Linear Upper Confidence Bound Algorithm for Contextual Bandit Problem with Piled Rewards

Kuan-Hao Huang and Hsuan-Tien Lin

Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
{r03922062,htlin}@csie.ntu.edu.tw

Abstract. We study the contextual bandit problem with linear payoff function. In the traditional contextual bandit problem, the algorithm iteratively chooses an action based on the observed context, and immediately receives a reward for the chosen action. Motivated by a practical need in many applications, we study the design of algorithms under the piled-reward setting, where the rewards are received as a pile instead of immediately. We present how the Linear Upper Confidence Bound (LinUCB) algorithm for the traditional problem can be naïvely applied under the piled-reward setting, and prove its regret bound. Then, we extend LinUCB to a novel algorithm, called Linear Upper Confidence Bound with Pseudo Reward (LinUCBPR), which digests the observed contexts to choose actions more strategically before the piled rewards are received. We prove that LinUCBPR can match LinUCB in the regret bound under the piled-reward setting. Experiments on the artificial and real-world datasets demonstrate the strong performance of LinUCBPR in practice.

Keywords: contextual bandit, piled rewards, upper confidence bound

1 Introduction

We study the contextual bandit problem (CBP) [13], which is an interactive process between an algorithm and an environment. In the traditional CBP, the algorithm observes a *context* from the environment in each time step. Then, the algorithm is asked to strategically choose an *action* from the action set based on the *context*, and receives a corresponding feedback, called *reward*, while the reward for other actions are hidden from the algorithm. The goal of the algorithm is to maximize the cumulative reward over all time steps.

Because only the reward of the chosen action is revealed, the algorithm needs to choose different actions to estimate their goodness, called *exploration.* On the other hand, the algorithm also needs to choose the better actions to maximize the reward, called *exploitation.* Balancing between exploration and exploitation is arguably the most important issue for designing algorithms of CBP.

ϵ-Greedy [3] and Linear Upper Confidence Bound (LinUCB) [10] are two representative algorithms for CBP. ϵ-Greedy learns one model per action for exploitation and randomly explores different actions with a small probability ϵ.

LinUCB is based on online ridge regression, and takes the concept of *upper-confidence bound* [5] [2] to strategically balance between exploration and exploitation. LinUCB enjoys a strong theoretical guarantee [5] and is state-of-the-art in many practical applications [10].

The traditional CBP setting assumes that the algorithm receives the reward immediately after choosing an action. In some practical applications, however, the environment cannot present the reward to the algorithm immediately. This work is motivated from one such application. Consider an online advertisement system operated by a contextual bandit algorithm. For each user visit (time step), the system (algorithm) receives the information of the user (context) from an ad exchange, and chooses an appropriate ad (action) to display to the user. In the application, the click from the user naturally acts as the reward of the action. Nevertheless, to reduce the cost of communication, the ad exchange often does not reveal the individual reward immediately after choosing an action. Instead, the ad exchange stores the individual reward first, and only sends a pile of rewards back to the system until sufficient number of rewards are gathered. We call the scenario as the contextual bandit problem under the piled-reward setting.

A related setting in the literature is the delayed-reward setting, where the reward is assumed to come at several time steps after the algorithm chooses an action. Most existing works on the delayed-reward setting consider constant delays [6] and cannot be easily applied to the piled-reward setting. Several works [8] [9] [12] propose algorithms for bandit problems with arbitrarily-delayed rewards, but their algorithms are non-contextual. Thus, to the best of our knowledge, no existing work has carefully studied the CBP under the piled-reward setting.

In this paper, we study how LinUCB can be applied under the piled-reward setting. We present a naïve use of LinUCB for the setting and prove its theoretical guarantee in the form of the regret bound. The result helps us understand the difference between the traditional setting and the piled-reward setting. Then, we design a novel algorithm, Linear Upper Confidence Bound with Pseudo Reward (LinUCBPR), which is a variant of LinUCB that allows more strategic use of the context information before the piled rewards are received. We prove that LinUCBPR can match the naïve LinUCB in its regret bound under the piled-reward setting. Experiments on the artificial and real-world datasets demonstrate that LinUCBPR results in strong and stable performance in practice.

This paper is organized as follows. Section 2 formalizes the CBP with the piled-reward setting. Section 3 describes our design of LinUCB and LinUCBPR under the piled-reward setting. The theoretical guarantees of the algorithms are analyzed in Section 4. We discuss the experiment results in Section 5 and conclude in Section 6.

2 Preliminaries

We use bold lower-case symbol like $\mathbf{u}$ to denote a column vector, bold upper-case symbol like $\mathbf{A}$ to denote a matrix, $\mathbf{I}_d$ to denote the $d \times d$ identity matrix, and $[K]$ to denote the set $\{1, 2, ..., K\}$.

We first introduce the CBP under the traditional setting. Let T be the total number of rounds and K be the number of actions. In each round $t \in [T]$, the algorithm observes a context $\mathbf{x}_t \in \mathbb{R}^d$ with $\|\mathbf{x}_t\|_2 \leq 1$ from the environment. Upon observing the context $\mathbf{x}_t$, the algorithm chooses an action a_t from K actions based on the context. Right after choosing a_t, the algorithm receives a reward r_{t,a_t} that corresponds to the context $\mathbf{x}_t$ and the chosen action a_t, while other rewards $r_{t,a}$ for $a \neq a_t$ are hidden from the algorithm. The goal of the algorithm is to maximize the cumulative reward after T rounds.

