Non-Stationary Contextual Multi-Armed Bandit with Application in Online Recommendations.

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Master of Science

by

Muhammad Ammar Hassan

August

2015

APPROVAL SHEET

The thesis

is submitted in partial fulfillment of the requirements

for the degree of

Master of Science

AUTHOR

The thesis has been read and approved by the examining committee:

Peter A. Beling

Advisor

Stephen D. Patek

Hongning Wang

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science

August

2015

Non-stationary contextual multi-armed bandits

Muhammad Ammar Hassan

August 3, 2015

Supervisor

Peter A. Beiling

Stephen D. Patek

Committee members

Hongning Wang

Department of Systems and Information Engineering

University of Virginia

Contents

Abstract

The only constant in the online social behavior is forever changing user intent and preferences. These changes could be inspired by myriad of factors but still have an overall trend e.g. todays popular news will be stale tomorrow etc. Such patterns are especially noticeable in viral trends where an immediate gain in popularity is followed by gradual lost of holistic interest. In this study we focus on design of a recommendation system which accounts for this non-stationary behavior of declining popularity. Contextual multi-armed bandit (contextual MAB) is a popular framework for learning user behavior and personalized recommendations based on the past behavior. Fundamentally MAB solves the exploitation-exploration dilemma which aims at minimal guided experimentation required to gain certain level of confidence in its recommendation. Traditionally contextual MABs (e.g. LinUCB) have been used to model stationary user behavior that is not appropriate for the target environment where LinUCB can accumulate linear regret. Here we extend this LinUCB type algorithm to model a decaying environment. We present three algorithms with variable levels of specificity in the assumptions they make about the non-stationary environment. We show by simulation the effectiveness of our methods which illustrates the usefulness of modeling meta-trends in user behavior.

1 Introduction

In this thesis our focus is on line algorithms in non-stationary environments. On line algorithms are widely applicable in real life and offer many applications based on fast growing corpus of information. We make recommendation systems our prime example for study in this research although the results derived in simulation and theory are completely independent of the application which is only there to make the framework comprehensible. Multi-armed bandit is one such algorithm and is the focus of our study. Lets use recommendation systems to show case the different aspects of the Multi-armed bandit problem. On line recommendation systems are getting bigger and more complex with time, as more information is cached in the browsers, there is plenty of opportunities for recommendation engines to personalize the suggestions to user's taste. Apart from utilizing history and other information about the user, on line systems present the challenge of on the fly decision making, which implies that algorithms have to be designed around providing quick service to a heavy traffic and obviously this should avoid bulky recalculations. Personalized recommendation is a learning task, any learning algorithm will require a training dataset to build a model. In online learning this data set can be carefully collected on a portion of the incoming traffic, however gathering learning data is a form of exploration where a new kind of user is presented with a suggestion and the response is recorded

to form a supervised classification task. This exploration phase is expensive since it can cost the system a potential user, hence an ideal algorithm will also optimize for minimal exploration. On the other hand this exploration is necessary to form a well rounded dataset which covers many dimensions of user behaviors, and the larger the dataset better the model learned by the system (exploitation). This hints at the fundamental exploitation vs. exploration dilemma which is at the root of the multi-armed bandit problems. Multi-armed bandits are specially attractive since they provide nice theoretical results of lower and upper bounds of regret for any type of a problem and the algorithm respectively. The theory makes sure that no more exploration is done than required to achieve good over all reward and further more the exploration should fall off with time. Another view of this trade off is balancing long and short term rewards, while exploitation secures short term reward, exploration aims at long terms reward by improving the model. Contextual bandits provide the framework to combine the learning task while considering the exploration-exploitation trade off. This problem is challenging enough which is only further complicated by the fact that the user preferences or action rewards can be non-stationary. This takes us into a realm of an entirely new and unexplored territory which has been avoided either because it is beyond the reach of current mathematical tools of analysis or in most general case the solutions are trivial with linear regret. There is currently (to the best of our knowledge) no algorithm that takes on the non-stationary environment while considering the contextual information (user history etc.). This study is an exploration into this new space of research where we characterize a specific non-stationary setting while a linear function maps the contextual information to the reward with a meta-trend of gradual decay. This is indeed a difficult task and we present three different methods which either follow a specific idea (TUCB follow UCB style approach) or extend a non-stationary approach taken in simple UCB algorithms (decay LinUCB).

We start our discussion with simple multi-armed bandit (MAB) setting where an agent (algorithm) repeatedly decides between multiple actions (recommendations) with the goal to find the action with highest reward. On selecting an action, the agent receives a reward according to a fixed but unknown distribution specific to that action. This is called stochastic MAB since the feedback to the agent is noisy. The MAB maximizes the reward over some given horizon, T trails. The most popular strategy proposed for this problem is based on calculating an index called upper confidence bound (UCB) for every action at trial t and choosing the action with the highest index. The UCB incorporates, in its index and hence the decision, the estimate of reward and confidence in the estimation simultaneously. At the beginning, decisions are partial to confidence and decision weight gradually shifts to the estimate as the confidence width shrinks with accumulation of more data. This ensures that each action gets tested enough so that the algorithm does not converge to

an inferior action while on the other hand shrinking of the confidence allows for limited exploration with time. The performance of MAB is measured by the regret of the algorithm's missed opportunity on reward by choosing a sub-optimal action. In the work on MABs the regret is theoretically bounded above for any algorithm and if the upper bound is sub-linear with respect to the time horizon the algorithm is said to be learning, otherwise the algorithm is not much better than a random strategy. For regret to be sub-linear it is also necessary that exploration should be trimmed as the time goes by.

For most real world applications solving the exploration-exploitation tradeoff alone is not enough. For example in recommendations domain a learning algorithm should train on the data and the exploration should be guided based on the specifics of this algorithm; its confidence bounds. Simple MAB does not provide this flexibility which only focuses on the tradeoff. For these applications we have side information e.g. usage history, demographic information specifically for big companies; who can track their customers with lower granularity since their users will sustain unique connections by logging into the sites e.g. Walmart, Amazon etc. This side information can be used to build models of user preferences.

Contextual MAB provide the framework e.g. LinUCB [6] for learning the user preferences while making minimal exploration. LinUCB uses Ridge regression to model the relation between a context vector (a transformed history mapping) and the user response (binary feedback whether the recommendation was accepted or not, silence is taken as rejection). This relation is assumed to be linear and fixed temporally and hence implies the 'Lin' in LinUCB, further more the UCB part of the algorithm queries the confidence of the regression model and chooses the action corresponding to the highest confidence bound. LinUCB has been shown to work well, boosting the click through rate of news articles for Yahoo front page module dataset.

In this work we extend LinUCB for decaying popularity or click through rate of online recommendations. This is a challenging task since the implicit assumption of static environment is violated. Many algorithms can be thought for this environment based off the strength of assumptions on the specificity of the environment. For most general scenarios, we proposed Decay LinUCB algorithm, where no more assumptions about the environment are made other than the changing property. Under this general assumption, our algorithm will slowly forgets the past(gives decreasing weight to the past examples). Regret analysis shows that this algorithm is linear with regret. This is not surprising considering the Decay LinUCB's limited information of the environment. Aiming to improve the regret property, we proposed Restart LinUCB, where we will restart LinUCB algorithm regularly, assuming stationarity within intervals of increasing lengths. This algorithm may sounds naive, but with the exponential decaying environment assumption, we carefully control the length of restart intervals using doubling tricks. By making use of the stationarity within intervals, and the

doubling trick, a nice regret upper bound can be proved, which is sublinear. Further more, making more specification about the environment, we proposed Temporal LinUCB(TUCB), where we explicitly modeled this exponentially decay environment. This specific algorithm will parametrize the environment and hence can give the best performance. Following the logic of proofs for Decay LinUCB and Restart LinUCB, it is possible that we can get an even tighter bound of regret for TUCB. The non-stationary setting however complicates using such an estimator in place of Ridge regression that aims to simultaneously learn two parameters; linear mapping of user interests and potentially a non-linear of decaying reward or user interest in aging news. We will describe these three algorithms in details in Section 4. And theoretical guarantee of regret upper bound of this algorithm is provided is Section 5. All these algorithms with their theoretical properties, practical advantages and results are detailed in next sections.

2 Literature Review

Here we review the types of bandit problems and show why certain types are not fit for our application. In literature there are 3 fundamental types of explore and exploit strategies that differ in the kind of environments they are dealing with [15]. The difference manifests in the way the environment generates the reward. We review these types as they are generic in principal and extend to all kinds of MABP including contextual MABP.

2.1 Regret

Bandit algorithms can be evaluated based on the total reward they collect. However the more useful quantity from theoretical perspective turns out to be regret. Regret is the lost opportunity for not choosing the 'optimal' arm. The optimal arm is defined as the action with the highest expected reward or the arm that the algorithm will choose if it knew the true expected rewards for all the action. Definition of regret is application specific. For example in UCB1 for stochastic bandit the regret is defined as the number of times the algorithm chooses the sub-optimal action.

