and thus maintaining multiple slave models is always beneficial. Besides, the order of upper regret bound of dLinUCB in both cases is* $O(\Gamma_T \sqrt{S_{max}} \log S_{max})$, *which is the best upper regret bound a bandit algorithm can achieve in such a non-stationary environment [10], and it matches the lower bound up to a* $\Gamma_T \log S_{max}$ *factor.*

Remark 4 (Generalization of dLinUCB). *Our theoretical analysis confirms that any contextual bandit algorithm can be used as the slave model in dLinUCB, as long as the its reward estimation error is bounded with a high probability, which corresponds* $B_t(m, a)$ *in Eq* (7). *The overall regret of dLinUCB will only be a factor of the actual number of changes in the environment, which is arguably inevitable without further assumptions about the environment.*

4 EVALUATIONS

We performed extensive empirical evaluations of dLinUCB against several related baseline algorithms, including: 1) the state-of-the-art contextual bandit algorithm LinUCB [18]; 2) adaptive Thompson Sampling algorithm [15] (named as adTS) which has a change detection module; 3) windowed mean-shift detection algorithm [26] (named as WMDUCB1), which is a UCB1-type algorithm with a change detection module ; and 4) Meta-Bandit algorithm [16], which switches between two UCB1 models.

4.1 Experiments on synthetic datasets

In simulation, we generate a size-K ($K = 1000$) arm pool $\mathcal{A}$, in which each arm a is associated with a d-dimensional feature vector $\mathbf{x}_a \in \mathbb{R}^d$ with $\|\mathbf{x}_a\|_2 \leq 1$. Similarly, we create the ground-truth bandit parameter $\boldsymbol{\theta}^* \in \mathbb{R}^d$ with $\|\boldsymbol{\theta}^*\|_2 \leq 1$, which is not disclosed to the learners. Each dimension of $\mathbf{x}_a$ and $\boldsymbol{\theta}^*$ is drawn from a uniform distribution $U(0, 1)$. At each round t, only a subset of arms in $\mathcal{A}$ are disclosed to the learner for selection, e.g., randomly sample 10 arms from $\mathcal{A}$ without replacement. The ground-truth reward r_a is corrupted by Gaussian noise $\eta \sim N(0, \sigma^2)$ before being fed back to the learner. The standard deviation of Gaussian noise σ is set to 0.05 by default. To make the comparison fair, at each round t, the same set of arms are presented to all the algorithms being evaluated. To simulate an abruptly changing environment, after every S rounds, we randomize $\boldsymbol{\theta}^*$ with respect to the constraint $|\mathbf{x}_a^\intercal \boldsymbol{\theta}^*_{t_{c_j}} - \mathbf{x}_a^\intercal \boldsymbol{\theta}^*_{t_{c_{j+1}}}| > \Delta_j$ for ρ proportion of arms in $\mathcal{A}$. We set λ to 0.1, S to 800 and Δ to 0.9 by default.

Under this simulation setting, all algorithms are executed to 5000 iterations and the parameter τ in dLinUCB is set to 200. Accumulated regret defined in Eq (6) is used to evaluate different algorithms and is reported in Figure 2. The bad performance of LinUCB illustrates the necessity of modeling the non-stationarity of the environment – its regret only converges in the first stationary period, and it suffers from an almost linearly increasing regret, which is expected according to our theoretical analysis in Section 3.3. adTS is able to detect and react to the changes in the environment, but it is slow in doing so and therefore suffers from a linear regret at the beginning of each stationary period before converging. dLinUCB, on the other hand, can quickly identify the changes and create corresponding slave models to capture the new reward distributions, which makes the regret of dLinUCB converge much faster in each detected stationary period. In Figure 2 we use the black

Table 1: Accumulated regret with different noise level σ, environment change Δ and stationary period length S.