Now, we introduce the CBP under the piled-reward setting. Instead of receiving the reward right after choosing an action (and thus right before observing the next context), the setting assumes that the rewards come as a pile after observing multiple contexts in a round. We shall extend our notation above to the piled-reward setting as follows. In each round t, the algorithm sequentially observes n contexts $\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, ..., \mathbf{x}_{t_n} \in \mathbb{R}^d$ with $\|\mathbf{x}_{t_i}\|_2 \leq 1$ from the environment. For simplicity, we assume that n is a fixed number while all the technical results in this paper can be easily extended to the case where n can vary in each round. We use t_i to denote the i-th step in round t. For example, 3_5 means the 5-th step in round 3. Upon observing the context $\mathbf{x}_{t_i}$ in round t, the algorithm chooses an action a_{t_i} from K actions based on the context, and observes the next context $\mathbf{x}_{t_{i+1}}$. In the end of round t, the algorithm receives n rewards $r_{t_1,a_{t_1}}, r_{t_2,a_{t_2}}, ..., r_{t_n,a_{t_n}}$ that correspond to $\mathbf{x}_{t_i}$ and a_{t_i}, while other rewards $r_{t_i,a}$ for $a \neq a_{t_i}$ are hidden from the algorithm. The goal is again to maximize the cumulative reward $\sum_{t=1}^{T} \sum_{i=1}^{n} r_{t_i,a_{t_i}}$ after T rounds.

In other words, the piled-reward setting assumes that the context comes at time steps $\{1_1, 1_2, ..., 1_n, 2_1, 2_2, ..., t_1, t_2, ..., T_{n-1}, T_n\}$, while the rewards come after every n contexts as a pile. Note that the traditional setting is a special case of the piled-reward setting when $n = 1$.

In this paper, we consider the CBP with linear payoff function. We assume that $r_{t_i,a}$ connects with $\mathbf{x}_{t_i}$ linearly through K hidden weight vectors $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K \in \mathbb{R}^d$ with $\|\mathbf{u}_i\|_2 \leq 1$. That is, $\mathbb{E}\left[r_{t_i,a} \mid \mathbf{x}_{t_i}\right] = \mathbf{x}_{t_i}^\top \mathbf{u}_a$. Let $a_{t_i}^* = \arg\max_{a \in [K]} \mathbf{x}_{t_i}^\top \mathbf{u}_a$ be the optimal action for $\mathbf{x}_{t_i}$. We define regret of an algorithm to be $(\sum_{t=1}^{T} \sum_{i=1}^{n} r_{t_i,a_{t_i}^*} - \sum_{t=1}^{T} \sum_{i=1}^{n} r_{t_i,a_{t_i}})$. The goal of maximizing the cumulative reward is equivalent to minimizing the regret.

Linear Upper Confidence Bound (LinUCB) [5] is a state-of-the-art algorithm for the traditional CBP ($n = 1$). LinUCB maintains K weight vectors $\mathbf{w}_{t_1,1}, \mathbf{w}_{t_1,2}, ..., \mathbf{w}_{t_1,K}$ to estimate $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K$ at time step t_1. The K weight vectors are calculated by ridge regression

$$\mathbf{w}_{t_1,a} = \underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmin}} \left(\|\mathbf{w}\|^2 + \|\mathbf{X}_{(t-1)_1,a}\mathbf{w} - \mathbf{r}_{(t-1)_1,a}\|^2 \right), \tag{1}$$

where $\mathbf{X}_{(t-1)_1,a}$ is a matrix with rows being the contexts $\mathbf{x}_\tau^\top$ where τ are the time steps before round t and $a_\tau = a$, and $\mathbf{r}_{(t-1)_1,a}$ is a column vector with each element representing the corresponding reward for each context in $\mathbf{X}_{(t-1)_1,a}$. Let $\mathbf{A}_{t_1,a} = (\mathbf{I}_d + \mathbf{X}_{(t-1)_1,a}^\top \mathbf{X}_{(t-1)_1,a})$ and $\mathbf{b}_{t_1,a} = (\mathbf{X}_{(t-1)_1,a}^\top \mathbf{r}_{(t-1)_1,a})$. The solution to (1) is $\mathbf{w}_{t_1,a} = \mathbf{A}_{t_1,a}^{-1} \mathbf{b}_{t_1,a}$.

When a new context $\mathbf{x}_{t_1}$ comes, LinUCB calculates two terms for each action a: the estimated reward $\tilde{r}_{t_1,a} = \mathbf{x}_{t_1}^\top \mathbf{w}_{t_1,a}$ and the uncertainty $c_{t_1,a} = \sqrt{\mathbf{x}_{t_1}^\top \mathbf{A}_{t_1,a}^{-1} \mathbf{x}_{t_1}}$, and chooses the action with the highest score $\tilde{r}_{t_1,a} + \alpha c_{t_1,a}$, where α is a trade-off parameter. After receiving the reward, LinUCB updates the weight vector $\mathbf{w}_{t_1,a_{t_1}}$ immediately, and uses the new weight vector to choose the action for the next context. LinUCB conducts exploration when the chosen action is of high uncertainty. After sufficient (context, action, reward) information is received, $\tilde{r}_{t_1,a}$ shall be close the the expected reward, and $c_{t_1,a}$will be smaller. Then, LinUCB conducts exploitation with the learned weight vectors to choose the action with the highest expected reward.

3 Proposed Algorithm

We first discuss how LinUCB can be naïvely applied under the piled-reward setting. Then, we extend LinUCB to a more general framework that utilizes the additional information within the contexts before the true rewards are received.

Since no rewards are received before the end of the current round t, the naïve LinUCB does not update the model during round t, and only takes the fixed $\mathbf{w}_{t_1,a}$ and $\mathbf{A}_{t_1,a}$ to calculate the estimated reward $\tilde{r}_{t_i,a}$ and the uncertainty $c_{t_i,a}$ for each action a. That is, LinUCB only updates $\mathbf{w}_{t_1,a}$ before the beginning of round t as the solution to (1) with $(\mathbf{X}_{(t-1)_1,a}, \mathbf{r}_{(t-1)_1,a})$ under the traditional setting replaced by $(\mathbf{X}_{(t-1)_n,a}, \mathbf{r}_{(t-1)_n,a})$ under the piled-reward setting. In addition, $\mathbf{A}_{t_1,a}$ can be similarly defined from $\mathbf{X}_{(t-1)_n,a}$ instead.