2.2 Stochastic bandits

In stochastic bandits the environment is supposed to be simple where it generates reward by drawing a sample from a static distribution. This reward sample is independent from the past samples and also other machine distributions. These distributions are unknown to the agent and the best long term decision is to

select the machine or action with highest expected value μ_i. E.g in case of two biased coins with different probability of heads the rewards will follow a Bernoulli distribution. The objective in this example will be to discover and repeatedly choose the coin with higher probability (expected value) of lets say heads. In stochastic bandits the desired objective function is called the pseudo-regret given by the equation 1 [13]. It was proved that the lower bound for this regret is $O(\log T)$ where T is the number of turns the agent gets and the strategy called UCB1 was shown that achieves this regret described in section 2.2.1. This low regret was possible with the strong assumption of stochastic environment. Low regret makes this strategy attractive for many applications.

$$\overline{R}_n = n\mu^* - E\sum_{t=1}^{n} \mu_{It} \tag{1}$$

$$\mu^* \overset{\text{def}}{=} \max_{1 \geq i \geq K} \mu_i$$

2.2.1 Upper Confidence Bounds

This strategy relies on the statistical confidence in the estimation of the means from the prior seen examples. This strategy calculates the upper confidence bound on the standard deviation of the expectation [13]. The following equation gives the formula for UCB1 strategy given by [13] with best logarithmic regret. Let n be the number of turns so far and n_i is the times the action i is tried, then choose the action with,

$$a_t = arg\max_{i\ \epsilon 1,\ldots,K}\ \hat{\mu} + \alpha\hat{\sigma}_i = arg\max_{i\ \epsilon 1,\ldots,K}\ \bar{x}_i + \sqrt{\frac{2\ln n}{n_i}}$$

2.2.2 ϵ-Greedy

ϵ-greedy is a simple and popular heuristic algorithm for stochastic bandit problems. At each repeated stage or turn the agent choose to pull the arm with maximum empirical mean with probability $1-\epsilon$ (exploits) and a randomly chosen arm with probability ϵ (explores). This algorithm is naive in sense that it explores with the same rate despite the confidence in current estimation of the expectations of rewards. Another variant of the ϵ-greedy family is the strategy with discounting ϵ with a factor γ. However, in all these algorithms the constant ϵ or the discount factor γ is chosen experimentally and has nothing to do with the current estimates. Another similar algorithm is the epoch-greedy algorithm given by [20], which after single exploration exploits for an epoch of length determined by the quality of upper bound on the regret.

2.3 Adversarial or non-stochastic bandits

In adversarial bandits also called the non-stochastic bandits the reward structure is assumed to be adversarial, for example in case of a casino deciding the rewards for a gambler such that the gambler's wins are lot less than what s/he could hope for. The adversary decides the rewards before the game even begins [18]. This kind of model takes a pessimistic view of the world where the stochastic case can be considered an optimistic one. This view borrows itself from the game theory context where an agent plays repeated games against an opponent and follows minimax strategies.

In adversarial bandits the agent can not play a deterministic strategy as this can be guessed by the adversary resulting in the agent choosing worst reward for ever. So the agent takes refuge in randomization to escape the fate of predictability. The popular strategy given by [12] is called Exp3 as described in subsection.

2.3.1 Exp3

The Exp3 family of strategies uses weights on the actions and then chooses action probabilistically on these weights. The weights contain two parts, one is the random exploration part is the minimum probability of selecting an arm uniformly; $\frac{1}{K}$. The other part assign weights to actions proportional to the payoffs produced by those actions in the past. Let $x_j(t)$ be the reward of choosing action j at time t and $w_j(t)$ be the weight of that action.

$$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K} \qquad i = 1, ..., K.$$

Where the weights are updated by the following rules.

$$\hat{x}_j(t) = \begin{cases} x_j(t)/p_j(t) & if\ j = i_t \\ 0 & otherwise \end{cases}$$

$$w_j(t+1) = w_j(t)\ exp\ (\gamma \hat{x}_j(t)/K).$$

γ controls the exploration of the random arm while the $1-\gamma$ the exploitation of the best action. This model considers the worst possible adversary while optimizing the weak regret which could be undesirable since the sequence of actions might not be as bad as the worst. However, Exp3 is a safer strategy if the reward structure is not i.i.d. and we detect the environment is hostile against the agent [15]. In fact, UCB can have linear regret in certain deterministic reward sequences. This an extreme version of non-stationary bandits.

2.4 Markovian bandits

In Markovian bandits each slot machine is itself a Markov process with its own state space. On selecting that machine the Markov process undergoes a transition and produces a reward where only the state of the selected machine changes. Generally the transition matrices M_i for the Markov processes are known in advance and the problem turns out to be an optimization one that can be computed by dynamic programming [15]. Gittens gave a computationally efficient greedy policy for calculating the optimal policy in his seminal paper [26].

A special case of Markovian bandits is Bayesian bandits. Here the rewards are assumed to be generated by some parametric distribution with known priors. On further observations of rewards the posterior distribution is updated which corresponds to the transition in the Markovian Bandits [15]. Recently Bayesian bandits have become popular because of their capability of incorporating prior information into the calculations. One of the interesting applications of Bayesian bandits is automatic tunning [25]. Another variant is restless bandits in which on action the state of all the machine changes. This problem is notoriously hard and in fact intractable.

2.5 Contextual Bandits

Contextual bandits are bandits with side information called the context. Context will mean different things in different applications. For example in an article recommendation system the context is the information about the user or the article that will be presented to the user while in the treatment selection the context can be the medical history of the patient. The context makes these models applicable and more useful in real life examples. Both adversarial and stochastic bandit problems have their counterparts in contextual settings. In stochastic contextual bandits the matter becomes of finding a mapping from the context to the actions. This can be posed as a supervised learning problem. Lets look at the following application that we will use to test our ideas.

2.5.1 Article recommendation

Personalized news article recommendation is a challenging problem where in the real world application the dynamic nature of emerging news render the traditional recommendation techniques like collaborative filtering [23] impractical. Such problems can be modeled as contextual multi-armed bandits. One such practical application is the Yahoo! front page article recommendation, which is the challenge of presenting

interesting news article to the user.

Each user and the article is represented by a feature vector which defines the context at any time. The features are extracted from user and article information by conjoint analysis [21]. The feedback of the user is collected by recording the response; the article is clicked or not. With partial feedback and dynamic environment this problem naturally lends itself to the formulation of a contextual multi-armed bandit problem. This problem is further modeled in stochastic settings as the practicality of the problem renders adversarial and Bayesian approaches ineffective. In adversarial context the computational complexity is exponential in the number of features while the Bayesian approach require extensive off line engineering to get good priors [6]. Here we use the notation of [6].

Notation. Let trail $t = 1, 2, 3, ...$ the given action $a_t \epsilon A_t$. The user is represented by feature vector $\mathbf{u}_t$. The context is $\mathbf{x}_{t,a}$ a d dimensional vector that covers both user and the article. As the action a_t a reward r_{t,a_t} is observed. The expectation of payoff depends on both user and the action taken. The choice and reward is the tuple $(\mathbf{x}_{t,a_t}, a_t, r_{t,a_t})$.

The algorithm proposed by [6] is called LinUCB as the assumption is that the payoff expectation is linear function of the context. Let the coefficient of this linear function be $\boldsymbol{\theta}_a^*$ then for all t,

$$E[r_{t,a}|\mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^\top \boldsymbol{\theta}_a^* \tag{2}$$

$\boldsymbol{\theta}_t^*$ can be learned by supervised learning like linear regression or better ridge regression. Let $\mathbf{D}_a$ be the d dimensional m contexts seen so far for article a while $\mathbf{c}_a$ is the corresponding clicks vector for each example. The ridge regression formula coefficient estimation is given as,

$$\hat{\boldsymbol{\theta}}_a = (\mathbf{D}_a^\top \mathbf{D}_a + \mathbf{I}_d)^{-1} \mathbf{D}_a^\top \mathbf{c}_a \tag{3}$$

This is called the disjoint model as the $\boldsymbol{\theta}_a$ only depends on the article. In this scenario the confidence bound on the estimate of the expected value is given by,

$$|\mathbf{x}_{t,a}^\top \hat{\boldsymbol{\theta}}_a - E[r_{t,a}|\mathbf{x}_{t,a}]| \leq \alpha \sqrt{\mathbf{x}_{t,a}^\top (\mathbf{D}_a^\top \mathbf{D}_a + \mathbf{I}_d)^{-1} \mathbf{x}_{t,a}}$$

This confidence bound is used in calculating the UCB index. The action with the highest bound is selected

in each trial or turn t,

$$a_t \stackrel{\text{def}}{=} arg\max_{a \epsilon A_t}(\mathbf{x}_{t,a}^{\top}\hat{\boldsymbol{\theta}}_a + \alpha\sqrt{\mathbf{x}_{t,a}^{\top}(\mathbf{D}_a^{\top}\mathbf{D}_a + \mathbf{I}_d)^{-1}\mathbf{x}_{t,a}})$$

The $\boldsymbol{\theta}_a^*$ estimated in the above formulation reflects the preferences of users belonging to context $\mathbf{c}_a$. Here an implicit assumption is made about the preferences in that these are not dependent on time or trial t or in other words these preferences are static over time.