(σ, Δ, S)	(0.1, 0.9, 800)	(0.05, 0.9, 800)	(0.01, 0.9, 800)	(0.01, 0.5, 800)	(0.01, 0.1, 800)	(0.01, 0.9, 400)
dLinUCB	**87.46 ± 3.61**	**65.94± 2.30**	**54.07± 3.95**	**44.94 ± 2.90**	**46.12 ± 4.63**	**111.72 ± 4.87**
adTS	360.75± 39.59	249.63 ± 27.26	207.95± 22.28	189.07±18.39	177.55.±20.36	412.55 ± 14.53
LinUCB	436.84 ± 40.23	386.10±21.88	347.19± 14.95	264.87± 21.53	226.87± 32.15	405.82 ± 33.38
Meta-Bandit	1822.31± 80.67	1340.01±29.94	1354.03 ± 22.29	1329.51 ± 18.93	1402.63 ± 24.85	1388.81 ± 115.91
WMDUCB1	2219.36± 142.16	1652.99± 21.33	1635.35 ± 73.96	1464.11 ± 89.16	1506.55 ± 41.52	1691.75 ± 48.09

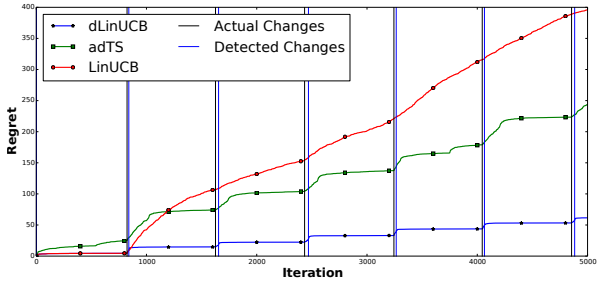


Figure 2: Results from simulation.

and blue vertical lines to indicate the actual change points and the detected ones by dLinUCB respectively. It is clear that dLinUCB detects the changes almost immediately every time. WMDUCB1 and Meta-Bandit are also compared, but since they are context-free bandits, they performed much worse than the above contextual bandits. To improve visibility of the result, we exclude them from Figure 2 and instead report their performance in Table 1.

As proved in our regret analysis, dLinUCB's performance depends the magnitude of change Δ between two consecutive stationary periods, the Gaussian noise σ in the feedback, and the length S of stationary period. In order to investigate how these factors affect dLinUCB, we varied these three factors in simulation. We ran all the algorithms for 10 times and report the mean and standard deviation of obtained regret in Table 1. In all of our environment settings, dLinUCB consistently achieved the best performance against all baselines. In particular, we can notice that the length S of stationary period plays an important role in affecting dLinUCB's regret (and also in adTS). This is expected from our regret analysis: since T is fixed, a smaller S leads to a larger Γ_T, which linearly scales dLinUCB's regret in Eq (8) and (9). A smaller noise level σ leads to reduced regret in dLinUCB, as it makes the change detection easier. Last but not least, the magnitude of change Δ does not affect dLinUCB: when Δ is large, the change is easy to detect; when Δ is small, the difference between two consecutive reward distributions is small, and thus the added regret from an out-of-date slave model is also small. Again the context-free algorithms WMDUCB1 and Meta-Bandit performed much worse than those contextual bandit algorithms in all the experiments.

In addition, we also studied the effect of ρ in dLinUCB by varying ρ from 0.0 to 1.0. dLinUCB achieved the lowest regret when $\rho = 0$, since the environment becomes stationary. When $\rho > 0$: dLinUCB achieves the best regret (with regret of 54.07 ± 3.95) when $\rho = 1.0$, however as ρ becomes smaller the regret is not affected too much (with regret of 57.59 ± 3.44). These results further validate our theoretical regret analysis and unveil the nature of dLinUCB in a piecewise stationary environment.