The naïve LinUCB can be viewed as a baseline upper confidence bound algorithm under the piled-reward setting. There is a possible drawback for the naïve LinUCB. If similar contexts come repeatedly in the same round, because $\mathbf{w}_{t_i,a}$ and $\mathbf{A}_{t_i,a}$ stay unchanged within the round, LinUCB will choose similar actions repeatedly. Then, if the chosen action suffers from low reward, LinUCB suffers from making the low-reward choice repeatedly before the end of the round.

The question is, can we do even better? Our idea is that the contexts $\mathbf{x}_{t_i}$ received during round t can be utilized to update the model *before* the rewards come. That is, at time step t_i, in addition to the *labelled data* (context, action, reward) gathered before time step $(t-1)_n$ that LinUCB uses, the *unlabelled data* (context, action) gathered at time steps $\{t_1, t_2, \ldots, t_{i-1}\}$ can also be included to learn a more decent model. In other words, we hope to design some *semi-supervised* learning scheme within round t to guide the upper-confidence bound algorithm towards more strategic exploration within the round.

Our idea is motivated from the regret analysis. In Section 4, we will show that the regret of LinUCB under the piled-reward setting is bounded by the summation of $c_{t_i,a}$ over all time steps. But note that $c_{t_i,a}$ only depends on $\mathbf{x}_{t_i,a}$ and $\mathbf{A}_{t_i,a}$, but not the reward. That is, upon receiving $\mathbf{x}_{t_i}$ and choosing an action a_{t_i}, the term $c_{t_i,a_{t_i}}$ can readily be updated without the true reward. By updating $c_{t_i,a_{t_i}}$ within the round, the algorithm can explore different actions strategically instead of following similar actions when similar contexts come repeatedly in the same round.

Algorithm 1 LinUCBPR under the piled-reward setting

```
Parameter: $\alpha \in \mathbb{R}^+$
Initialize: $\hat{\mathbf{A}}_{1_1,a} \leftarrow \mathbf{I}_d, \hat{\mathbf{b}}_{1_1,a} \leftarrow \mathbf{0}_{d\times 1}, \hat{\mathbf{w}}_{1_1,a} \leftarrow \hat{\mathbf{A}}^{-1}_{1_1,a}\hat{\mathbf{b}}_{1_1,a}$
for $t = 1, 2, 3, ..., T$ do
    for $i = 1, 2, 3, ..., n$ do
        Observe $\mathbf{x}_{t_i}$ and choose $a_{t_i} = \text{argmax}_{a \in [K]} \mathbf{x}_{t_i}^\top \hat{\mathbf{w}}_{t_i,a} + \alpha\sqrt{\mathbf{x}_{t_i}^\top \hat{\mathbf{A}}^{-1}_{t_i,a} \mathbf{x}_{t_i}}$
        Calculate the pseudo reward $p_{t_i,a_{t_i}}$
        $\hat{\mathbf{A}}_{t_{i+1},a_{t_i}} \leftarrow \hat{\mathbf{A}}_{t_i,a_{t_i}} + \mathbf{x}_{t_i}\mathbf{x}_{t_i}^\top, \quad \hat{\mathbf{b}}_{t_{i+1},a_{t_i}} \leftarrow \hat{\mathbf{b}}_{t_i,a_{t_i}} + \mathbf{x}_{t_i} p_{t_i,a_{t_i}}$
        $\hat{\mathbf{w}}_{t_{i+1},a_{t_i}} \leftarrow \hat{\mathbf{A}}^{-1}_{t_{i+1},a_{t_i}} \hat{\mathbf{b}}_{t_{i+1},a_{t_i}}$
    end for
    Receive rewards $r_{t_1,a_{t_1}}, r_{t_2,a_{t_2}}, ..., r_{t_n,a_{t_n}}$
    for $a \in [K]$ do
        $\hat{\mathbf{A}}_{(t+1)_1,a} \leftarrow \hat{\mathbf{A}}_{t_1,a} + \sum_{a_{t_i}=a} \mathbf{x}_{t_i}\mathbf{x}_{t_i}^\top, \quad \hat{\mathbf{b}}_{(t+1)_1,a} \leftarrow \hat{\mathbf{b}}_{t_1,a} + \sum_{a_{t_i}=a} \mathbf{x}_{t_i} r_{t_i,a_{t_i}}$
        $\hat{\mathbf{w}}_{(t+1)_1,a} \leftarrow \hat{\mathbf{A}}^{-1}_{(t+1)_1,a} \hat{\mathbf{b}}_{(t+1)_1,a}$
    end for
end for
```

This idea can be extended to the following framework. We propose to couple each context $\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \ldots, \mathbf{x}_{t_{i-1}}$ with a pseudo reward p_{τ,a_τ}, where τ is the time step, before receiving the true reward r_{τ,a_τ}. The pseudo reward can then pretend to be the true reward and allow the algorithm to keep updating the model before the true rewards are received. Note that pseudo rewards have been used to speed up exploration in the traditional CBP [4], and can encourage more strategic exploration in our framework. We name the framework Linear Upper Confidence Bound with Pseudo Reward (LinUCBPR). The framework updates the weight vector and the estimated covariance matrix by

$$\hat{\mathbf{w}}_{t_i,a} = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d} \left(\|\mathbf{w}\|^2 + \|\mathbf{X}_{(t-1)_n,a}\mathbf{w} - \mathbf{r}_{(t-1)_n,a}\|^2 + \|\hat{\mathbf{X}}_{t_{i-1},a}\mathbf{w} - \mathbf{p}_{t_{i-1},a}\|^2\right) \quad (2)$$

$$\hat{\mathbf{A}}_{t_i,a} = \mathbf{I}_d + \mathbf{X}^\top_{(t-1)_n,a}\mathbf{X}_{(t-1)_n,a} + \hat{\mathbf{X}}^\top_{t_{i-1},a}\hat{\mathbf{X}}_{t_{i-1},a} \quad (3)$$

where $\hat{\mathbf{X}}_{t_{i-1},a}$ is a matrix with rows being the contexts $\mathbf{x}_\tau^\top$ with $t_1 \le \tau \le t_{i-1}$ and $a_\tau = a$, and $\mathbf{p}_{t_{i-1},a}$ is a column vector with each element representing the corresponding pseudo reward for each context in $\hat{\mathbf{X}}_{t_{i-1},a}$.