2.6 Non-stationary bandits

In this section we turn to the more relevant part of the thesis. We describe the research relevant to the non-stationary bandits. This review provides the previous results which although do not provide any direct solutions but give insights into the working of the bandits and also show the gap that can not be filled by any present methods.

2.6.1 Abrupt changing environments

In [17] non-stationary environment is defined as "the rewards for arm i are modeled by a sequence of independent random variables from potentially different distributions (unknown to user) which may vary across time". However they only consider the situation where the environment is abruptly changing at unknown time instants called breakpoints while it remains stationary during intervals between breakpoints. For this abrupt changing environment they propose "Sliding-window UCB" where only the observation inside the current window are considered. They also use "Decay-UCB", the past observations are decayed with exponential decay γ. They show the analysis of regret for both algorithms and find that "Sliding-Window UCB" does better. Also they show that UCB like policy can not attain regret smaller than $\frac{T}{\log T}$ in presence of break points. The analysis is done by development of a novel deviation inequality for self-normalized averages with random number of summands. They also prove the lower bound of $\sqrt{T}$ for the non-stationary case.

2.6.1.1 Proof of Decay-UCB

For any action a define the empirical discounted mean as follows.

$$\overline{y}_t(\gamma, a) = \frac{1}{N_t(\gamma, a)}\sum_{s=1}^{t}\gamma^{t-s}y_s(a)\mathbb{1}_{\{a_s=a\}}, \quad N_t(\gamma, a) = \sum_{s=1}^{t}\gamma^{t-s}\mathbb{1}_{\{a_s=a\}}$$

$$c_t(\gamma, a) = 2B\sqrt{\frac{\xi \log n_t(\gamma)}{N_t(\gamma, a)}}, \quad n_t(\gamma, a) = \sum_{a \in \mathcal{A}} N_t(\gamma, a) \tag{4}$$

Play action with,

$$a_t = \arg\max_{a \in \mathcal{A}} \overline{y}_t(\gamma, a) + c_t(\gamma, a)$$

The proof is similar to that of Auer [13] except for the two main differences. Due to discounting the empirical mean is now a biased estimator. Second they designed a deviation inequality instead using Chernoff-Hoeffding's inequality. In Auer UCB1 proof follows the following approach,

1. Separate the horizon into two sequences, first when the UCB1 is not confident and second when UCB1 is confident. By confident we mean this

$$\mu^* - \mu_a \geq +2c_{t,s_i}$$

   i.e. Each action has been played enough such that difference of means of optimal and sub-optimal arms is greater than the confidence bound for that arm.

2. Regret on first sequence is simply bounded by its length, which is inversely proportional to the difference of the means squared.

3. In the second sequence algorithm only makes a mistake if either the sub-optimal arm is heavily over-estimated or the optimal arm is heavily under estimated. Both of these events are due to randomness and are rare events and Chernof-Hoeffdings inequality can be applied to determine the bounds.

The proof for Decay UCB follows the same approach.

1. Separate the horizon in to two sequences. The first where the algorithm is not confident. Unlike in UCB1 these sequences can interleave.

2. Bound the first sequence by its length

3. Divide the second sequence into parts,first $D(\gamma)$ trials after a break point and the rest of trials. The first $D(\gamma)$ trials will give erroneous estimation and the regret is bounded by the number of breakpoints multiplied by $D(\gamma)$.

4. The rest of trials, after $D(\gamma)$ of a break point are when the algorithm is confident and only makes mistake due to under estimation of optimal arm or over estimation of the sub-optimal arm.

5. This time the errors in estimation are due to randomness and bias of the discounted mean. First the bias is bounded by $\frac{1}{2}c_t(\gamma, a)$. Then the stochastic process is bounded using the specially proved inequality which deals with self-normalized means.

If there are $\mathcal{O}(T^{\beta})$ breakpoints, the regret acheived is $\mathcal{O}(T^{\frac{(1+\beta)}{2}} \log T)$.

2.6.1.2 Sliding Window UCB Proof

The proof follows exactly the same steps as of Decay UCB except for the bias disappears in this case, which makes it relatively easier. The upper bound on regret is $\mathcal{O}(T^{\frac{1+\beta}{2}} \sqrt{\log(T)})$ which is slightly better than decay UCB.

2.6.2 Brownian restless bandits

Upfal etal in [24] consider the situation where mean for each arm/action is a bounded stochastic process. The means drift with certain volatility (σ_a, variance of Gaussian distribution) with in reflective boundaries $\mu \in [0, 1]$.

They proposed a UCB type algorithm given in equ. 5. Let $N_a(t)$ be the times action a was played until trial t and $\mu_a(t)$ be the empirical average reward for action a till time t.

$$a_t = \arg\max_{a \in \mathcal{A}} \quad [\overline{\mu}_a(t) + \sqrt{2\ln(t)/N_a(t)} + \sigma_a \sqrt{8t \log t}] \tag{5}$$

This algorithm looks very similar to UCB1 except for the last component including σ_a. This component bounds the drift of the means for each arm given the volatility. The drift is carefully converted into a martingale process and Optional Stopping Theorem is used to show that $\mu_a(t) = \mu_a$ i.e. the final state of means is the starting state of the means.

The type of regret chosen for evaluation of strategy in this case is "steady state regret". It is the maximal average regret over any sub-sequence (consecutive trials) in all the trials. Further more the regret is concerned with respect to a non-stationary oracle that knows the optimal arm. The steady-state regret is shown to be $\mathcal{O}(k\sigma_{av})$.

2.6.3 Solving Non-Stationary Bandit Problems by Random sampling from sibling Kalman Filters

[16] studies an empirical approach for solving non-stationary normal distributions for multi-armed bandit problem using Bayesian methods. The upside of this approach is that it avoids its inherent computational complexity by relying on updating hyper parameters of sibling Kalman filters and on random sampling from theses posteriors. Here the transition variance σ_{tr} is assumed to be known in advance and is used to track the mean and variance of the respective Gaussian distributions using formula for Kalman filters. The action is picked based of the random samples from posterior distributions.

2.6.4 Optimal Exploration-Exploitation in a multi-armed bandit problem with non-stationary rewards

Reward means can be non-stationary with a bound on the total variation of expected rewards. There is no restriction in the evolution of means so long they obey the variation bound. This allows a host of sequential trajectories of expected rewards to occur. They develop an algorithm called REXP3 which is just EXP3 with restart $\frac{T}{\Delta_T}$ times where Δ_T is the batch size for which one instance of EXP3 is run. The regret considered is against a dynamic oracle that knows the optimal arm at any trial t which is in contrast to the [12] EXP3 algorithm where the regret is against the single best arm (that one arm if played all the time will give the highest reward).

They calculate the lower bound to be $\mathcal{O}(T^{2/3})$ for this case. Knowing the variation budget in advance they work out the upper bound on regret to match the lower bound for this simple algorithm. The proof of REXP3 is simple and follows that of EXP3. The regret is divided into two components, the first component is the loss of using the single best arm against the dynamic oracle and the second component is the regret of the single best arm. The second component is bounded using the result from [12]. The first component is bounded using the following argument. X_t^a is the actual reward from action a at trial t and $\mathcal{T}_j$ is the indexes of trails in batch j.

$$\max_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}_j} \mu_t^a \leq E\Big[\max_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}_j} X_t^a\Big]$$

Choosing a single action with reward random variables has a larger sum than the single action with largest expected reward. Hence we can say:

$$\sum_{t \in \mathcal{T}_j} \mu_t^* - E\Big[\max_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}_j} X_t^a\Big] \leq \Delta_T \max_{\mathcal{T}_j} \Big\{\mu_t^* - \mu^{a_o}\Big\} \leq 2V_j\Delta_T \tag{6}$$

Where V_j is the variation budget for batch j and $\mu^{a_o} = \max_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}_j} \mu_t^a$ is the single best arm. In other words since the evolution of means is budgeted, means can not move much and also since the single best arm was optimal at least once, the most it could move in a batch is $\Delta_T V_j$

2.6.5 Multi-armed Bandit, Dynamic Environments

This paper consider the abruptly changing environments. They test multiple algorithms with UCB1 as a subroutine or adaptation of UCB1. γ-restart is a discounted UCB1 algorithm just like in Decay-UCB in [17]. They also try a Meta-algorithm that uses Page-Hinkly statistic to detect change. The Meta-algorithm builds two UCB1 algorithms. One that believe that the detector was true (new bandit) and the other assumes that detector was false (old bandit). The Meta-algorithm is also a UCB1 algorithm that chooses among the two bandits, the new and the old one. After some fixed time step the Meta-UCB drop the UCB1 algorithm with lower reward. Also a third type is the where they calculate a posterior probability whether a change occured or not. All these algorithms are verfied empirically and no theoratical justification is provided.

