Pairwise Regression with Upper Confidence Bound for Contextual Bandit with Multiple Actions

Ya-Hsuan Chang
Dept of Computer Science & Information Engineering
National Taiwan University, Taipei, Taiwan
r00922044@csie.ntu.edu.tw

Hsuan-Tien Lin
Dept of Computer Science & Information Engineering
National Taiwan University, Taipei, Taiwan
htlin@csie.ntu.edu.tw

***Abstract*—The contextual bandit problem is typically used to model online applications such as article recommendation. However, the problem cannot fully meet certain needs of these applications, such as performing multiple actions at the same time. We defined a new Contextual Bandit Problem with Multiple Actions (CBMA), which is an extension of the traditional contextual bandit problem and fits the online applications better. We adapt some existing contextual bandit algorithms for our CBMA problem, and developed the new Pairwise Regression with Upper Confidence Bound (PairUCB) algorithm which addresses the new properties of the new CBMA problem. Experimental results demonstrate that PairUCB significantly outperforms other approaches.**

***Keywords*-machine learning; contextual bandit; upper confidence bound;**

I. Introduction

The contextual bandit problem [1], [2] in machine learning, also known as K-arms bandit problem with context [3], is an extension of the traditional bandit problem. At each iteration of the setting, an external environment provides a set of contexts, where each context connects to a possible action of the learning algorithm (usually called player). The player must choose one action based on the contexts. Then, the environment reveals the reward of the chosen action, while the rewards for other unselected actions remains unknown. The goal of the algorithm is to earn as much reward as possible throughout all iterations.

Based on the flexibility of using contexts, the contextual bandit problem is more realistic than the traditional bandit problem (which does not consider contexts) for many applications, such as advertising recommendation and personalized new article recommendation [2], [4], [5]. There are some variants of the contextual bandit problem that represent the needs of different applications. For instance, some existing studies [6], [7] assume that contexts connect to the same set of possible actions at each iteration; whereas, other studies [4] assume that the action set could change at each iteration. We call the latter assumption as the dynamic action setting, which will be the main focus of this paper.

Because only the reward of the chosen action is revealed, the player only has partial information about the goodness of the chosen action. A major issue when dealing with partial information is the trade-off between exploitation and exploration. Exploitation is greedily choosing the action with the highest estimated reward; exploration is bravely choosing the uncertain action to learn more about the environment. Currently, Upper Confidence Bound (UCB) [8] is a leading framework that strikes some balance between exploitation and exploration. One state-of-the-art representative of the UCB framework is Linear Hypothesis with Upper Confidence Bound (LinUCB) [4], [9], which provides a strong theoretical guarantee and promising empirical results.

The contextual bandit problem allows the player to choose *one* action per iteration [2], [4], [5]; however, it is possible for the player to choose more than one action in some applications. For instance, when designing a personalized recommendation system, an action corresponds to an item shown to a user, and it is common for the system to recommend multiple items to the user *simultaneously* rather than only recommending one. This scenario is not fully addressed by the contextual bandit problem with single action; therefore, in this paper, we extend the contextual bandit problem to the more general version, Contextual Bandit with Multiple Actions (CBMA) problem, to model the scenario in those applications. CBMA allows the player to choose *multiple* actions and receive multiple rewards that correspond to those actions at each iteration, while considering the original contextual bandit problem (with single action) as a special case.

To the best of our knowledge, this study is the first work that formalizes the CBMA problem. The problem is non-trivial to tackle with existing contextual bandit algorithms. In particular, because of the need to choose actions *simultaneously* before receiving any rewards, we cannot directly solve CBMA by running a contextual bandit algorithm several times to select multiple actions.

In this work, we first systematically adapted existing contextual bandit algorithms as baseline approaches for CBMA. Then, we developed a better approach that utilizes the feedback from multiple rewards properly. The approach is based on two key ideas. First, we model the choice of actions as a (pairwise) ranking task, and we solve the task efficiently with pairwise linear regression. Second, we defined an upper

confidence bound for pairwise linear regression based on its connection to usual linear regression. Combining the two ideas leaded to a novel approach: Pairwise Regression with Upper Confidence Bound (PairUCB), which properly balances exploitation and exploration for CBMA. Experimental results demonstrated that PairUCB is almost always significantly better than other approaches.