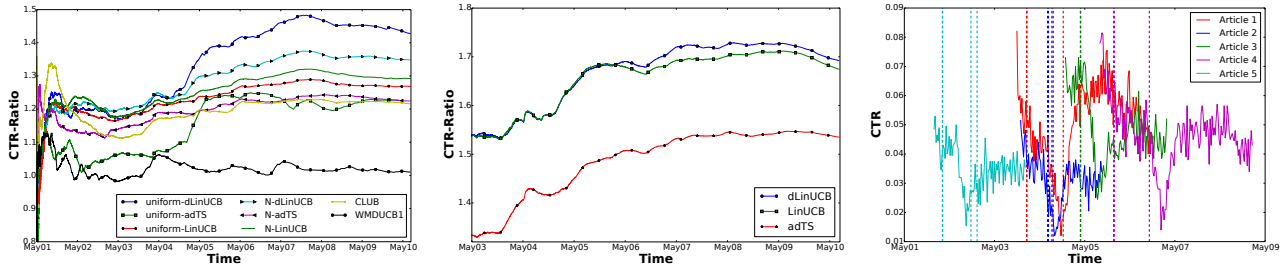
4.2 Experiments on Yahoo! Today Module

We compared all the algorithms on the large-scale clickstream dataset made available by the Yahoo Webscope program. This dataset contains 45,811,883 user visits to Yahoo Today Module in a ten-day period in May 2009. For each visit, both the user and each of the 10 candidate articles are associated with a feature vector of six dimensions (including a constant bias term) [18]. In the news recommendation problem, it is generally believed that users' interests on news articles change over time; and it is confirmed in this large-scale dataset by our quantitative analysis. To illustrate our observations, we randomly sampled 5 articles and reported their real-time click-through-rate (CTR) in Figure 3 (c), where each point is the average CTR over 2000 observations. Clearly, there are dramatic changes in those articles' popularity over time. For example, article 1's CTR kept decreasing after its debut, then increased in the next two days, and dropped eventually. Any recommendation algorithm failing to recognize such changes would suffer from a sub-optimal recommendation quality over time.

The unbiased offline evaluation protocol proposed in [19] is used to compare different algorithms. CTR is used as the performance metric of all bandit algorithms. Following the same evaluation principle used in [18], we normalized the resulting CTR from different algorithms by the corresponding logged random strategy's CTR. We tested two different settings on this dataset based on where to place the bandit model for reward estimation.

The first setting is to build bandit models for users, i.e., attaching $\boldsymbol{\theta}$ on the user side to learn users' preferences over articles. We included a non-personalized variant and a personalized variant of all the contextual bandit algorithms. In the non-personalized variant, the bandit parameters are shared across all users, and thus the detected changes are synchronized across users. We name the resulting algorithms as uniform-LinUCB, uniform-adTS, and uniform-dLinUCB. In the personalized variant, each individual user is associated with an independent bandit parameter $\boldsymbol{\theta}_u$, and the change is only about him/herself. Since this dataset does not provide user identities, we followed [25] to cluster users into N user groups and assume those in the same group share the same bandit parameter. We name the resulting algorithms as N-LinUCB, N-adTS and N-dLinUCB. To make the comparison more competitive, we also include a recently introduced collaborative bandit algorithm CLUB [12], which combines collaborative filtering with bandit learning.

From Figure 3 (a), we can find that both the personalized and non-personalized variants of dLinUCB achieved significant improvement compared with all baselines. It is worth noticing that uniform-dLinUCB obtained around 50% improvement against uniform-LinUCB, 15% against N-LinUCB, and 25% against CLUB. Clearly assuming all the users share the same preference over the recommendation candidates is very restrictive, which is confirmed by the improved performance from the personalized version over the non-personalized



(a) Bandit models on the user side (b) Bandit models on the article side (c) Detected changes on sample articles

Figure 3: Performance comparison in Yahoo! Today Module.

version of all bandit algorithms. Because dLinUCB maintains multiple slave models concurrently, each slave model is able to cover preference in a subgroup of users, i.e., achieving personalization automatically. We looked into those created slave models and found they closely correlated with the similarity between user features in different groups created by [25], although such external grouping was not disclosed to uniform-dLinUCB. Although adTS and WMDUCB1 can also detect changes, its slow detection and reaction to the changes made it even worse than LinUCB on this dataset. MetaBandit is sensitive to its hyper-parameters and performed similarly to WMDUCB1, so that we excluded it from this comparison.

The second setting is to build bandit models for each article, i.e., attaching $\boldsymbol{\theta}$ on the article side to learn its popularity over time. Based on our quantitative analysis in the data set, we found that articles with short lifespans tend to have constant popularity. To emphasize the non-stationarity in this problem, we removed articles which existed less than 18 hours, and report the resulting performance in Figure 3 (b). We can find that dLinUCB performed comparably to LinUCB at the beginning, while the adTS baselines failed to recognize the popularity of those articles from the beginning, as the popularity of most articles did not change immediately. In the second half of this period, however, we can clearly realize the improvement from dLinUCB. To understand what kind of changes dLinUCB recognized in this data set, we plot the detected changes of five randomly selected articles in Figure 3 (c), in which dotted vertical lines are our detected change points on corresponding articles. As we can find in most articles the critical changes of ground-truth CTR can be accurately recognized. For example, article 1 and article 2 at around May 4, and article 3 at around May 5. Unfortunately, we do not have any detailed information about these articles to verify the changes; otherwise it would be interesting to correspond these detected changes to real-world events. In Figure 3 (b), we excluded the context-free bandit algorithms because they performed much worse and complicate the plots.