2.6.6 Piecewise-stationary Bandit Problems with Side Observations

In this paper they consider again abrupt changes at arbitrary intervals. Other wise the distributions of rewards are stationary. The number of change points are allowed to grow linearly in time $k(T)$. However in this study the environment also presents with some side information that is selected by the algorithm. More specifically the aglorithm can query observations on a set of arms. The total queries are limited by the horizon T. They develop a Meta-algorithm that runs an algorithm as a sub-routine with a given regret guarantee, passed by UCB1 and Robins and Lai. Further this meta-algorithm employs a change detection algorithm that asses the shift in the mean rewards over predetermined intervals. When ever a change is detected the sub-routine algorithm is reset. Further more the detected changes are ignored if they are smaller than some threshold. With this setting they proved a regret guarantee of $\mathcal{O}(|\mathcal{A}|k(T)\log(T))$. The proof follows by combining regret from various types of sources e.g. regret due to false positives or delay in detection of the actual change or regret of the sub-routine.

2.6.7 Adapting to the Shifting Intent of Search Queries

This paper deals with Intent shift in search queries for a web browser. They devise a Meta-algorithm that uses Bandit algorithm as a sub-routine and a classifier that detects the shift in the intent and resets the bandit sub-routine. The approach is similar to 2.6.6 except that the meta-algorithm can not query a chosen set of

arms (they assume that from time-to-time a bandit algorithm receives information about how users would have responded to search results that are never actually displayed). The classifier receives side information that helps it predict the event of intent shift. A typical search engine receives many signals that can be used to predict events, such as bursts in query reformulation, average age of retrieved document, etc. The regret is proved to be $O(k + d_F)(\frac{n}{\Delta} \log T)$ where k is the number of events, d_F is a certain measure of the complexity of the concept class F used by the classifier, n is the number of possible search results, Δ is the "minimum sub optimality" ($\Delta = \min_t \min_{i \neq i^{*t}} p_t(i_t^*) - p_t(i)$) of any search result (defined formally in Section 2), and T is the total number of impressions (observations). They assume that for any event the probability of user clicking the suggestion will change dramatically. Like other studies they also measure regret against non-stationary oracle.

This algorithm is sophisticated than other algorithms. It proceeds in phases. Imagine the bandit is reset at the start of the phase and this is an odd number of new phase, the algorithm will run bandit for L trials for new phase $phOdd_i$, after which it will enter a second phase $phEven_j$ which will continue indefinitely until the classifier predicts an event. The purpose of the phase $phOdd_i$ is to judge whether the last prediction of the classifier was correct or not. The algorithm has stored the bandit results from the phase $phOdd_{i-1}$ which it compares to the result $phOdd_{i-1}$ to affirm if the event of intent shift did occurred. For meeting the proved regret bounds, they assume that used Classifier and Bandit algorithm satisfy certain properties.

2.7 Evaluation of bandits on datasets

Bandit algorithms are hard to evaluate on a real life dataset because of their sequential and interactive nature. This is a general problem in evaluation of any reinforcement learning algorithm. [6] came up with a 'rejection algorithm' for evaluation of bandit algorithms described here as given by [6],

Algorithm 1 Policy Evaluator

Inputs: $T > 0$; policy π; stream of events
$h_0 \leftarrow \varnothing$
$R_0 \leftarrow 0$
for $t = 1, 2, 3, ..., T$ **do**
    repeat
        Get next event $(x_1, ..., x_K, a, r_a)$
    until $\pi(h_{t-1}, (x_1, ..., x_K)) = a$
    $h_t \leftarrow$ CONCATENATE$(h_{t-1}, (x_1, ..., x_K, a, r_a)$
    $R_t \leftarrow R_{t-1} + r_a$
end for
return R_T/T

At Yahoo! a dataset of about 50 million user interactions was collected with chosen articles over random users. The randomization was uniform to keep the policy evaluator unbiased. The policy evaluator chooses the events that the policy agrees with from the real data and the policy is evaluated against this matched data. We will use this same dataset and policy evaluator given in algorithm 1 for evaluating our approach for the recommendation system.

3 Problem Formulation

We formally describe the problem here and use the notation of [6]. A bandit algorithm $\mathcal{B}$ proceeds in trails t and is faced with arms $a \in \mathcal{A}$. Without the loss of generality we assume that $\mathcal{A}$ is fixed. Context $\mathbf{x}_{t,a}$ observed by the algorithm summarizes information for both the user and the action a (recommendation) at time t. Parts of context vector corresponding to user and the recommendation are obviously independent of each other. Also $\|\mathbf{x}_{t,a}\| \leq 1$. Based on the history $\mathcal{B}$ chooses action a_t and receives payoff r_{t,a_t}. The optimal action is the one with highest expected reward, call it a_t^*. At time t the total reward gain is $\sum_{i=1}^{t} r_{i,a_i}$ and the optimal reward is given by $\mathbf{E}[\sum_{i=1}^{t} r_{i,a_i}]$. The goal is to find an algorithm that maximizes the expected reward for T trials or in other terms minimize the regret $\mathbf{R}$.

$$\mathbf{R}_{\mathcal{B}}(\mathbf{T}) = \mathbf{E}[\sum_{t=1}^{T} r_{t,a_t^*}] - \mathbf{E}[\sum_{t=1}^{T} r_{t,a_t}] \tag{7}$$

This is dynamic definition of regret in which the optimal action can change with time. In such a case stationary policy (algorithm) will not be optimal. We assume the expected payoff is given by the following non-stationary Gaussian distribution.

$$\begin{aligned} & r_{t,a_t} \sim \beta_{a_t}^t \mathcal{N}(\mathbf{x}_t^\top \boldsymbol{\theta}_{a_t}, \sigma^2) \\ & \beta_a \in [0,1] \quad and \quad \|\boldsymbol{\theta}_a\| \leq 1 \end{aligned} \tag{8}$$

Equ. 8 characterizes the exponentially decaying reward structure. $\boldsymbol{\theta}$ as before parametrizes the linear relation between the context vector $\mathbf{x}_{t,a}$ and the payoff $r_{t,a}$. β_a is the decay factor that makes the article a an unpopular recommendation with time. Notice that the decaying factor affects both the expected reward ($\mathbf{x}^\top \boldsymbol{\theta}$) and the variance σ of the signal, hence keeping the rewards from being overwhelmed by the noise. Each action a has its own decaying rate β_a which means the optimal action can decay faster and will be taken over by some other actions. In this problem there is no bound on the change in the expectation of

rewards and hence the algorithms mentioned in section 2.6 dealing with non-stationary environment and depending on the variation budget will not be optimal. An optimal algorithm will parametrize both the functional dependence of reward on the context and the decaying trend. This guides us to a UCB type algorithm which uses a non-stationary classifier to learn the parameters $\boldsymbol{\theta}$ and β in approach 1.

4 Method

In this section we state our proposed approaches and explain their motivation. We will proceed in a way that more and more specific assumptions are made about the environment. As mentioned in the introduction, from Decay LinUCB, to Restart LinUCB, to Temporal LinUCB, we will see improving performance and improving regret property, which can be verified with experimental results in Section 6 and theoretical analysis in Section 5.