The paper is organized as follows. In Section II, we define the CBMA problem and review related works. In Section III, we describe how adapt existing contextual bandit algorithms for the CBMA problem. In Section IV, we propose the novel PairUCB approach. Finally, we present the experiment results in Section V and the conclusion in Section VI.

II. Problem Setup and Related Work

Define $[N] = \{1, \ldots, N\}$ for any $N \in \mathbb{N}$. The Contextual Bandit with Multiple Actions (CBMA) problem is formally defined in Algorithm 1. CBMA is an iterative procedure with T iterations. For each iteration, the environment provides a context matrix $\mathbf{X}_t = [\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,K}]^\top \in \mathbb{R}^{K \times d}$, where each row represents the context of a potential action. Then, the algorithm $\mathcal{A}$ is asked to choose M different actions from the K potential actions. We denote an action vector $\mathbf{a}_t = (a_{t,1}, \ldots, a_{t,M}) \in [K]^M$ to be the M chosen actions.

After choosing the actions, the environment reveals the M corresponding rewards to $\mathcal{A}$. The M rewards form a reward vector $\mathbf{r}_t = (\mathrm{rw}(a_{t,1}), \ldots, \mathrm{rw}(a_{t,M})) \in \mathbb{R}^M$, where $\mathrm{rw}(\cdot)$ is some unknown reward function. At the end of each iteration, $\mathcal{A}$ updates its internal model with the information collected in this iteration. The goal of the CBMA problem is to maximize the cumulative reward of all chosen actions $\sum_{t=1}^{T} \sum_{m=1}^{M} \mathrm{rw}(a_{t,m})$. The traditional contextual bandit problem is simply a special case of CBMA where $M = 1$.

Algorithm 1 Framework of CBMA

1: **Input:** T, K, M
2: **for** $t = 1 \ldots T$ **do**
3: environment provides context matrix $\mathbf{X}_t$
4: $\mathbf{a}_t = \mathcal{A}.\textit{choose}(\mathbf{X}_t, M)$
5: environment reveals reward vector $\mathbf{r}_t$
6: $\mathcal{A}.\textit{update}(\mathbf{X}_t, \mathbf{a}_t, \mathbf{r}_t)$
7: **end for**

The non-contextual bandit problem with multiple actions has been studied by [10], but the work cannot be directly applied to CBMA because of the non-contextual setting; moreover, it essentially limits the potential actions to fixed number K rather than dynamically-connected with contexts.

Next, we review some of the key components that are used by existing contextual bandit algorithms for choosing one action to motivate the design of CBMA algorithms.

The reward estimator is a key component for almost any contextual bandit algorithm. Because the contextual bandit problem occurs in online procedures, most existing works tend to use the estimator that could be updated in a memoryless web form. One popular choice can be found in [4], [9], which takes ridge regression as the reward estimator. The closed form solution of ridge regression takes a constant memory usage and can be computed efficiently; therefore, we selected ridge regression as our reward estimator.

Another important part of contextual bandit algorithms is the strategy for balancing between exploitation and exploration. A family of algorithms was developed from reinforcement learning, including the ϵ-greedy algorithm [11] and the decreasing-ϵ algorithm. These algorithms use stochastic approaches to balance exploitation and exploration by greedily exploiting the action with the largest estimated reward while randomly exploring actions with a tiny probability of ϵ.

Another popular family of algorithms is the Upper Confidence Bound (UCB) [8] approach, which computes both the estimated reward and the uncertainty level of the estimation, and then chooses the action that comes with the largest sum of the two computed values (so-called upper confidence bound). Linear UCB (LinUCB) [9] is a theoretically [9] and practically [5], [12] successful representative of UCB. It takes ridge regression as the reward estimator, and uses the confidence interval of ridge regression to form the upper confidence bound.

III. Baseline Approaches

The contextual bandit algorithms can be systematically adapted for CBMA as the baseline approaches. The key idea here is to view existing contextual bandit algorithms [4], [9] as a process of *scoring* the actions and then choosing the action with the greatest score. For CBMA, we choose the actions with the first M greatest scores.

A. Stochastic Exploration Algorithms

Stochastic exploration algorithms contain two parts. One is a greedy algorithm for estimating the rewards, and the other is a stochastic exploration procedure that chooses actions uniformly at random. In other words, we can view the stochastic exploration algorithms as scoring ones by using the estimated rewards as the scores in the greedy part, while using random numbers as the scores in the stochastic part.