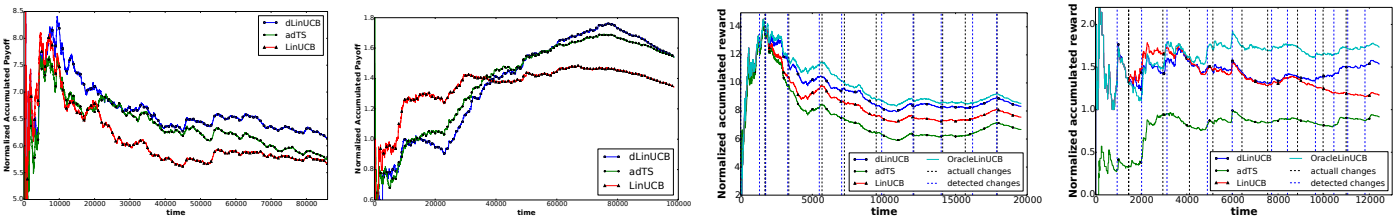
4.3 Experiments on LastFM & Delicious

The LastFM dataset is extracted from the music streaming service Last.fm, and the Delicious dataset is extracted from the social bookmark sharing service Delicious. They were made availalbe on the HetRec 2011 workshop. The LastFM dataset contains 1892 users and 17632 items (artists). We treat the 'listened artists' in each user as positive feedback. The Delicious dataset contains 1861 users and 69226 items (URLs). We treat the bookmarked URLs in each user as positive feedback. Following the settings in [5], we pre-processed these two datasets in order to fit them into the contextual bandit setting. Firstly, we used all tags associated with an item to create a TF-IDF feature vector to represent it. Then we used PCA to reduce the dimensionality of the feature vectors and retained the first 25 principle components to construct the context vectors, i.e., $d = 25$. We fixed the size of candidate arm pool to $K = 25$; for a particular user u, we randomly picked one item from his/her nonzero reward items, and randomly picked the other 24 from those zero reward items. We followed [16] to simulate a non-stationary environment: we ordered observations chronologically inside each user, and built a single hybrid user by merging different users. Hence, the boundary between two consecutive batches of observations from two original users is treated as the preference change of the hybrid user.

Normalized rewards on these two datasets are reported in Figure 4 (a) & (b). dLinUCB outperformed both LinUCB and adTS on LastFM. As Delicious is a much sparser dataset, both adTS and dLinUCB are worse than LinUCB at the beginning; but as more observations become available, they quickly catch up. Since the distribution of items in these two datasets are highly skewed [5], which makes the observations for each item very sparse, the context-free bandits performed very poorly on these two datasets. We therefore chose to exclude the context-free bandit algorithms from all the comparisons on these two datasets in our result report.

Each slave model created for this hybrid user can be understood as serving for a sub-population of users. We qualitatively studied those created slave models to investigate what kind of stationarity they have captured. On the LastFM dataset, each user is associated with a list of tags he/she gave to the artists. The tags are usually descriptive and reflect users' preference on music genres or artist styles. In each slave model, we use all the tags from the users being served by this model to generate a word cloud. Figure 5 are four representative groups identified on LastFM, which clearly correspond to four different music genres – rock music, metal music, pop music and hip-hop music. dLinUCB recognizes those meaningful clusters purely from user click feedback.

The way we simulate the non-stationary environment on these two datasets makes it possible for us to assess how well dLinUCB detects the changes. To ensure result visibility, we decide to report results obtained from user groups (otherwise there will be too many change points to plot). We first clustered all users in both of datasets into user groups according to their social network structure using spectral clustering [5]. Then we selected the top 10 user groups according to the number of observations to create the hybrid user. We created a semi-oracle algorithm named as OracleLinUCB, which knows where the boundary is in the environment and resets LinUCB at each change point. The normalized rewards from these two datasets are reported in Figure 4 (c) & (d), in which the vertical lines



(a) Normalized reward on LastFM (b) Normalized reward on Delicious (c) Cluster detection on LastFM (d) Cluster detection on Delicious

Figure 4: Performance comparison in LastFM & Delicious.