4.1 Method 1: Decay LinUCB

The main problem with any stationary algorithm is that it gives equal weight to its history. In a non-stationary environment, if no specific assumption about how the environment will change, a simple idea is to use a weighing function to have less effect of the past on to the current decisions. This idea is widely used and hence fairly general and for its application the algorithm has to support instance weighting in its learning algorithm. Luckily since we are using LinUCB as our base model and it uses ridge regression, implementation of weighing instances is straight forward. Decay LinUCB is a simple variation on the LinUCB. The only difference is observed by assigning exponentially decaying weights to the past examples in the ridge regression. Similarly same kind of weights are also applied in the calculation of the confidence width in the UCB. It is not hard to see that it only diverges from LinUCB in its updating the design matrix $\mathbf{A}_{t,a}$ and the reward weighted context vector $\mathbf{b}_{t,a}$. Following the notation from algorithm 3 we state the following equations that govern the update of appropriate matrices. In this case γ represents the decay parameter. Notice that this is also exponential decay since an example will be weighed exponentially proportional to how far it is in the past from the current time.

$$\begin{aligned}
\mathbf{D}_{t+1,a} &= \gamma \mathbf{D}_{t,a} + \mathbf{x}_t \mathbf{x}_t^\top \\
\mathbf{A}_{t+1,a} &= \mathbf{D}_{t+1,a} + \mathbf{I} \\
\mathbf{b}_{t+1,a} &= \gamma \mathbf{b}_{t,a} + \mathbf{x}_t r_{t,a}
\end{aligned} \tag{9}$$

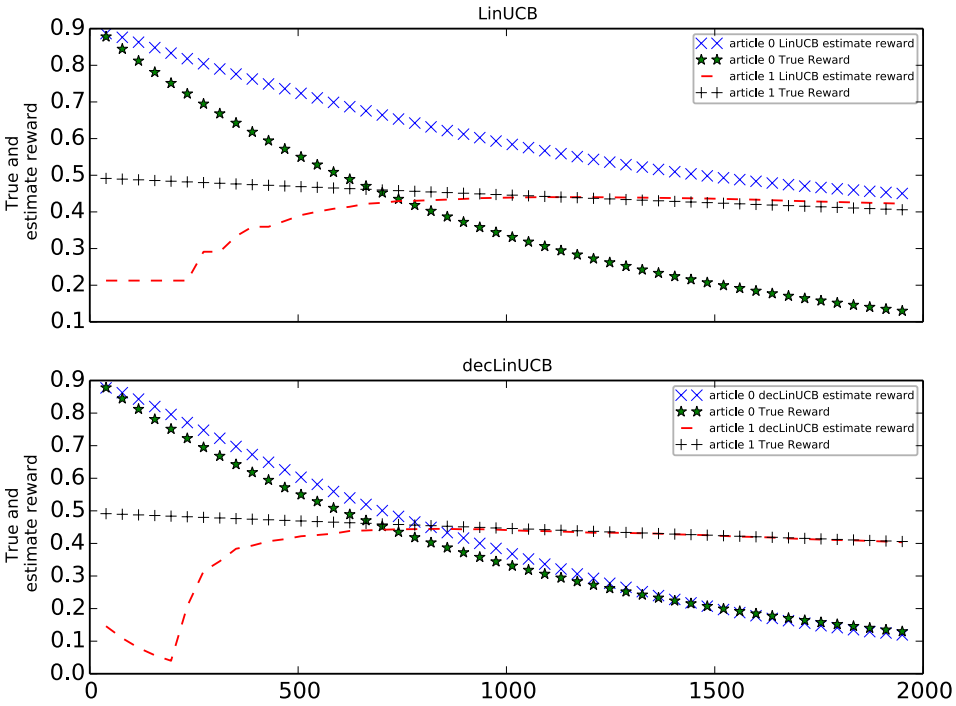

Figure 1: Decay LinUCB tracks the true rewards better than LinUCB, in fact it tracks very well.

This style of algorithm is very general in nature since it does not assume anything specific about the type of the evolution. Still for it to perform well the environment has to observe a certain rate of change beyond which this most general algorithm will be practically useless. Conceptually decay-LinUCB gradually forgets the past and keeps re-estimating the expected reward based of the recent events. Due to its constant re-estimation the algorithm's confidence width does not decrease monotonically, in fact it retires to a never ending and certain level of minimal exploration, which it assumes is necessary since it is always anticipating change in the environment. Practically decay-LinUCB makes a lot of sense in the general case however due to a constant minimal exploration it suffers from linear regret. In case of a generic non-stationary environment where little can be inferred about its surroundings, decay-LinUCB is probably our best bet (in practical sense) since it does not make any assumptions.

In fig. 1 we simulate the environment with just 2 articles and a single user or context. Essentially this is a context free setting however it illustrates the aspects of the exponentially decaying environment and also just one user allows for easier analysis with out compromising any completeness of the full picture. We plot in the figure the true rewards for two articles and the algorithms tracking these rewards. Note that the true

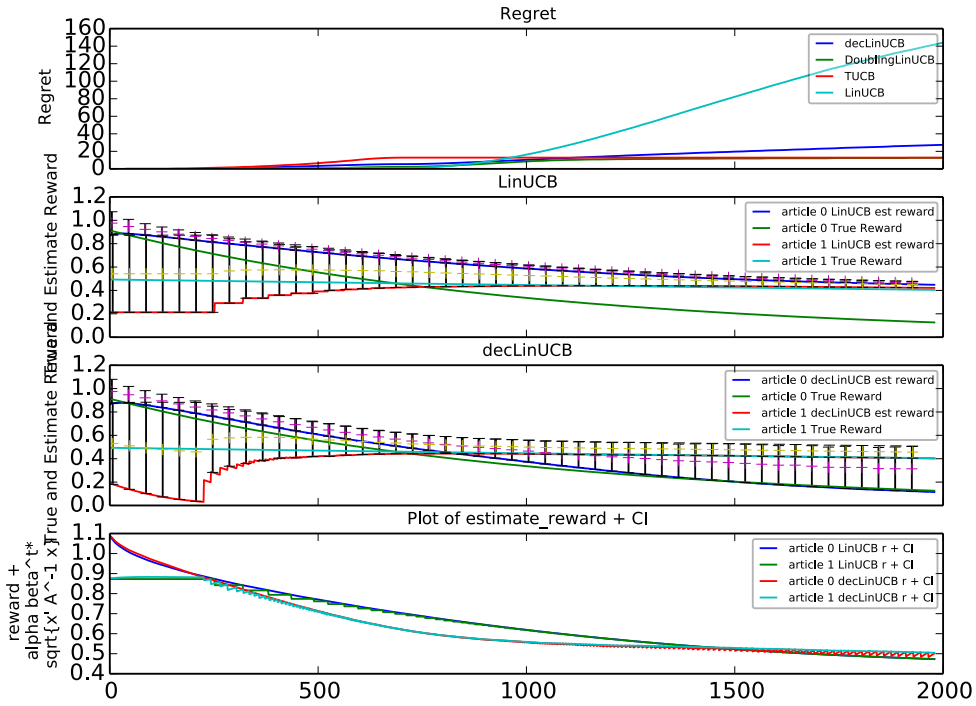

Figure 2: For two articles Decay-LinUCB switches very close to the optimal switch and avoids linear regret.

rewards switch in the middle of and hence show case an adverse effect of this non-stationary setting. This subplot highlights the weakness of the LinUCB which over estimates the rewards and negatively affects its performance. In the later subplot it is apparent that decay-LinUCB is able to track these rewards much better. This was the initial motivation for the general algorithm however we did not imagine the situation if more articles were present in the pool. Even in that case the estimation is close to the real reward however since Decay-LinUCB does not realize it keeps exploring all the arms constantly and hence incurs linear regret. Fig. 2 in addition shows the interaction of the algorithms decision criterion by plotting the sum of the estimated reward and the confidence width. The first subplot shows the regret, second and third are the same as before except for the added bars that show the magnitude of the confidence width. The comparison of the bars show which action or article was actually picked. The last subplot again shows the bars tops as curves for the two articles and the algorithms. Here notice a "ladder" behavior where the suboptimal action's curve looks like a ladder. This makes sure that the estimated superior action is chosen much more often than the other arms. This behavior is due the fact that for LinUCB the confidence bound decays only

when the article is chosen. When a superior option is chosen its width decays much less than the inferior options/actions. This is due to the fact that statistically the actions that have relatively smaller expected rewards will be selected less and the range that the confidence lies will offer larger decay on choice than later when the action has been chosen many times. This is due to the quadratic confidence width curves for which the first derivative gets smaller the more they are chosen.

4.2 Method 2: Restart LinUCB

Decay LinUCB as described does not obey any assumption but also on the other side it affords linear regret. From here we move on to a different idea where we specify a relatively strict assumption. We concoct an assumption that the variation in the environment is decreasing with time. If this is indeed the case then we can use this information to develop a better algorithm. Since our setting of exponential decay falls under the allowed set of non-stationary environments, this new algorithm will work on our setting as well.

We describe the simple idea of resetting LinUCB after predetermined time intervals. This modest approach has been proposed before in context of non-stationary adversarial algorithms [14] and similarly in other scenarios with diverse range of settings. The ease of implementation makes this idea very attractive. We further introduce the doubling trick where the reset time interval length doubles every time. This is motivated from a useful side effect of the decaying environment. This doubling of interval length has been tried with a profound notion of limiting exploration to achieve sub-linear regret without specifically estimating any decay parameters. This approach derives its motivation from the fact in an exponentially decaying environment, the change in the expected reward later will be much lower than at the start of the run. This intuition is further supported by the theoretical guarantee in section 5. Restarting the algorithm makes a very particular statement that the history is worthless and it will be beneficial to discard it. This is a very strong statement and also not entirely correct; indeed the non-stationary setting renders the past less useful however the past still carries some useful information and discarding it completely can be detrimental. However based on the proof and simulation results, we argue resetting with doubling interval length is more good than bad. Considering the downside of the algorithm of discarding the complete history another idea is to use warm restart instead of complete reset. In this warm restart the history is given a large decay instead a reset. This algorithm is a fusion of the decay and restart which could be the focus of future research, here however we focus on the fundamental results since warm restarts are more of a practical idea that will give a better constant but may not help with the over all regret bound. On the upside restart LinUCB is relatively a generic algorithm and can cater to multiple scenarios of evolutionary settings e.g any setting where the

variation diminishes with time and the actions converge on static values, the restart-LinUCB type algorithm can perform well.