When receiving the true rewards in the end of round t, we discard the change from pseudo rewards, and use the true rewards to update model again. We show the framework of LinUCBPR in Algorithm 1.

The only remained task is what $p_{\tau,a}$ should be. We will study two variants, one is to use $p_{\tau,a} = \tilde{r}_{\tau,a}$, the estimated reward of actions. We name the variant LinUCBPR with estimated reward (LinUCBPR-ER). Another variant is to be even more aggressive, and set $p_{\tau,a} = \tilde{r}_{\tau,a} - \beta c_{\tau,a}$, a lower-confidence bound of the reward, where β is a trade-off parameter. The lower-confidence bound can be viewed as the underestimated reward, and should allow more exploration within the round, at the cost of more computation. We name the variant LinUCBPR with underestimated reward (LinUCBPR-UR).

Algorithm 2 BaseLinUCB under the piled-reward setting at round t

```
Parameter: $\alpha \in \mathbb{R}^+, \Psi_t \subseteq \{1_1, 1_2, ..., (t-1)_n\}$
$\bar{\mathbf{A}}_{t_1} \leftarrow \mathbf{I}_{dK} + \sum_{\tau\in\Psi_t} \bar{\mathbf{x}}_{\tau,a_\tau}\bar{\mathbf{x}}_{\tau,a_\tau}^\top,\ \bar{\mathbf{b}}_{t_1} \leftarrow \mathbf{0}_{dK\times 1} + \sum_{\tau\in\Psi_t} \bar{\mathbf{x}}_{\tau,a_\tau} r_{\tau,a_\tau},\ \bar{\mathbf{w}}_{t_1} \leftarrow \bar{\mathbf{A}}_{t_1}^{-1}\bar{\mathbf{b}}_{t_1}$
for $i = 1, 2, 3, ..., n$ do
    Observe $\mathbf{x}_{t_i}$ and calculate $\bar{\mathbf{x}}_{t_i,1}, \bar{\mathbf{x}}_{t_i,2}, ..., \bar{\mathbf{x}}_{t_i,K}$
    for $a \in [K]$ do
        $width_{t_i,a} \leftarrow (1+\alpha)\sqrt{\bar{\mathbf{x}}_{t_i,a}^\top \bar{\mathbf{A}}_{t_1}^{-1} \bar{\mathbf{x}}_{t_i,a}}$
        $ucb_{t_i,a} \leftarrow \bar{\mathbf{x}}_{t_i,a}^\top \bar{\mathbf{w}}_{t_1} + width_{t_i,a}$
    end for
end for
```

4 Theoretical Analysis

In this section, we establish the theoretical guarantee for the regret bound of LinUCB and LinUCBPR-ER under the piled-reward setting. Similar to the analysis of LinUCB in the immediate-reward setting [5], there is a difficulty. In particular, the algorithms choose actions based on previous outcomes. Hence, the rewards in each round are not independent random variables. To deal with this problem, we follow the approach of [5]. We modify the algorithm to a base algorithm which assumes the independent rewards, and construct a master algorithm which ensures that the assumption holds.

Note that [5] takes a CBP setting with one context *per action* instead for our setting of the one context *share by actions*. To let the notation be consistent with [5], we simply cast our setting as theirs by following steps. We define a (dK)-dimensional vector $\bar{\mathbf{u}}$ to be the concatenation of $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K$, and define a (dK)-dimensional context $\bar{\mathbf{x}}_{\tau,a}$ per action with $\mathbf{x}_\tau$, where $\bar{\mathbf{x}}_{\tau,a} = \begin{bmatrix}\mathbf{0}\ \mathbf{0}\cdots\mathbf{0}\ \mathbf{x}_\tau^\top\ \mathbf{0}\cdots\mathbf{0}\end{bmatrix}^\top$ with $\mathbf{x}_\tau$ being the a-th vector within the concatenation. All $\bar{\mathbf{X}}_\tau$, $\bar{\mathbf{A}}_\tau$, $\bar{\mathbf{r}}_\tau$, $\bar{\mathbf{b}}_\tau$, and $\bar{\mathbf{w}}_\tau$ can be similarly defined from $\hat{\mathbf{X}}_{\tau,a}$, $\hat{\mathbf{A}}_{\tau,a}$, $\mathbf{r}_{\tau,a}$, $\hat{\mathbf{b}}_{\tau,a}$, and $\hat{\mathbf{w}}_{\tau,a}$.

4.1 Regret for LinUCB under the piled-reward setting

Algorithm 2 lists the base algorithm for LinUCB under the piled-reward setting, called BaseLinUCB. We first prove the theoretical guarantee of BaseLinUCB. Let $\bar{c}_{t_i,a} = \sqrt{\bar{\mathbf{x}}_{t_i,a}^\top \bar{\mathbf{A}}_{t_1}^{-1} \bar{\mathbf{x}}_{t_i,a}}$. We can establish the following lemmas.