When using linear ridge regression as the reward estimator for the contextual bandit problem with single action, the chosen action and its observed reward are used to update the ridge regression model. For CBMA, we can update the model with all the chosen action*s* and their observed reward*s*. We denote $\mathbf{r}_t \in \mathbb{R}^M$ as the reward vector, where the m-th component is the reward of the chosen action $a_{t,m}$. We denote $\tilde{\mathbf{X}}_t \in \mathbb{R}^{M \times d}$ as the context matrix, where the m-th row is the context vector of the chosen action $a_{t,m}$. That is, $\tilde{\mathbf{X}}_t = \left[\mathbf{x}_{t,a_{t,1}}, \cdots, \mathbf{x}_{t,a_{t,M}}\right]^\top \in \mathbb{R}^{M \times d}$. Then, we denote $\bar{\mathbf{X}}_t \in \mathbb{R}^{tM \times d}$ as a big context matrix with

all $\tilde{\mathbf{X}}_\tau$ concatenated vertically, for $1 \leq \tau \leq t$ and we denote $\bar{\mathbf{r}}_t \in \mathbb{R}^{tM}$ as a long reward vector with all observed reward vectors $\mathbf{r}_\tau$ concatenated vertically, for $1 \leq \tau \leq t$.

After obtaining $\bar{\mathbf{X}}_t$, and $\bar{\mathbf{r}}_t$ at the end of the t-th iteration, the ridge regression model $\mathbf{w}_{t+1}$ can then be updated as the reward estimator for the next iteration by solving the following optimization problem:

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^d}{\arg\min} \left(\left\| \bar{\mathbf{X}}_t \mathbf{w} - \bar{\mathbf{r}}_t \right\|^2 + \lambda \left\| \mathbf{w} \right\|^2 \right),$$

where λ is a regularization parameter. Then, the closed form solution is

$$\mathbf{w}_{t+1} = \left(\bar{\mathbf{X}}_t^\top \bar{\mathbf{X}}_t + \lambda \mathbf{I}_d \right)^{-1} \left(\bar{\mathbf{X}}_t^\top \bar{\mathbf{r}}_t \right), \tag{1}$$

where $\mathbf{I}_d$ is a $d \times d$ identity matrix.

To simplify the notations, we define $\left(\bar{\mathbf{X}}_t^\top \bar{\mathbf{X}}_t + \lambda \mathbf{I}_d \right)^{-1}$ as $\mathbf{Q}_t$ and define $\bar{\mathbf{X}}_t^\top \bar{\mathbf{r}}_t$ as $\mathbf{b}_t$. Some basic linear algebra derivations show that both $\mathbf{Q}_{t+1}$ and $\mathbf{b}_{t+1}$ can be incrementally updated from $\mathbf{Q}_t$ and $\mathbf{b}_t$ after observing a new pair of $\tilde{\mathbf{X}}_t$ and $\mathbf{r}_t$.

After computing $\mathbf{w}_t$, the greedy part of the stochastic exploration algorithms assigns a score to every action using a score vector $\mathbf{s}_t = \mathbf{X}_t \mathbf{w}_t$. For CBMA, we chose the actions with the M largest scores.

We study two representative stochastic exploration algorithms in this research. The ϵ-greedy algorithm [11] runs the greedy part (ridge regression) described above with a fixed probability of $1 - \epsilon$, and the stochastic part with a probability of ϵ. The decreasing-ϵ algorithm [11] is a variant that iteratively decreases the exploration probability ϵ with a decreasing rate ρ, to enforce more exploration in the beginning and more exploitation in the latter iterations. The stochastic part of both algorithms can be easily taken into the scoring view by taking random numbers as the scores.

In summary, we adapted the ϵ-greedy and decreasing-ϵ algorithms as CBMA approaches by taking the scoring view as well as feeding the reward estimator with all the chosen actions and observed rewards. The special case of $\epsilon = 0$ (no exploration) will be used as the baseline greedy approach and will be compared in the experiments.