Algorithm 2 Restart LinUCB

1: Initialize LinUCB; $\mathbf{A}_a = \mathbf{I}_d, \mathbf{b}_a = \mathbf{0}_{d\times 1}$ for $a \in \mathcal{A}$
2: **for** $j = 1, 2, 3, ..., \log T$ **do**
3: $\quad t_j = 2^j$, Reset LinUCB
4: $\quad$ **for** $t = 2^{t_j-1}, 2^{t_j-1}+1, ..., 2^{t_j}-1$ **do**
5: $\quad\quad$ Play LinUCB
6: $\quad$ **end for**
7: **end for**

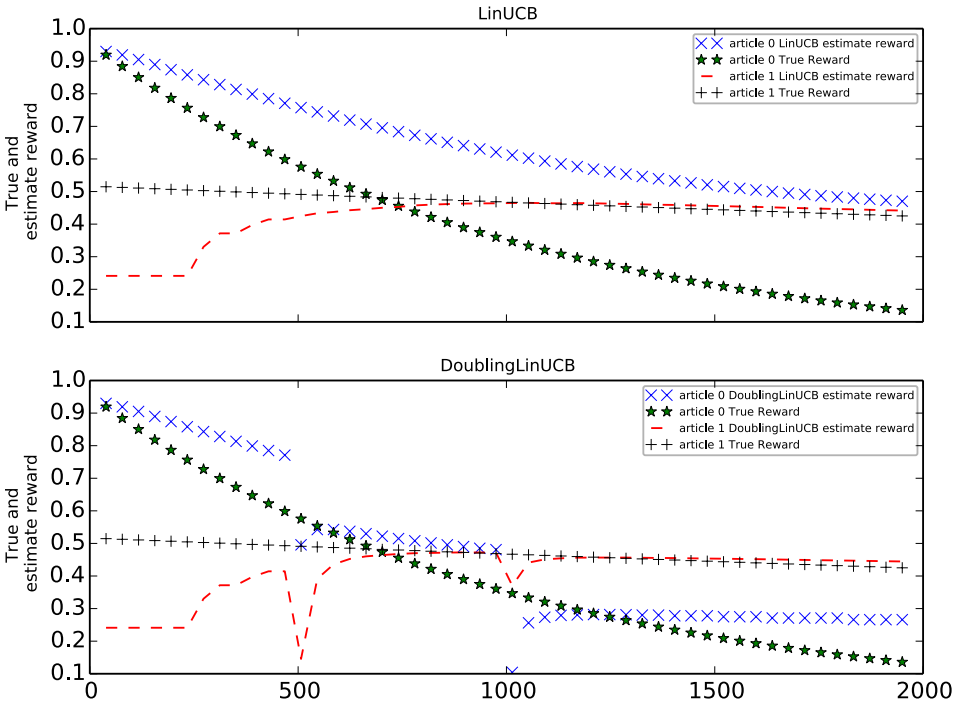

Figure 3: Restart LinUCB due to restart does better.

In fig. 3 the same experiment is repeated as the previous one. Here we observe that the LinUCB resets after doubling intervals and kills the over estimation by an effective amount that can help to make it perform better. Similarly in the fig. 4 the restarting exploration lets the algorithm converge on a better alternative. The nature of the confidence width curves is also more apparent in the 4h subplot where at start there is more room for decay than later on.

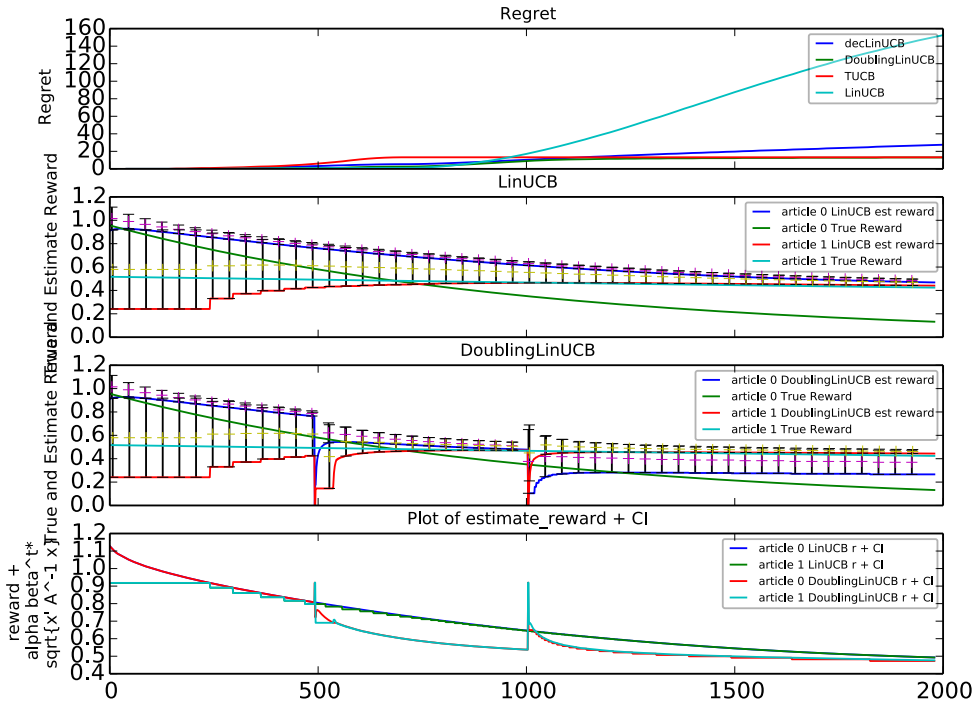

Figure 4: Sudden exploration when LinUCB resets.

4.3 Method 3: Temporal LinUCB (TUCB)

While using a strict assumption in restart LinUCB limited the number of feasible settings, it did give us a better performing algorithm than the general decay LinUCB. We keep heading in this direction to further strict the assumptions and in fact consider the strongest possible that one can make. Here we assume that the algorithm knows the environment exactly. TUCB is a UCB style algorithm that is in principal an index based policy introduced by [13]. In every trial, the algorithm calculates an index for each arm. This index is the sum of the predicted expected reward and a confidence bound on the error of the prediction. These type of algorithms are close to optimal and are well suited for situations where the model of the environment is known. Since we characterize the environment in Equ. 8 as a normally distributed decaying linear expected reward, a parametric estimation, we utilize the UCB framework to design the algorithm. To estimate the expected reward we develop MLE for our parameters. For estimation of $\boldsymbol{\theta}$ we adhere to the approach in [6] and use ridge regression which does not require the design matrix to be invertible. Very similar to LinUCB another algorithm uses SVD to map the context to the rewards, known as LinREL in [9] which also needs

the design matrix to be sufficiently regular. For practical purposes LinUCB [6] is shown to be sufficient and hence is the motivation for TUCB.

Following the notation of LinUCB, let $T_{t,a}$ be a vector of time index when ever action a was played till trail t and let $\mathbf{G}_{t,a} = diag(T_{t,a})$. Further more let $\mathbf{D}_{t,a} = [\mathbf{x}_{\tau,a_\tau}]_{\tau \in T_{t,a}}$ and $\mathbf{y}_{t,a} = [r_{\tau,a_\tau}]_{\tau \in T_{t,a}}$. We define an estimator $\mathcal{E}$ which is a maximum likelihood estimator for the parameter $\boldsymbol{\theta}$ and β. Let $\mathbf{A}_{t,a} = (\mathbf{D}_{t,a}^\top \mathbf{D}_{t,a} + \mathbf{I}_d)$ where $\mathbf{I}_{d\times d}$ is the identity matrix. $\mathcal{E}$ uses the following equations to estimate the parameters.

$$\begin{aligned} \widehat{\beta}_{t,a} &= \arg\min_{\beta} \ \ \|(\mathbf{I}_t - \mathbf{D}_{t,a}\mathbf{A}_{t,a}^{-1}\mathbf{D}_{t,a}^\top) det(\mathbf{G}_{t,a})^{\frac{2}{t}} \mathbf{G}_{t,a}^{-1}\mathbf{y}_{t,a}\|^t \\ \widehat{\boldsymbol{\theta}}_{t,a} &= \mathbf{A}_{t,a}^{-1}\mathbf{D}_{t,a}^\top \mathbf{G}_{t,a}^{-1}\mathbf{y}_{t,a} \end{aligned} \tag{10}$$

This objective function while solving for the parameter β, solves for the for $\boldsymbol{\theta}$ as a starting point. In other terms, it guesses a β, removes the decay and finds the best $\boldsymbol{\theta}$ using ridge regression to calculate the error that guides the search for the optimal β. After estimating the β, linear parameter $\boldsymbol{\theta}$ can be determined by straight forward linear regression. Notice the matrix $\mathbf{G}$ that takes the non-stationarity out of the equation and let the linear parameter maps on the residuals. Here we give the complete description of the TUCB algorithm. Due to non-linear decay, the objective function for estimating β does not have a close form and so the optimization problem can be solved using Newtons method. Also the objective function of β is non-convex but enjoys a global minimum.