Lemma 1 (Li et al. [5], Lemma 1). *Suppose the input time step set $\Psi_t \subseteq \{1_1, 1_2, ..., (t-1)_n\}$ given to BaseLinUCB has property that for fixed context $\bar{\mathbf{x}}_{t_i,a}$ with $t_i \in \Psi_t$, the corresponding rewards $r_{t_i,a}$ are independent random variables with means $\bar{\mathbf{x}}_{t_i,a}^\top \bar{\mathbf{u}}$. Then, for some $\alpha = \mathcal{O}(\sqrt{\ln(nTK/\delta)})$, we have with probability at least $1 - \delta/(nT)$ that $\left|\bar{\mathbf{x}}_{t_i,a}^\top \bar{\mathbf{w}}_{t_i} - \bar{\mathbf{x}}_{t_i,a}^\top \bar{\mathbf{u}}\right| \leq (1+\alpha)\bar{c}_{t_i,a}$.*

Note that in Lemma 1, the bound is related to the time steps. We want the bound to be related to the rounds, and hence establish Lemma 2 and Lemma 3.

Lemma 2. *Let ψ_t be a subset of $\{t_1, t_2, ..., t_n\}$. Suppose $\Psi_{t+1} = \Psi_t \cup \psi_t$ in BaseLinUCB. Then, the eigenvalues of $\bar{A}_{t_1}$ and $\bar{A}_{(t+1)_1}$ can be arranged so that $\lambda_{t_1,j} \le \lambda_{(t+1)_1,j}$ for all j and $\bar{c}^2_{t_i,a} \le 10 \sum_{j=1}^{dK} \frac{\lambda_{(t+1)_1,j} - \lambda_{t_1,j}}{\lambda_{t_1,j}}$.*

Proof. The proof can be done by combining Lemma 2 and Lemma 8 in [5].

Lemma 3. *Let $\Phi_{t+1} = \{t \mid t \in [T]$ and $\exists\, j$ such that $t_j \in \Psi_{t+1}\}$, and assume $|\Phi_{t+1}| \ge 2$. Then $\sum_{t_i \in \Psi_{t+1}} \bar{c}_{t_i,a} \le 5n\sqrt{dK\, |\Phi_{t+1}| \ln |\Phi_{t+1}|}$.*

Proof. By Lemma 2 and the technique in the proof of Lemma 3 in [5], we have

$$\begin{aligned}\sum_{t_i \in \Psi_{t+1}} \bar{c}_{t_i,a} &\le \sum_{t_i \in \Psi_{t+1}} \sqrt{10 \sum_{j=1}^{dK} \frac{\lambda_{(t+1)_1,j} - \lambda_{t_1,j}}{\lambda_{t_1,j}}} \\ &\le \sum_{t \in \Phi_{t+1}} n \sqrt{10 \sum_{j=1}^{dK} \frac{\lambda_{(t+1)_1,j} - \lambda_{t_1,j}}{\lambda_{t_1,j}}} \le 5n\sqrt{dK\, |\Phi_{t+1}| \ln |\Phi_{t+1}|}.\end{aligned}$$

We construct SupLinUCB on each round similar to [5]. Then, we borrow Lemma 14 and Lemma 15 of [2], and extend Lemma 16 of [2] to the following lemma.

Lemma 4. *For each $s \in [S]$, $\left|\Psi^s_{T+1}\right| \le 5n \cdot 2^s (1+\alpha)\sqrt{dK \left|\Phi^s_{t+1}\right| \ln \left|\Phi^s_{t+1}\right|}$.*

Based on the lemmas, we can then establish the following theorem for the regret bound of LinUCB under the piled-reward setting.

Theorem 1. *For some $\alpha = \mathcal{O}(\sqrt{\ln(nTK/\delta)})$, with probability $1-\delta$, the regret of LinUCB under the piled-reward setting is $\mathcal{O}(\sqrt{dn^2TK \ln^3(nTK/\delta)})$.*

Proof. Let $\Psi^0 = \{1_1, 1_2, ..., T_n\} \setminus \bigcup_{s \in [S]} \Psi^s_{T+1}$. Observing that $s^{-s} \le 1/\sqrt{T}$, given the previous lemmas and Jensen's inequality, we have

$$\begin{aligned} Regret &= \sum_{t=1}^{T} \sum_{i=1}^{n} \left(\mathbb{E}[r_{t_i,a^*_{t_i}}] - \mathbb{E}[r_{t_i,a_{t_i}}]\right) \\ &= \sum_{t_i \in \Psi^0} \left(\mathbb{E}[r_{t_i,a^*_{t_i}}] - \mathbb{E}[r_{t_i,a_{t_i}}]\right) + \sum_{s=1}^{S} \sum_{t_i \in \Psi^s_{T+1}} \left(\mathbb{E}[r_{t_i,a^*_{t_i}}] - \mathbb{E}[r_{t_i,a_{t_i}}]\right) \\ &\le \frac{2}{\sqrt{T}} \left|\Psi^0\right| + \sum_{s=1}^{S} 2^{3-s} \left|\Psi^s_{T+1}\right| \\ &\le \frac{2}{\sqrt{T}} \left|\Psi^0\right| + \sum_{s=1}^{S} 40n(1+\alpha)\sqrt{dK \left|\Phi^s_{t+1}\right| \ln \left|\Phi^s_{t+1}\right|} \\ &\le 2n\sqrt{T} + 40n(1+\alpha)\sqrt{dK \ln T} \sum_{s=1}^{S} \sqrt{\left|\Phi^s_{t+1}\right|} \\ &\le 2n\sqrt{T} + 40n(1+\alpha)\sqrt{dK \ln T}\sqrt{ST}. \end{aligned}$$

The rest of proof is almost identical to the proof of Theorem 6 in [2]. By substituting $\alpha = \mathcal{O}(\sqrt{\ln(nTK/\delta})$, replacing δ with $\delta/(S+1)S$, substituting $S = \ln(nT)$, and applying Azuma's inequality, we obtain Theorem 1.

Note that if we let $nT = C$ to be a constant, the original regret bound under the traditional setting ($n = 1$) in [5] is $\mathcal{O}(\sqrt{dCK \ln^3(CK/\delta)})$, while the regret bound under the piled-reward setting is $\mathcal{O}(\sqrt{dnCK \ln^3(CK/\delta)})$, which is the original bound multiplied by $\sqrt{n}$.