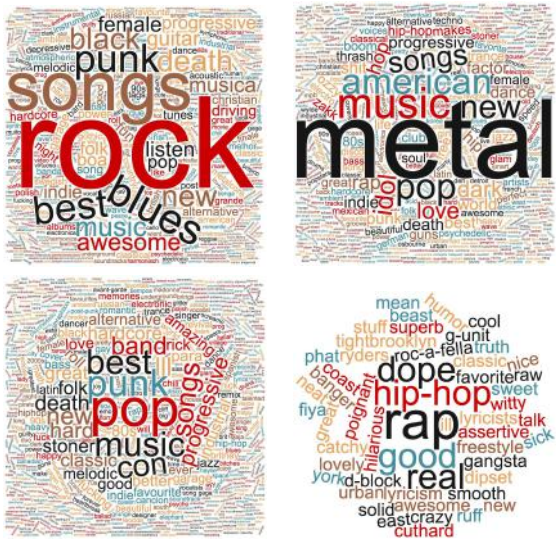


Figure 5: Word cloud of tags from four identified user groups in dLinUCB on LastFM dataset.

are the actual change points in the environment and the detected points by dLinUCB. Since OracleLinUCB knows where the change is ahead of time, its performance can be seen as optimal. On LastMF, the observations are denser per user group, so that dLinUCB can almost always correctly identify the changes and achieve quite close performance to this oracle. But on Delicious, the sparse observations make it much harder for change detection; and more early and late detection happened in dLinUCB.

5 CONCLUSIONS & FUTURE WORK

In this paper, we develop a contextual bandit model dLinUCB for a piecewise stationary environment, which is very common in many important real-world applications but insufficiently investigated in existing works. By maintaining multiple contextual bandit models and tracking their reward estimation quality over time, dLinUCB adaptively updates its strategy for interacting with a changing environment. We rigorously prove an $O(\Gamma_T\sqrt{S_T}\ln S_T)$ upper regret bound, which is arguably the tightest upper regret bound any algorithm can achieve in such an environment without further assumption about the environment. Extensive experimentation in simulation and three real-world datasets verified the effectiveness and the reliability of our proposed method.

As our future work, we are interested in extending dLinUCB to a continuously changing environment, such as Brownian motion, where reasonable approximation has to be made as a model becomes out of date right after it has been created. Right now, when serving for multiple users, dLinUCB treats them as identical or totally independent. As existing works have shed light on collaborative bandit learning [11, 24, 25], it is meaningful to study non-stationary bandits in a collaborative environment. Last but not least, currently the master bandit model in dLinUCB does not utilize the available context information for 'badness' estimation. It is necessary to incorporate such information to improve the change detection accuracy, which would lead to a further reduced regret.

6 ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. This work was supported in part by National Science Foundation Grant IIS-1553568 and IIS-1618948.

REFERENCES

[1] Yasin Abbasi-yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved Algorithms for Linear Stochastic Bandits. In *NIPS*. 2312–2320.
[2] Peter Auer. 2002. Using Confidence Bounds for Exploitation-Exploration Tradeoffs. *Journal of Machine Learning Research* 3 (2002), 397–422.
[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.* 47, 2-3 (May 2002), 235–256.
[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and Robert E. Schapire. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*. 322–331.
[5] Nicolò Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. 2013. A Gang of Bandits. *In Pro. NIPS* (2013).
[6] Wei Chu, Lihong Li, Lev Reyzin, and Robert E Schapire. 2011. Contextual bandits with linear payoff functions. In *AISTATS'11*. 208–214.
[7] Robert B Cialdini and Melanie R Trost. 1998. Social influence: Social norms, conformity and compliance. (1998).
[8] Fang Liu, Joohyun Lee, and Ness Shroff. 2018. A Change-Detection based Framework for Piecewise-stationary Multi-Armed Bandit Problem *(AAAI'18)*.
[9] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. 2010. Parametric bandits: The generalized linear case. In *NIPS*. 586–594.
[10] AurÃllien Garivier and Eric Moulines. On Upper-Confidence Bound Policies for Non-stationary Bandit Problems. In *arXiv preprint arXiv:0805.3415 (2008)*.
[11] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. 2017. On Context-Dependent Clustering of Bandits. In *ICML'17*. 1253–1262.
[12] Claudio Gentile, Shuai Li, and Giovanni Zappella. 2014. Online Clustering of Bandits. In *ICML'14*. 757–765.
[13] Avishek Ghosh, Sayak Ray Chowdhury, and Aditya Gopalan. 2017. Misspecified Linear Bandits. *CoRR* abs/1704.06880 (2017).
[14] John C Gittins. 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)* (1979), 148–177.
[15] Negar Hariri, Bamshad Mobasher, and Robin Burke. Adapting to User Preference Changes in Interactive Recommendation.
[16] Cedric Hartland, Sylvain Gelly, Nicolas Baskiotis, Olivier Teytaud, and Michele Sebag. 2006. Multi-armed Bandit, Dynamic Environments and Meta-Bandits. (Nov. 2006). https://hal.archives-ouvertes.fr/hal-00113668
[17] John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*. 817–824.
[18] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of 19th WWW*. ACM, 661–670.
[19] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of 4th WSDM*. ACM, 297–306.