B. Linear Upper Confidence Bound

We extended the LinUCB [9] algorithm for CBMA. LinUCB assumes an unknown weight vector $\mathbf{w}^* \in \mathbb{R}^d$, such that the reward $r_{t,k}$ is an random variable with expectation $\mathbf{w}^{*\top} \mathbf{x}_{t,k}$. With this assumption, [9] states that $\mathbf{w}^{*\top} \mathbf{x}_{t,k} \leq \mathbf{w}_t^\top \mathbf{x}_{t,k} + \alpha \sqrt{\mathbf{x}_{t,k}^\top \mathbf{Q}_{t-1} \mathbf{x}_{t,k}}$, holds with a high probability (depending on α), where $\mathbf{w}_t$ comes from linear ridge regression, similar to the greedy part of stochastic exploration algorithms. The right-hand-side of the inequality is called the upper confidence bound, which includes the estimated reward and the uncertainty of the estimate. Note that to compute the upper confidence, we only need to keep $\mathbf{w}_t$ and $\mathbf{Q}_{t-1}$ in the memory. Therefore, the *update* function of LinUCB can be exactly the same as the *update* function of the greedy approach.

If we can view the upper confidence bound of each action as the score that directs the choice of LinUCB, then, LinUCB can be easily adapted for CBMA by taking the first M largest upper confidence bounds.

IV. Pairwise Regression with Upper Confidence Bound

While existing algorithms for the traditional contextual bandit problem can be adapted as baseline approaches for CBMA, the simple adaptations do not deeply consider the specialties of CBMA. We developed an approach that considers these special properties. Note that CBMA aims to choose the *top-rewarded* actions, which is similar to the ranking problem in information retrieval, where the more relevant items are ranked higher than the less irrelevant ones. Our design is based on this similarity, and adds the flavor of UCB to balance between exploitation and exploration.

Many existing works [13], [14] indicate that pairwise methods often work well for ranking. Instead of precisely predicting the relevance of each item, the pairwise methods aim to correctly order each pair of items to construct a proper overall ranking.

To use the pairwise methods for CBMA, we need to rank actions according to the estimated rewards. One particular way based on the linear model is to find some $\mathbf{w} \in \mathbb{R}^d$ that satisfies the following:

$$\operatorname{sign}\left(\mathbf{w}^\top \mathbf{x}_i - \mathbf{w}^\top \mathbf{x}_j \right) = \operatorname{sign}\left(r_i - r_j \right),$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are any two contexts with associated rewards r_i and r_j, respectively. A sufficient but more restricted condition for the equation above is

$$\mathbf{w}^\top \mathbf{x}_i - \mathbf{w}^\top \mathbf{x}_j = r_i - r_j.$$

Note that the equation could also be rewritten as

$$\mathbf{w}^\top \left(\mathbf{x}_i - \mathbf{x}_j \right) = \left(r_i - r_j \right).$$

By defining $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ as the pairwise context vector and $r_{ij} = r_i - r_j$ as the pairwise reward, we see that $\mathbf{w}$ can model the linear relationship between $\mathbf{x}_{ij}$ and r_{ij}, much like how $\mathbf{w}$ in ridge regression can model the linear relationship between $\mathbf{x}_i$ and r_i. Thus, we can obtain a proper $\mathbf{w}$ by performing ridge regression between known pairs of $\mathbf{x}_{ij}$ and r_{ij}.

We denote $\mathbf{r}_t^p$ as the pairwise reward vector that contains all the $\binom{M}{2}$ pairwise rewards we got obtained as components during iteration t, and we denote $\tilde{\mathbf{X}}_t^p$ as the pairwise context matrix that contains all the $\binom{M}{2}$ corresponding pairwise context vectors as rows. If we denote $\bar{\mathbf{X}}_t^p$ as a big matrix with $\tilde{\mathbf{X}}_\tau^p$ concatenated, for $1 \leq \tau \leq t$ and if we denote $\bar{\mathbf{r}}_t^p$

as a long vector with $\mathbf{r}_\tau^p$ concatenated, for $1 \leq \tau \leq t$, we can compute $\mathbf{w}_{t+1}$ by *pairwise* ridge regression as follows:

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^d}{\arg\min} \left(\left\| \bar{\mathbf{X}}_t^p \mathbf{w} - \bar{\mathbf{r}}_t^p \right\|^2 + \lambda \left\| \mathbf{w} \right\|^2 \right).$$