4.2 Regret for LinUCBPR-ER under the piled-reward setting

We first prove two lemmas for LinUCBPR-ER.

Lemma 5. *After updating with the context $\mathbf{x}_{t_i}$ and the pseudo reward $p_{t_i,a} = \tilde{r}_{t_i,a}$, the estimated reward of LinUCBPR-ER is the same. That is, $\tilde{r}_{t_{i+1}} = \tilde{r}_{t_i}$.*

Proof. Because $p_{t_i,a} = \mathbf{x}_{t_i}^\top \hat{\mathbf{w}}_{t_i,a} = \tilde{r}_{t_i,a}$, $\mathbf{x}_{t_i}$ and $p_{t_i,a}$ will not change $\hat{\mathbf{w}}_{t_i,a}$. Thus the reward stays the same.

Lemma 6. *After updating with the context $\mathbf{x}_{t_i}$ and the pseudo reward $p_{t_i,a} = \tilde{r}_{t_i,a}$, the uncertainty of LinUCBPR-ER for the context is non-increasing. That is, for any $\mathbf{x}$, $\sqrt{\mathbf{x}^\top \hat{\mathbf{A}}_{t_{i+1},a}^{-1} \mathbf{x}} \leq \sqrt{\mathbf{x}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x}}$.*

Proof. By Sherman-Morrison formula, we have

$$\mathbf{x}^\top \hat{\mathbf{A}}_{t_{i+1},a}^{-1} \mathbf{x} = \mathbf{x}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x} - \frac{\mathbf{x}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x}_{t_i} \mathbf{x}_{t_i}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x}}{1 + \mathbf{x}_{t_i}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x}_{t_i}} = \mathbf{x}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x} - \frac{\left(\mathbf{x}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x}_{t_i}\right)^2}{1 + \mathbf{x}_{t_i}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x}_{t_i}}.$$

The second term is greater than or equal to zero, and implies the lemma.

Similarly, we can construct BaseLinUCBPR-ER and SupLinUCBPR-ER. By Lemma 5, we have that for each time step t_i in the round t, the estimated reward $\bar{\mathbf{x}}_{t_i,a}^\top \bar{\mathbf{w}}_{t_i} = \bar{\mathbf{x}}_{t_i,a}^\top \bar{\mathbf{w}}_{t_1}$ does not change. Furthermore, By Lemma 6, we have $\sqrt{\mathbf{x}^\top \hat{\mathbf{A}}_{t_i,a}^{-1} \mathbf{x}} \leq \sqrt{\mathbf{x}^\top \hat{\mathbf{A}}_{t_1,a}^{-1} \mathbf{x}}$. Hence, all lemmas we need also hold for BaseLinUCBPR-ER. Similar to LinUCB, we can then establish the following theorem. The proof is almost identical to Theorem 1.

Theorem 2. *For some $\alpha = \mathcal{O}(\sqrt{\ln(nTK/\delta)})$, with probability $1 - \delta$, the regret of LinUCBPR-ER under the piled-reward setting is $\mathcal{O}(\sqrt{dn^2TK \ln^3(nTK/\delta)})$.*

5 Experiments

We apply the proposed algorithms on both artificial and real-world datasets to justify that using pseudo-rewards is useful. In addition, we follow [1], and take the *simple supervised-to-contextual-bandit transformation* [7] on 8 multi-class datasets to evaluate our idea.

Table 1. ACR on artificial datasets (mean ± std)

	$d = 10$				$d = 30$			
	$K = 50$		$K = 100$		$K = 50$		$K = 100$	
	$n = 500$	$n = 1000$	$n = 500$	$n = 1000$	$n = 500$	$n = 1000$	$n = 500$	$n = 1000$
Ideal	0.6607 ±0.0002	0.6607 ±0.0002	0.7061 ±0.0002	0.7061 ±0.0002	0.3930 ±0.0002	0.3930 ±0.0002	0.4252 ±0.0002	0.4252 ±0.0002
ϵ-Greedy	0.6265 ±0.0030	0.6329 ±0.0016	0.6317 ±0.0043	0.6538 ±0.0030	0.3566 ±0.0030	0.3690 ±0.0014	0.3537 ±0.0022	0.3739 ±0.0026
LinUCB	0.6555 ±0.0004	0.6513 ±0.0005	0.6866 ±0.0011	0.6868 ±0.0011	0.3905 ±0.0003	0.3880 ±0.0004	0.4188 ±0.0007	0.4164 ±0.0004
LinUCB PR-ER	**0.6591** **±0.0001**	**0.6535** **±0.0002**	**0.7000** **±0.0012**	**0.7040** **±0.0003**	**0.3917** **±0.0002**	**0.3896** **±0.0002**	**0.4227** **±0.0004**	**0.4224** **±0.0004**
LinUCB PR-UR	0.6586 ±0.0001	0.6533 ±0.0001	0.6978 ±0.0011	0.7027 ±0.0003	0.3911 ±0.0002	0.3887 ±0.0002	0.4210 ±0.0002	0.4215 ±0.0003
QPM-D	0.6552 ±0.0003	0.6502 ±0.0004	0.6925 ±0.0010	0.6860 ±0.0013	0.3897 ±0.0003	0.3871 ±0.0003	0.4172 ±0.0007	0.4123 ±0.0010