Algorithm 3 TUCB

1: $\alpha \in \mathbb{R}_+$
2: **for** $t = 1, 2, 3, ..., T$ **do** Observe features for all arms $a \in \mathcal{A}$; $\mathbf{x}_{t,a} \in \mathbb{R}^d$
3: $\quad p_{t,a} = \widehat{\beta}_{t,a}^t \mathbf{x}_{t,a}^\top \widehat{\boldsymbol{\theta}}_{t,a} + \alpha\rho_{t,a}\sqrt{\mathbf{x}^\top \mathbf{A}^{-1}\mathbf{x}}$
4: $\quad a_t \leftarrow \arg\max_{a\epsilon\mathcal{A}} p_{t,a}$ (break ties arbitrarily)
5: $\quad$ Play arm a_t and incur reward r_{t,a_t}
6: $\quad$ Observe reward r_t
7: $\quad$ Update classifier $\mathcal{C}$ with new observation $(\mathbf{x}_{t,a_t}, r_t, t)$ and estimate $\widehat{\beta}_{t,a}, \widehat{\boldsymbol{\theta}}_{t,a}$
8: **end for**

As can be recognized the algorithm 3 looks similar to the LinUCB style algorithms. These algorithms are simple in bare structure however the appropriate function and parameters are inferred from the proof. As mentioned before the index $p_{t,a}$ has two parts, an estimated reward and confidence width. By design we choose equation 10 for estimation and hence the first part of the index is set. For the second part we borrow the confidence width from LinUCB. The confidence width depends on the bound of the estimation accuracy. Curiously enough it has nothing to do with actual rewards, in fact it is solely dependent on the contexts observed. This comes from the UCB style approach where general statistical techniques like

Hoeffding inequality ensure that the estimates due to random noise do not go further than the bound with probability δ which is directly proportional to the number of observations and the times that action was played. The confidence width can also be interpreted as the variance around the expected payoff. In LinUCB this width is given by the factor $\sqrt{\mathbf{x}^\top \mathbf{A}^{-1}\mathbf{x}}$. In our case following the logic of the proof we infer that the variance will have one more factor ρ which is function of time and is owing to the estimation error bound on the decay factor β. In our experimental results we use the function $\rho_{t,a} = 1$ which we further illustrate in the proof.

As previously mentioned TUCB is implementation heavy and relies on the entire history to calculate its results. So from a practical concern we only update γ in batches specifically in later part of the experiment where each new data point gives us less improvement.

Below we show the derivation of the MLE. Starting of the environment definition.

$$r_t = \beta^{*t_i}(\mathbf{x}_{t,a}^\top \theta^* + \epsilon_{\mathcal{N}(0,\sigma^2)}) \tag{11}$$

We find β and θ using least squares and numerical optimization. Let $\mathbf{G} = diag(\beta^{t_1}, \beta^{t_2}, ..., \beta^{t_t})$. Let $\mathbf{y}_t = [r_i]_{i\in 1,..,t}$ be the vector of rewards seen till time t. Then the MLE for the parameter $\boldsymbol{\theta}$ can be worked out as such.

$$\theta_{\beta,t} = \mathbf{A}_{i,a}^{-1}\mathbf{D}_{i,a}^\top \mathbf{G}_{i,a}^{-1}\mathbf{y}_{i,a}$$

Given β we can find a θ as above using least squares method. With the above θ we will like to find a β for the eq.11.

$$y_t - \beta^{*t_i}\mathbf{x}_{t,a}^\top \theta^* \sim \mathcal{N}(0, \beta^{*2t_i}\sigma^2) \tag{12}$$

We write the likelihood function for the entire data.

$$\begin{aligned} \mathcal{L}(\beta|\mathbf{y}_t, \mathbf{D}_t, \mathbf{G}_t) &= \frac{1}{2\pi\sigma^{2^{t/2}}}\beta^{-\sum_{i=1}^t t_i} \exp \sum_{i=0}^{t} \frac{-(y_i - \beta^{t_i}\theta_{\beta,t}^\top \mathbf{x}_t)^2}{2\sigma^2\beta^{2t_i}} \\ \log \mathcal{L} &= -\frac{t}{2}\log 2\pi\sigma^2 - \log\beta \sum_{i=0}^{t} t_i - \sum_{i=0}^{t} \frac{(y_i - \beta^{t_i}\theta_{\beta,t}^\top \mathbf{x}_t)^2}{2\sigma^2\beta^{2t_i}} \end{aligned} \tag{13}$$

To find the β we should maximize the likelihood however due to σ we can not drive an objective function

in terms of β. We find the σ that maximizes likelihood and replace it in eq.13.

$$\begin{aligned}\frac{\partial log\mathcal{L}}{\partial\sigma^2} &= -\frac{t}{2\sigma^2} + \sum_{i=0}^{t}\frac{(y_i - \beta^{t_i}\theta_{\beta,t}^{\top}\mathbf{x}_t)^2}{2\sigma^4\beta^{2t_i}} = 0 \\ \widehat{\sigma^2} &= \frac{1}{t}\sum_{i=0}^{t}\frac{(y_i - \beta^{t_i}\theta_{\beta,t}^{\top}\mathbf{x}_t)^2}{\beta^{2t_i}}\end{aligned} \tag{14}$$

Placing σ back in log likelihood gives.

$$\begin{aligned}\log\mathcal{L} &= -\frac{t}{2}\log\frac{2\pi}{t}\sum_{i=0}^{t}\frac{(y_i - \beta^{t_i}\theta_{\beta,t}^{\top}\mathbf{x}_t)^2}{\beta^{2t_i}} - \sum_{i=0}^{t}t_i\log\beta - \frac{t}{2} \\ \log\mathcal{L} &\propto -\frac{t}{2}\log\sum_{i=0}^{t}(\frac{y_i}{\beta^{t_i}} - \theta_{\beta,t}^{\top}\mathbf{x}_t)^2 - \sum_{i=0}^{t}t_i\log\beta\end{aligned} \tag{15}$$

The objective function looks like below.

$$\widehat{\beta} = \arg\min_{\beta} \quad \frac{t}{2}\log\sum_{i=0}^{t}(\frac{y_i}{\beta^{t_i}} - \theta_{\beta,t}^{\top}\mathbf{x}_t)^2 + \log(\beta)\sum_{i=0}^{t}t_i \tag{16}$$

This objective function can be written in matrix notation.

$$\widehat{\beta} = \arg\min_{\beta} \quad \frac{t}{2}\log\|(\mathbf{I}_t - \mathbf{D}\mathbf{A}^{-1}\mathbf{D}^{\top})\mathbf{G}^{-1}\mathbf{y}_t\|^2 + \log det(\mathbf{G})$$

Simplifying further we can achieve the following form.

$$\widehat{\beta} = \arg\min_{\beta} \quad \|\mathbf{C}^{\top}det(\mathbf{G})^{\frac{2}{t}}\mathbf{G}^{-1}\mathbf{y}_t\|^t \tag{17}$$

In fig. 5 the objective function is drawn for different true β and different variance of noise for data size 100. Since our goal is to minimize the function, we use the Newton-CG method to find the minimum of this function. In fig. 5 we see that this objective function has global minimum and it lies close to the ground truth for different levels of noise. There is a drop close to the ground truth. The drop is less sharp for larger variance noise. In fig. 6 the data size is 1000 instances. In larger data the drop is narrower and deeper.

In fig. 7 we plot the convergence of estimated β as we add more data for estimation. The final data set size is 500 examples which are labeled on x-axis with increments of 10 examples. Y-axis is the estimated β or γ i.e. the decay factor. The horizontal line shows the true β also given by the sub-plot title. Each sub-plot shows convergence of estimates for different noise levels.