Similar to (1), the closed form solution of pairwise ridge regression is

$$\mathbf{w}_{t+1} = \left(\bar{\mathbf{X}}_t^{p\top} \bar{\mathbf{X}}_t^p + \lambda \mathbf{I}_d \right)^{-1} \left(\bar{\mathbf{X}}_t^{p\top} \bar{\mathbf{r}}_t^p \right). \tag{2}$$

To simplify the notation, we define $\left(\bar{\mathbf{X}}_t^{p\top} \bar{\mathbf{X}}_t^p + \lambda \mathbf{I}_d \right)^{-1}$ as $\mathbf{Q}_t^p$ and $\bar{\mathbf{X}}_t^{p\top} \bar{\mathbf{r}}_t^p$ as $\mathbf{b}_t^p$, where both terms can be incrementally updated like $\mathbf{Q}_t$ and $\mathbf{b}_t$ in the greedy approach.

If the estimated $\mathbf{w}_t$ from the pairwise ridge regression is accurate, we can use the term $\mathbf{w}_t^\top \mathbf{x}_{t,i}$ to rank the actions properly and choose the top-rewarded actions. The term is taken as for exploitation in our proposed Pairwise Regression with Upper Confidence Bound (PairUCB) approach.

The uncertainty term is also needed in PairUCB. We start from taking another look on the uncertainty term of LinUCB The uncertainty term of LinUCB $\sqrt{\mathbf{x}_{t,k}^\top \mathbf{Q}_{t-1} \mathbf{x}_{t,k}}$ measures the similarity between a new context vector $\mathbf{x}_{t,k}$ and the previous observed context vectors. Note that $\mathbf{Q}_{t-1}$ is the inverse matrix of the regularized projection matrix $\left(\bar{\mathbf{X}}_{t-1}^\top \bar{\mathbf{X}}_{t-1} + \lambda \mathbf{I}_d \right)$, which projects vectors to the space spanned by the observed context vectors. If a new context vector is similar to many observed context vectors, then the uncertainty value $\sqrt{\mathbf{x}_{t,k}^\top \mathbf{Q}_{t-1} \mathbf{x}_{t,k}}$ will be small.

In (2), our pairwise reward estimator $\mathbf{w}_{t+1}$ is computed from the pairwise reward vector $\bar{\mathbf{r}}_t^p$ and the pairwise context matrix $\bar{\mathbf{X}}_t^p$. Therefore, from the perspective of pairwise ridge regression, the observed "context vectors" are the pairwise ones $\mathbf{x}_{ij}$, and we can measure the uncertainty level for the pairwise ridge regression by replacing $\mathbf{Q}_{t-1}$ in LinUCB by $\mathbf{Q}_{t-1}^p$, which comes from the pairwise context vectors. That is, the uncertainty term in PairUCB is $\alpha \sqrt{\mathbf{x}_{t,k}^\top \mathbf{Q}_{t-1}^p \mathbf{x}_{t,k}}$.

In summary, PairUCB computes $\mathbf{w}_t$ using pairwise ridge regression, and then chooses the action with the first-M largest UCB scores $\mathbf{w}_t^\top \mathbf{x}_{t,k} + \alpha \sqrt{\mathbf{x}_{t,k}^\top \mathbf{Q}_{t-1}^p \mathbf{x}_{t,k}}$. Pairwise ridge regression respects the similarity between CBMA and the ranking problem, and it is expected to work better than naive adaptations of existing contextual bandit algorithms such as LinUCB.

V. Experiment

A. Dataset

We used three kinds of datasets to simulate the CBMA problem, including the Yahoo Today article recommendation dataset, artificial datasets, and regression datasets.

The Yahoo Today article recommendation dataset was collected from the Yahoo! Front Page. This dataset was proposed in [4], and to the best of our knowledge, it is the only public dataset that is available for the contextual bandit problem with a single action. Yahoo! randomly inserts recommended article titles on the Yahoo! front page hourly. Users can click on these titles to read the detail of the corresponding article. Yahoo! records only the click event for the headline article. Each log contains a time stamp, the context of the rticle and the user, and whether the user clicked on the article or not.

To simulate the CBMA problem with the Yahoo! dataset, we set the reward as a binary indicator for the click event. We found that 3.7% of the click event indicator were set to 1. We used articles as context, and we considered readers of the same article as the same user. We ran the CBMA procedure separately for each user. We grouped the articles with temporal contiguity nto sets of 10 articles and sets of 20 articles, which we denoted as yahoo10 and yahoo20, respectively. During the grouping, we randomly dropped some logs, and we performed the experiment 10 times. To ensure that we can run enough iterations for each user, we only considered the 466 users who had more than 1000 logs.