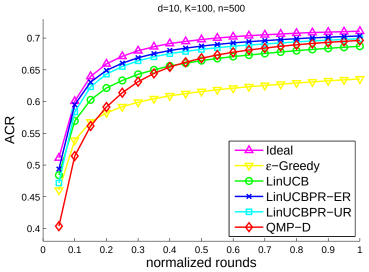


Fig. 1. ACR versus round

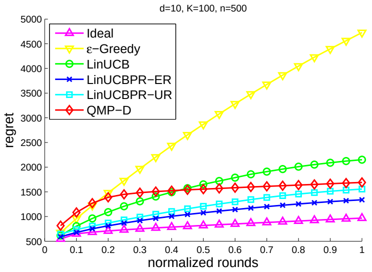


Fig. 2. regret versus round

Artificial Datasets. For each artificial dataset, we first sample unit vectors $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K$ uniformly from $\mathbb{R}^d$ to simulate the K actions. In each round t, the context $\mathbf{x}_{t_i}$ is sampled from an uniform distribution within $\|\mathbf{x}_{t_i}\| \leq 1$. The reward is generated by $r_{t_i,a_{t_i}} = \mathbf{u}_{a_{t_i}}^\top \mathbf{x}_{t_i} + \epsilon_{t_i}$, where $\epsilon_{t_i} \in [-0.05, 0.05]$ is a uniform random noise. In the experiments, we let $nT = 50000$ to be a constant, and consider parameters $d \in \{10, 30\}$, $K \in \{50, 100\}$, and $n \in \{500, 1000\}$.

We compare the performance of ϵ-Greedy, LinUCB, LinUCBPR-ER and LinUCBPR-UR under the piled-reward setting. We also compare Queued Partial Monitoring with Delays (QPM-D) [9], which uses a queue to handle arbitrarily-delay rewards. Furthermore, we consider an “ideal” LinUCB under the traditional setting ($n = 1$) to study the difference between the traditional setting and the piled-reward setting. The parameters of algorithms are selected by grid search, where $\alpha, \beta \in \{0.05, 0.10, ..., 1.00\}$ and $\epsilon \in \{0.025, 0, 05, ..., 0.1\}$. We run the experiment 20 times and show the average cumulative reward (ACR), which is the cumulative reward over the number of time steps, in Table 1. From the table, Ideal LinUCB clearly outperforms others. This verifies that the piled-reward setting introduces difficulty in applying upper-confidence bound algorithms. It also echoes the regret bound in Section 4, where LinUCB under the piled-reward setting suffers some penalty when compared with the original bound.

Next, we focus on the influence of the pseudo rewards. LinUCBPR-ER and LinUCBPR-UR are consistently better than LinUCB on all datasets. Figures 1

Table 2. datasets

dataset	D	K
shuttle	9	7
poker	10	10
pendigits	16	10
letter	16	26
satimage	36	6
acoustic	50	3
covtype	54	7
usps	256	10

Table 4. t-test at 95% confidence level (win/tie/loss)

	Competitor				
Algorithm	ϵ-Greedy	LinUCB	LinUCB PR-ER	LinUCB PR-UR	QPM-D
ϵ-Greedy	–	0/0/8	0/0/8	0/0/8	0/0/8
LinUCB	8/0/0	–	0/1/7	2/4/2	1/6/1
LinUCBPR-ER	8/0/0	7/1/0	–	5/3/0	6/2/0
LinUCBPR-UR	8/0/0	2/4/2	0/3/5	–	2/5/1
QPM-D	8/0/0	1/6/1	0/2/6	1/5/2	–

Table 3. ACR on supervised-to-contextual-bandit datasets (mean ± std)

	shuttle	poker	pendigits	letter	satimage	acoustic	covtype	usps
Ideal	0.9373 ±0.0005	0.4866 ±0.0075	0.8929 ±0.0056	0.6271 ±0.0117	0.8344 ±0.0014	0.7216 ±0.0006	0.6987 ±0.0020	0.9358 ±0.0012
ϵ-Greedy	0.8844 ±0.0092	0.4766 ±0.0086	0.8667 ±0.0058	0.4746 ±0.0247	0.8062 ±0.0024	0.6992 ±0.0012	0.6736 ±0.0035	0.9009 ±0.0023
LinUCB	0.9168 ±0.0068	0.4863 ±0.0087	0.8876 ±0.0043	0.5696 ±0.0176	0.8225 ±0.0016	0.7103 ±0.0016	0.6888 ±0.0039	0.9192 ±0.0017
LinUCB PR-ER	**0.9200** **±0.0029**	**0.4865** **±0.0046**	**0.8901** **±0.0019**	**0.6053** **±0.0137**	**0.8236** **±0.0022**	**0.7112** **±0.0007**	**0.6915** **±0.0021**	**0.9221** **±0.0011**
LinUCB PR-UR	0.9170 ±0.0027	0.4846 ±0.0107	0.8872 ±0.0043	0.6017 ±0.0167	0.8189 ±0.0045	0.7099 ±0.0021	0.6913 ±0.0014	0.9179 ±0.0025
QPM-D	0.9166 ±0.0046	0.4860 ±0.0033	0.8844 ±0.0044	0.5585 ±0.0225	0.8221 ±0.0024	0.7101 ±0.0012	**0.6915** **±0.0013**	0.9185 ±0.0018

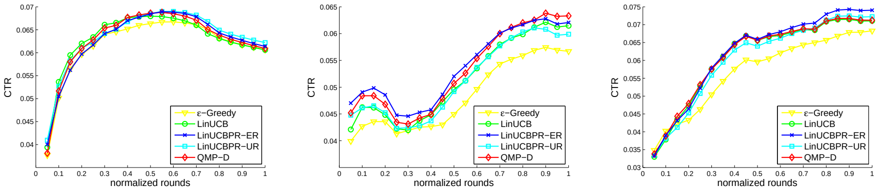
and 2 respectively depict the ACR and the regret along normalized rounds, which is t/T, when $d = 10$, $K = 100$, and $n = 500$. Note that LinUCBPR algorithms enjoy an advantage in the early rounds. This is because the exploration is generally more important than the exploitation in the early rounds, and LinUCBPR algorithms encourage more strategic exploration by using pseudo rewards.

We take ϵ-Greedy to compare the effect of conducting exploration within the round based on randomness rather than pseudo rewards. Table 1 suggests LinUCBPR algorithms reach much better performance, and justifies the effectiveness of the strategic exploration. We also compare LinUCBPR algorithms with QPM-D. Table 1 shows that LinUCBPR algorithms are consistently better than QPM-D. The results again justify the superiority of LinUCBPR algorithms.