[20] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. Collaborative Filtering Bandits. In *Proceedings of the 39th International ACM SIGIR*.
[21] Wei Li, Xuerui Wang, Ruofei Zhang, Ying Cui, Jianchang Mao, and Rong Jin. 2010. Exploitation and exploration in a performance based contextual advertising system. In *Proceedings of 16th SIGKDD*. ACM, 27–36.
[22] Haipeng Luo, Alekh Agarwal, and John Langford. 2017. Efficient Contextual Bandits in Non-stationary Worlds. *arXiv preprint arXiv:1708.01799* (2017).
[23] Alex Slivkins and Eli Upfal. 2008. Adapting to a Changing Environment: the Brownian Restless Bandits, In COLT08'. 343–354.
[24] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2017. Factorization Bandits for Interactive Recommendation.. In *AAAI*. 2695–2702.
[25] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. 2016. Contextual Bandits in a Collaborative Environment. In *Proceedings of the 39th International ACM SIGIR*. ACM, 529–538.
[26] Jia Yuan Yu and Shie Mannor. 2009. Piecewise-stationary Bandit Problems with Side Observations. In *Proceedings of the 26th ICML (ICML '09)*. 1177–1184.
[27] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of 26th ICML*. ACM, 1201–1208.

7 APPENDIX

7.1 Additional Theorems

If the training instances $\{(\mathbf{x}_i, r_i)\}_{i\in\mathcal{I}_{m,t}}$ in a linear bandit model come from multiple distributions/environments, we separate the training instances in $\mathcal{I}_{m,t}$ into two sets $\mathcal{H}_{m,t}$ and $\tilde{\mathcal{H}}_{m,t}$ so that instances from $\mathcal{H}_{m,t}$ are from the target stationary distribution, while instances in $\tilde{\mathcal{H}}_{m,t}$ are not. In this case, we provide the confidence bound for the reward estimation in Theorem 7.1.

Theorem 7.1 (LinUCB with contamination). *In LinUCB with a contaminated instance set $\tilde{\mathcal{H}}_{m,t}$, with probability at least $1-\delta_1$, we have $|\hat{r}_t(m) - \mathbb{E}[r_t]| \le \tilde{B}_t$, where $\tilde{B}_t(m,a) = \tilde{\alpha}_t \|\mathbf{x}_{a_t}\|_{\mathbf{A}_{t-1}^{-1}}$, $\tilde{\alpha}_t = \sigma^2\sqrt{d\ln(1+\frac{|\mathcal{I}_{m,t}|}{\lambda\delta_1})}+\sqrt{\lambda}+C_t(m)$, and $C_t = \sum_{i\in\tilde{\mathcal{H}}_{m,t}}(\mathbf{x}_{a_i}(\boldsymbol{\theta}_i^* - \boldsymbol{\theta}_{t_c}^*))$.*

Comparing $\tilde{B}_t(m,a)$ with $B_t(m,a)$, we can see that when the reward deviation of an arm (the $1-\rho$ portion of arms that do not satisfy Eq (1) in Assumption 1) is small with $(1-\rho)\Delta_{\text{small}} \le \left(\sigma^2\sqrt{d\ln(1+\frac{|\mathcal{I}_{m,t}|}{\lambda\delta_1}}\right))/\left(\tilde{\mathcal{H}}_{m,t}\right)$, the same confidence bound scaling can be achieved.