For the artificial dataset, we generated a unit vector $\mathbf{w}^* \in \mathbb{R}^d$ and $\mathbf{x}_i \in \mathbb{R}^d$ with $\left\| \mathbf{x}_i \right\|^2 \leq 1$ for $i = 1 \dots N$ as context vectors. Then we set the reward $r_i = \mathbf{x}_i^\top \mathbf{w}^* + \nu$ for each context vector, where ν is a random noise in the range of $[0.05, -0.05]$ thereby providing us with N (reward, context) tuples.

To simulate the CBMA problem, we grouped K tuples together as the potential action set for each of the T iterations. We used these N tuples to generate four artificial datasets, artR, artB5, artB10, and artB15. The dataset artR directly used the N tuples, so the reward type is a real number. In order to simulate a binary reward situation like the Yahoo! dataset, we sorted these tuples according to their reward value. Then we set the rewards of the first 5%, 10%, and 15% tuples as 1, which we denoted as artB5, artB10 and artB15, respectively. The rewards of the remaining tuples were set to 0.

In many online applications, the phenomenon of concept drifting, where the reward distribution changes, occurs. In order to simulate this phenomenon, we create an artificial dataset, named "drift." We first generated two hidden unit vectors $\mathbf{w}_1^*$ and $\mathbf{w}_2^*$. Then, we used different reward generators for each iteration. The reward generator $\mathbf{w}_t^*$ of the t-th iteration is as follows:

$$\mathbf{w}_t^* = \left(1 - \frac{t}{T} \right) \mathbf{w}_1^* + \frac{t}{T} \mathbf{w}_2^*.$$

Note that the reward generator slowly transforms from $\mathbf{w}_1^*$ to $\mathbf{w}_2^*$. Then we set the reward of the k-th action on t-th iteration to be $r_{t,k} = \mathbf{x}_{t,k}^\top \mathbf{w}_t^* + \nu$. Again, ν is a random noise.

We downloaded six regression datasets from the libsvm website [15]. We normalized their context vectors, such that the ℓ_2-norm of these context vectors are less than or equal

Table I: Dataset characteristics

	N	d	K	M	T
artR	50000	10	50	5	1000
artB{5,10,15}	50000	10	50	5	1000
drift	50000	10	50	5	1000
abalone	4177	8	40	4	100
bodyfat	252	14	5	3	50
cpusmall	8192	12	80	8	100
housing	506	13	5	3	100
mg	1385	6	13	4	100
mpg	392	7	6	3	65
space	3107	6	30	6	100
yahoo10	8402050	6	10	5	840147
yahoo20	8399840	6	20	5	419961

to 1. We used their original labels as rewards. To simulate the CBMA problem in this case, we grouped several (reward, context) tuples together to be the potential action set for each iteration.

For artificial datasets and regression datasets, the result varies with different data sequences and different group combinations. In order to fairly compare the performance, we randomly permuted each dataset 20 times and reported the means and standard errors as our results.

In Table I, we show the characteristics for each dataset. N is the number of (reward, context) tuples. d is the dimension of action context vector. K is the size of the action candidate set per iteration. M is the number of actions to be chosen at each iteration. T is the total number of iterations we run on the dataset. Note that $T = N/K$. To simulate the CBMA problem with a different ratio between K and M, and to simulate the CBMA problem with enough iterations we set different K and M for each dataset.

B. Setup

In order to fairly compare each algorithm, we reported the results using the best parameter combination for each algorithm. The parameters are ϵ for ϵ-greedy and decreasing-ϵ, ρ for the decreasing rate for decreasing-ϵ, α for LinUCB and PairUCB. We selected these parameters from the following ranges:

$$\begin{aligned}\epsilon &\in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}\\ \rho &\in \{0.95, 0.9, 0.85\}\\ \alpha &\in \{0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2\}.\end{aligned}$$

Note that, when $\alpha = 0$, LinUCB is simply greedy.