LinUCBPR-ER and LinUCBPR-UR perform quite comparably across all datasets. The results suggest that we do not need to be more aggressive than LinUCBPR-ER. The simple LinUCBPR-ER, which can be efficiently implemented by updating $\mathbf{A}_{t_i,a}$ only, can readily reach decent performance.

Supervised-to-contextual-bandit Datasets. Next, we take 8 public multiclass datasets[1] (Table 2). We randomly split each dataset into two parts: 30% for parameter tuning and 70% for testing. For each part, we repeatedly present the examples as an infinite data stream. We let $nT = 10000$ for parameter tuning and $nT = 30000$ for testing. We consider $n = 500$ for all datasets. The parameter setting is the same as the one for artificial datasets.

[1] available from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`



(a) R6A (b) R6B (c) R6B-clean

Fig. 3. CTR versus round on real-world datasets

Table 3 shows the average results of 20 experiments. LinUCBPR-ER is consistently better than others, and LinUCBPR-UR is competitive with others. The results again confirm that LinUCBPR algorithms are useful under the piled-reward setting, and also again confirm that LinUCBPR-ER to be the best algorithm. We further compare these algorithms with a two-sample t-test at 95% confidence level in Table 4. The results demonstrate the significance of the strong performance of LinUCB-ER.

Real-world Datasets. Finally, We use two real-world datasets R6A and R6B released by Yahoo! to examine our proposed algorithms. The datasets are the only two public datasets for the CBP to the best of our knowledge. They first appear in ICML 2012 workshop on New Challenges for Exploration and Exploitation 3 and also appear in [10]. They are about the news article recommendation.

Note that the action set of the two datasets are dynamic. To deal with this, we let algorithms maintain a weight vector $\mathbf{w}_{t_i,a}$ for each action. The dimensions of the contexts for R6A and R6B are 6 and 136 separately. The rewards for both datasets are in $\{0, 1\}$, which represent the clicks from users. We note that in R6B, there are some examples that do not come with valid contexts. Hence we remove these examples and form a new dataset, R6B-clean. We use *click through rate* (CTR) to evaluate the algorithms, and use the technique described in [11] to achieve an unbiased off-line evaluation.

We split the datasets into two parts: parameter tuning part and testing testing part. For R6A, we let $nT = 10000$ for parameter tuning and $nT = 300000$ for testing. For R6B and R6B-clean, we let $nT = 10000$ for parameter tuning and $nT = 100000$ for testing. We consider $n = 500$ for each dataset. The parameter setting is the same as the one for artificial datasets.

Figures 3 shows the experiment results. Unlike the results of artificial datasets, the CTR curve is non-monotonic. This is possibly because the action set is dynamic, and the better actions may disappear from the action set in the middle, which leads to some dropping of CTR.

LinUCBPR algorithms and QPM-D usually perform better than LinUCB and ϵ-Greedy in these datasets. LinUCBPR-ER is stable among the better choices, while LinUCBPR-UR and QPM-D can sometimes be inferior. The results again suggest LinUCBPR-ER to be a promising algorithm for the piled-reward setting.

6 Conclusion

We introduce the contextual bandit problem under the piled-reward setting and show how to apply LinUCB to this setting. We also propose a novel algorithm, LinUCBPR, which uses the pseudo reward to encourage strategic exploration to utilize received contexts that are temporarily without rewards. We prove a regret bound for both LinUCB and the LinUCBPR with estimated reward (-ER), and discuss how the bound compares with the original bound. Empirical results show that LinUCBPR perform better in early time steps, and is competitive in the long term. Most importantly, LinUCBPR-ER yields promising performance on all datasets. The results suggest LinUCBPR-ER to be the best choice in practice.

Acknowledgements

We thank the anonymous reviewers and the members of the NTU CLLab for valuable suggestions. This work is partially supported by the Ministry of Science and Technology of Taiwan (MOST 103-2221-E-002 -148 -MY3) and Asian Office of Aerospace Research and Development (AOARD FA2386-15-1-4012).

References

1. Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., Schapire, R.E.: Taming the monster: A fast and simple algorithm for contextual bandits. In: ICML. pp. 1638–1646 (2014)
2. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. Journal of Machine Learning Research 3, 397–422 (2003)
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning 47(2-3), 235–256 (2002)
4. Chou, K.C., Chiang, C.K., Lin, H.T., Lu, C.J.: Pseudo-reward algorithms for contextual bandits with linear payoff functions. In: ACML. pp. 344–359 (2014)
5. Chu, W., Li, L., Reyzin, L., Schapire, R.E.: Contextual bandits with linear payoff functions. In: AISTATS. pp. 208–214 (2011)
6. Dudík, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., Zhang, T.: Efficient optimal learning for contextual bandits. In: UAI. pp. 169–178 (2011)
7. Dudík, M., Langford, J., Li, L.: Doubly robust policy evaluation and learning. In: ICML. pp. 1097–1104 (2011)
8. Guha, S., Munagala, K., Pal, M.: Multiarmed bandit problems with delayed feedback. arXiv:1011.1161 (2010)
9. Joulani, P., György, A., Szepesvári, C.: Online learning under delayed feedback. In: ICML. pp. 1453–1461 (2013)
10. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: WWW. pp. 661–670 (2010)
11. Li, L., Chu, W., Langford, J., Wang, X.: Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In: WSDM. pp. 297–306 (2011)
12. Mandel, T., Liu, Y.E., Brunskill, E., Popovic, Z.: The queue method: Handling delay, heuristics, prior data, and evaluation in bandits. In: AAAI (2015)
13. Wang, C.C., Kulkarni, S.R., Poor, H.V.: Bandit problems with side observations. IEEE Transactions on Automatic Control 50(3), 338–355 (2005)