Theorem 7.2 (Chernoff Bound). *Let $Z_1, Z_2, ..., Z_n$ be random variables on $\mathbb{R}$ such that $a \le Z_i \le b$. Define $W_n = \sum_{i=1}^n Z_i$, for all $c > 0$ we have,*

$$\mathbb{P}(|W_n - \mathbb{E}(W_n)| > c\mathbb{E}(W_n)) \le 2\exp\left(\frac{-2c^2\mathbb{E}(W_n)^2}{n(b-a)^2}\right)$$

7.2 Proof of Theorems and Lemmas

Proof sketch of Theorem 3.1 and Theorem 7.1. The proof of Eq (7) in Theorem 3.1 and Theorem 7.1 are mainly based on the proof of Theorem 2 in [1] and the concentration property of Gaussian noise. □

Proof of Lemma 3.3. According to Chernoff Bound, we have $P(\hat{\epsilon}_t(m) \le \delta_1 + \frac{\ln(1/\delta_2)}{2\tilde{\tau}(m)}) \ge 1-\delta_2$, which concludes the proof. □

Proof of Lemma 3.4. At time $i \ge t_{c_{j+1}}$, which means the environment has already changed from $\boldsymbol{\theta}_{c_j}^*$ to $\boldsymbol{\theta}_{c_{j+1}}^*$, we have,

$$\begin{aligned}&\mathbb{P}(e_i(m)=1) = \mathbb{P}(|\hat{r}_i - r_i| > B_i(m,a_i)+\epsilon) \qquad (12)\\ =&\mathbb{P}(|(\mathbf{x}_i^\top\hat{\boldsymbol{\theta}}_i - \mathbf{x}_i^\top\boldsymbol{\theta}_{t_{c_j}}^* - \eta_i) + (\mathbf{x}_i^\top\boldsymbol{\theta}_{t_c}^* - \mathbf{x}_i^\top\boldsymbol{\theta}_i^*)| > B_i(m,a_i)+\epsilon)\\ \ge&\mathbb{P}(|\mathbf{x}_i^\top\hat{\boldsymbol{\theta}}_i - \mathbf{x}_i^\top\boldsymbol{\theta}_{t_{c_j}}^* - \eta_i| \le \tilde{B}_i(m,a_i)+\epsilon)\\ &\times\mathbb{P}(|\mathbf{x}_i^\top\boldsymbol{\theta}_{t_{c_j}}^* - \mathbf{x}_i^\top\boldsymbol{\theta}_i^*| > \tilde{B}_i(m,a_i) + B_i(m,a_i) + 2\epsilon)\end{aligned}$$

According to Theorem 7.1, we have $\mathbb{P}\big(|\mathbf{x}_i^\top\hat{\boldsymbol{\theta}}_i - \mathbf{x}_i^\top\boldsymbol{\theta}_{t_{c_j}}^* - \eta_i| \le \tilde{B}_i(m,a_i)+\epsilon\big) \ge 1-\delta_1$. Define U_{c_j} as the upper bound of $\tilde{B}_i(m,a_i)+B_{t_c}(m,a_i)+2\epsilon$. If the change gap Δ_{c_j} satisfies $\Delta_{c_j} \ge U_{c_j}$, we have $\mathbb{P}(e_i(m)=1) \ge \rho(1-\delta_1)$.

Next, we will prove that $\Delta_{c_j} \ge U_{c_j}$ can be achieved by a properly set δ_1. Similar as the proof in Step 2 of Theorem 3.2, where we bound k_c, we have with a high probability that $\tilde{B}_i(m,a_i)+B_{t_{c_j}}(m,a_i)+2\epsilon \le 2\epsilon + 2\sqrt{\lambda} + \frac{2}{\sqrt{\lambda}}S_{c_j}\rho\big(1-\rho(1-\delta_1)\big) = U_{c_j}$. When $\Delta_{c_j} > 2\sqrt{\lambda}+2\epsilon$, $S_{\min} > \frac{\sqrt{\lambda}}{2\rho}(\Delta_{c_j}-2\sqrt{\lambda}-2\epsilon)$ and $\delta_1 \le 1-\frac{1}{\rho}(1-\frac{\sqrt{\lambda}}{2S_{\min}\rho}\big(\Delta_{c_j}-2\sqrt{\lambda}-2\epsilon)\big)$, $\Delta_{c_j} > U_{c_j}$ can be achieved.