We use the averaged cumulative regret (ACR) as our measurement criterion. By comparing the cumulative reward obtained through algorithm $\mathcal{A}$ and the largest possible cumulative reward achieved previously, we defined the ACR of algorithm $\mathcal{A}$ as follows:

$$\mathrm{ACR}_{\mathcal{A}}(T) = \frac{1}{TM}\sum_{t=1}^{T}\left(\sum_{m=1}^{M}\mathrm{rw}\left(a^*_{t,m}\right) - \sum_{m=1}^{M}\mathrm{rw}\left(a_{t,m}\right)\right),$$

where $a^*_{t,m}$ is the action with the m-th largest reward on the t-th iteration. The smaller the ACR is, the better the algorithm performs.

C. Performance Comparison

Table II shows that our proposed PairUCB approach performs the best on most artificial and regression datasets and is slightly better than other approaches on the yahoo10 and yahoo20 datasets.

For most datasets, the performance of greedy, ϵ-greedy, and decreasing-ϵ with different ϵ are similar to Figure 1a. When $\epsilon = 0$ (the left most point of each line), these approaches become the greedy approach. The performance worsens with larger values of ϵ. This phenomenon indicates that the stochastic exploration approach is not helpful and may even be harmful for the CBMA problem.

We compared the performances of the UCB approaches in Figures 1b and 1c. Note that the leftmost point of LinUCB is the greedy approach. For most datasets we found that, with all α values in our searching range, PairUCB outperforms LinUCB as shown in Figure 1b. With a little UCB exploration ($\alpha > 0$), both PairUCB and LinUCB could achieve better performance on most datasets. In other datasets that do not need exploration like Figure 1c, such as mg and space, we observe that PairUCB is less sensitive to the parameter α than LinUCB, which demonstrates another benefit of PairUCB.

In summary, we found that the PairUCB approach often outperforms other approaches. In addition, PairUCB appears less sensitive to the parameter α than LinUCB. The results show that PairUCB with a small α is the best choice for the CBMA problem.

VI. Conclusion

We formalized a new CBMA problem, which extends the traditional contextual bandit problem to allow multiple actions. This extension can model online applications more generally. We reviewed existing contextual bandit algorithms and adapted them for the CBMA problem. We also proposed a new PairUCB approach, which addresses the properties of the CBMA problem. Then, we simulated the CBMA problem on fourteen artificial and real-world datasets. The experiment results demonstrated that PairUCB achieves a better performance than other approaches on most datasets.

References

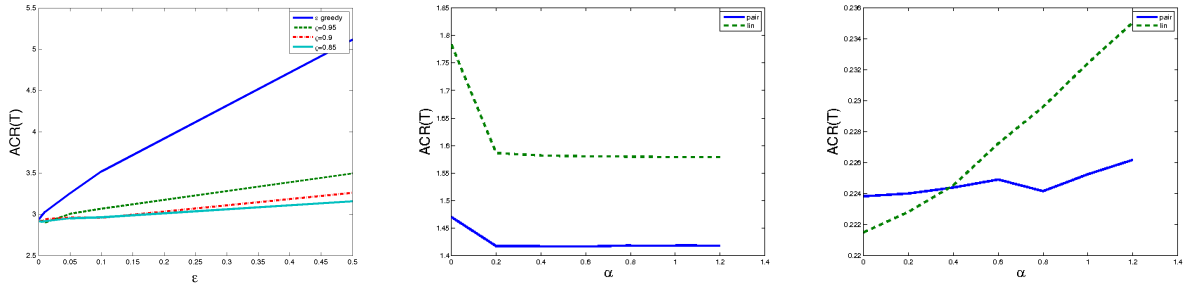
[1] C.-C. Wang, S. R. Kulkarni, and H. V. Poor, "Bandit problems with side observations," *Automatic Control, IEEE Transactions on*, vol. 50, no. 3, pp. 338–355, 2005.