Eq (12) indicates when the environment has changed for a slave model m, with a high probability of $e_i(m)=1$ and slave model m will not be updated, which avoids possible contamination in m. According to the concentration inequality in Theorem 7.2, with a probability at least $1-\delta_2$, we have,

$$\sum_{i=t-\tau}^{t} e_i(m) \ge \mathbb{E}[\sum_{i=t-\tau}^{t} e_i(m)] - \sqrt{\frac{\tau}{2}\ln\frac{1}{\delta_2}} \ge \rho(1-\delta_1)\tau - \sqrt{\frac{\tau}{2}\ln\frac{1}{\delta_2}}$$

With simple rewriting, we have when $\tau \ge \frac{2\ln\frac{2}{\delta_2}}{(\rho(1-\delta_1)-\delta_1)^2}$, $\rho(1-\delta_1)\tau - \sqrt{\frac{\tau}{2}\ln\frac{1}{\delta_2}} \ge \delta_1\tau + \sqrt{\frac{\tau}{2}\ln\frac{1}{\delta_2}}$, which means that with a probability at least $1-\delta_2$, $\hat{\epsilon}_t(m) = \frac{\sum_{i=t-\tau}^t e_i}{\tau} \ge \delta_1 + \sqrt{\frac{1}{2\tau}\ln\frac{1}{\delta_2}}$ □

Proof of Lemma 3.6. Under the model selection strategy in line 4 of Algorithm 1, using similar proof techniques as in Theorem 1 of [3], we have

$$\tilde{N}_{c_j}(m) \le l + \sum_{t=t_{c_j}}^{\infty}\sum_{s=1}^{t-1}\sum_{s_i=l}^{t-1}\mathbb{1}\{\hat{e}_{s_i}(m) - d_{s_i}(m) \le \hat{e}_s(m_{c_j}^*) - d_s(m_{c_j}^*)\} \qquad (13)$$

in which $l = \lceil(8\log S_{c_j})/g^2_{m,m_{c_j}}\rceil$. $\hat{e}_{s_i}(m) - d_{s_i}(m) \le \hat{e}_s(m_{c_j}^*) - d_s(m_{c_j}^*)$ implies that at least one of the three following inequalities must hold,

$$\hat{e}_i(m_{c_j}^*) \ge \mathbb{E}[e(m_{c_j}^*)] + d_i(m_{c_j}^*) \qquad (14)$$

$$\hat{e}_i(m) \le \mathbb{E}[e(m)] - d_i(m) \qquad (15)$$

$$\mathbb{E}[e_i(m)] - 2d_i(m) \le \mathbb{E}[e(m_{c_j}^*)] \qquad (16)$$

Intuitively, Eq (14), (15) and (16) correspond to the following three cases respectively. First, the expected 'badness' of the optimal slave model $m_{c_j}^*$ is substantially over-estimated. Second, the expected 'badness' of the slave model m is substantially under-estimated. Third, the expected 'badness' of the two slave models $m_{c_j}^*$ and m are very close to each other.

According to Theorem 7.2, we have $\mathbb{P}\big(\hat{e}_i(m_{c_j}^*) \ge \mathbb{E}[e(m_{c_j}^*)] + d_i(m_{c_j}^*)\big) \le \delta_2$ and $\mathbb{P}\big(\hat{e}_i(m) \le \mathbb{E}[e(m)] - d_i(m)\big) \le \delta_2$. For $s_i \ge l$, we have $\mathbb{E}[e(m)] - \mathbb{E}[e(m_{c_j}^*)] - 2d_i(m) \ge 0$, which means the case in Eq (16) almost never occurs. Substituting the probability for Eq (14), (15) and (16) into Eq (13), and when $\delta_2 = t^{-4}$,

$$\tilde{N}_{c_j}(m) \le \lceil(8\ln S_{c_j})/g^2_{m,m_{c_j}^*}\rceil + \sum_{t=1}^{\infty}\sum_{s=1}^{t-1}\sum_{s_i=l}^{t-1} 2t^{-4} \le (8\ln S_{c_j})/g^2_{m,m_{c_j}^*} + 1 + \frac{\pi^2}{3}$$

which finishes the proof. □