[2] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski, "Bandits for taxonomies: A model-based approach." in *SIAM on DATA MINING*, 2007.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

Table II: ACR for each algorithms. The bold numbers indicate the best performance for each dataset

	greedy	ϵ-greedy	decreasing-ϵ	LinUCB	PairUCB
artR	0.1902 ± 0.0052	0.1904 ± 0.0051	0.1901 ± 0.0051	0.1904 ± 0.0051	$\mathbf{0.1311 \pm 0.0187}$
artB5	0.1534 ± 0.0460	0.1537 ± 0.0038	0.1529 ± 0.0039	0.1618 ± 0.0038	$\mathbf{0.0885 \pm 0.0168}$
artB10	0.2789 ± 0.0123	0.2791 ± 0.0126	0.2785 ± 0.0125	0.2891 ± 0.0126	$\mathbf{0.1785 \pm 0.0330}$
artB15	0.2961 ± 0.0125	0.2967 ± 0.0112	0.2950 ± 0.0114	0.3044 ± 0.0112	$\mathbf{0.1924 \pm 0.0325}$
drift	0.4469 ± 0.0439	0.4375 ± 0.0417	0.4370 ± 0.0427	0.4350 ± 0.0433	$\mathbf{0.2367 \pm 0.1231}$
abalone	2.9168 ± 0.0322	2.9408 ± 0.0386	2.9003 ± 0.0354	2.8704 ± 0.0321	$\mathbf{2.6237 \pm 0.0417}$
bodyfat	0.0041 ± 0.0002	0.0040 ± 0.0002	0.0040 ± 0.0002	0.0041 ± 0.0002	$\mathbf{0.0012 \pm 0.0001}$
cpusamll	1.7829 ± 0.0997	1.7874 ± 0.1076	1.7660 ± 0.0857	1.5790 ± 0.0461	$\mathbf{1.4129 \pm 0.0164}$
housing	1.7652 ± 0.0479	1.7660 ± 0.0476	1.7640 ± 0.0475	1.7527 ± 0.0482	$\mathbf{1.6694 \pm 0.0576}$
mg	0.0451 ± 0.0009	0.0450 ± 0.0008	0.0450 ± 0.0008	0.0454 ± 0.0008	$\mathbf{0.0438 \pm 0.0007}$
mpg	0.7864 ± 0.0340	0.7901 ± 0.0339	0.7864 ± 0.0340	0.7878 ± 0.0340	$\mathbf{0.7464 \pm 0.0288}$
space	0.2215 ± 0.0008	0.2215 ± 0.0008	$\mathbf{0.2212 \pm 0.0008}$	0.2228 ± 0.0008	0.2238 ± 0.0008
yahoo10	0.1835 ± 0.0000	0.1834 ± 0.0001	0.1834 ± 0.0000	0.1834 ± 0.0000	$\mathbf{0.1833 \pm 0.0001}$
yahoo20	0.5504 ± 0.0001	0.5504 ± 0.0001	0.5502 ± 0.0001	0.5500 ± 0.0001	$\mathbf{0.5500 \pm 0.0001}$



(a) abalone (b) cpusmall (c) space

Figure 1: (a) ACR for ϵ-greedy (blue solid) and decreasing-ϵ with different ρ on abalone dataset with different ϵ; (b) (c) ACR for LinUCB (green dashed), PairUCB (blue solid) with different α on cpusmall and space dataset

[4] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 661–670.

[5] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 297–306.

[6] M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient optimal learning for contextual bandits," *arXiv preprint arXiv:1106.2369*, 2011.

[7] A. Beygelzimer, J. Langford, L. Li, L. Reyzin, and R. E. Schapire, "Contextual bandit algorithms with supervised learning guarantees," *arXiv preprint arXiv:1002.4058*, 2010.

[8] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *Journal of Machine Learning Research*, vol. 3, pp. 397–422, 2003.

[9] W. Chu, L. Li, L. Reyzin, and R. E. Schapire, "Contextual bandits with linear payoff functions," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[10] S. Kale, L. Reyzin, and R. Schapire, "Non-stochastic bandit slate problems," *Advances in Neural Information Processing Systems (NIPS)*, pp. 1054–1062, 2010.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998. [Online]. Available: http://www.cs.ualberta.ca/Esutton/book/ebook/the-book.html

[12] K.-C. Chou and H.-T. Lin, "Balancing between estimated reward and uncertainty during news article recommendation for ICML 2012 exploration and exploitation challenge," 2012.

[13] U. Brefeld and T. Scheffer, "Auc maximizing support vector learning," in *Proceedings of the ICML 2005 workshop on ROC Analysis in Machine Learning*, 2005.

[14] E. L. Mencia and J. Furnkranz, "Pairwise learning of multilabel classifications with perceptrons," in *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN-08)*. IEEE, 2008, pp. 2899–2906.

[15] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.