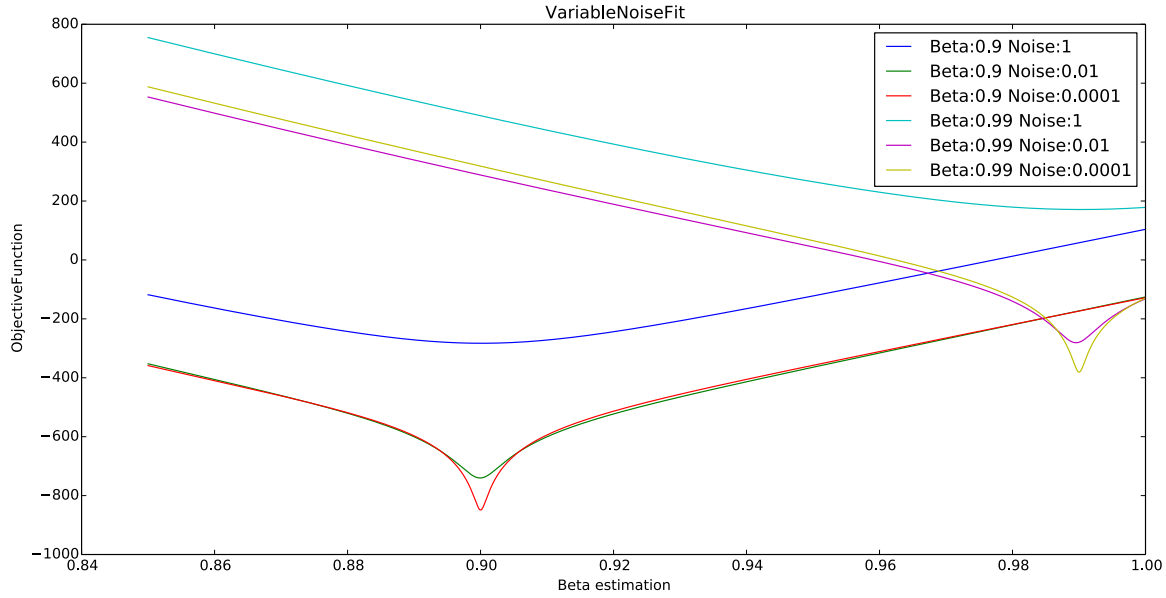

Figure 5: Plot of objective function of equ. 16. Here sample size was 100. For different noise levels it is robust. **Note:** Noise=1, beta=.99 is increasing at 1.

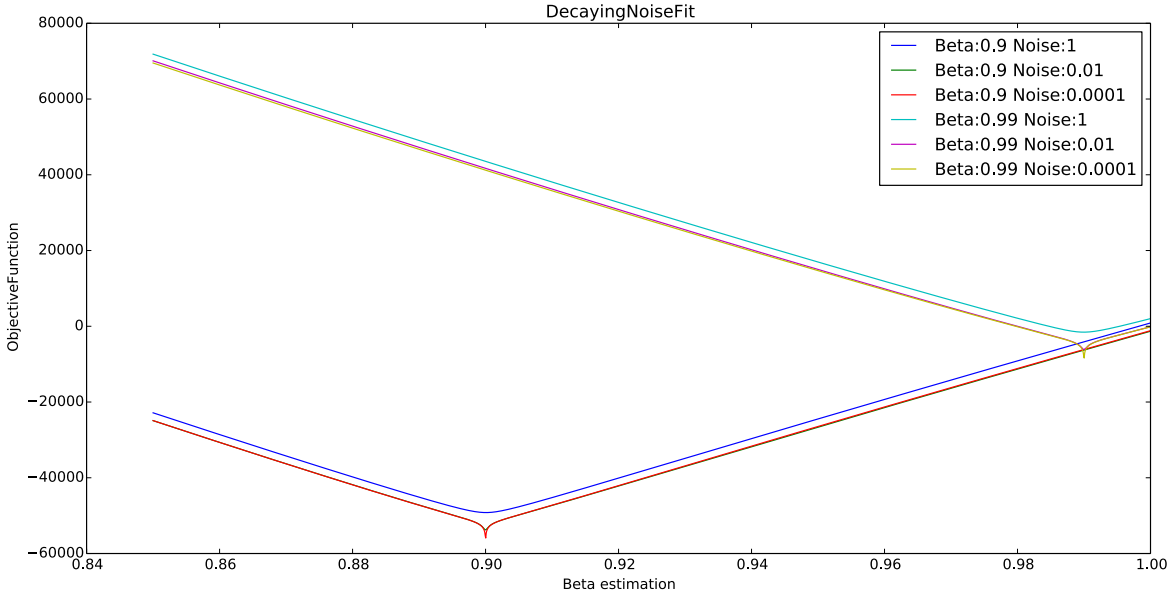

Figure 6: Plot of objective function of equ. 16. Here sample size was 1000.

For different values of true β and noise variance we observe that the estimate converges after seeing 100 or less examples. The convergence rates are larger for smaller noise. The effect of true β is not clear, perhaps

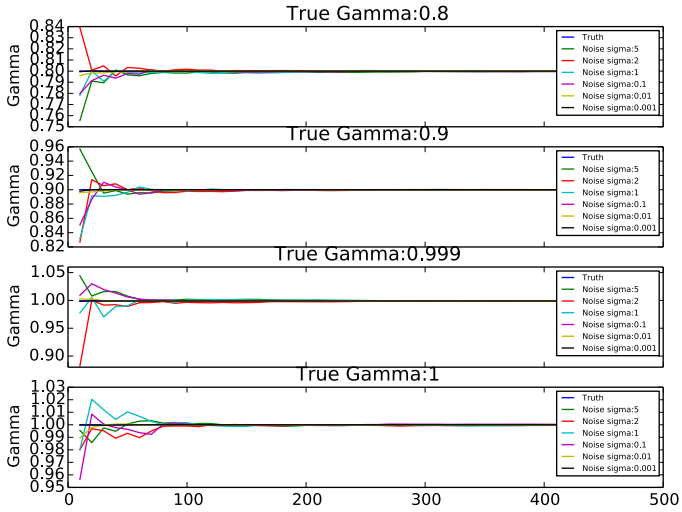

Figure 7: This figure compares convergence of numerical method to approximate β for different values of true β and noise levels. X-axis shows the data used for the approximation. **Conclusion:** We see that larger noise slows convergence to true β. Also noise has more effect if true β is closer to 1.

it is slightly slower when it is closer to 1.

Fig. 8 shows the convergence of θ and plot the $l1$ norm between the estimate and true θ i.e $|\theta - \hat{\theta}|$. Here Y-axis is incorrectly labeled as "theta MSE". We see that this estimation is unstable for large noise variance 5,2 and 1. For lower noise levels it is closer to 0. Also not clear in this figure is the convergence of lower noise levels as they are invisible at this scale so we plotted those separately in fig. 9 where we see that for lower variance of 0.01 and 0.001 the estimator is better behaved than for variance .1 which is more erratic.

Again same simulation on the toy example is repeated for the TUCB which shows good estimation obviously. One behavior that is not UCB style is that the decision curve for the inferior arm departs permanently from the superior arm permanently which is not UCB style since it shunts the exploration for ever. This is due to the incorrect confidence bound formulation used in our algorithm. Due to limitation of finding new statistical properties about such estimation we could not complete the proof and hence derive the true version of the bound.

5 Theoretical guarantees

In the following proofs, in order to apply the Azuma/Hoeffding inequality for deriving upper bound of regret, we will first assume the statical independence among the samples and use a master algorithm, which

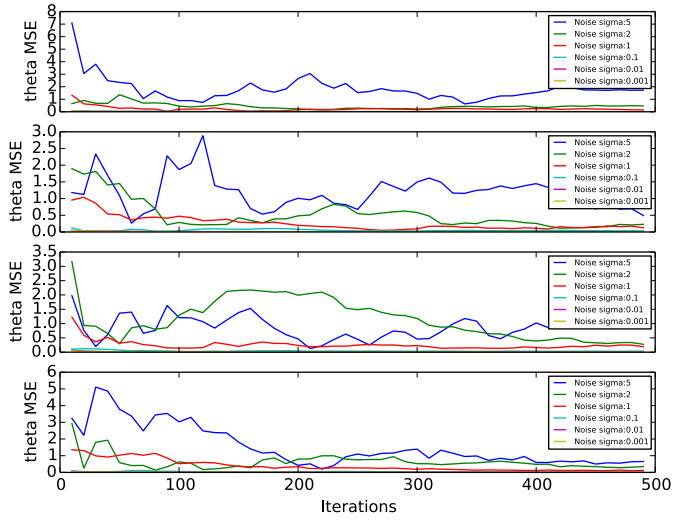

Figure 8: This figure shows convergence of numerical method to approximate θ for different values of true β and noise levels. X-axis shows the data used for the estimation. From top sub-plot the true β is .8, .9, .999 and 1.

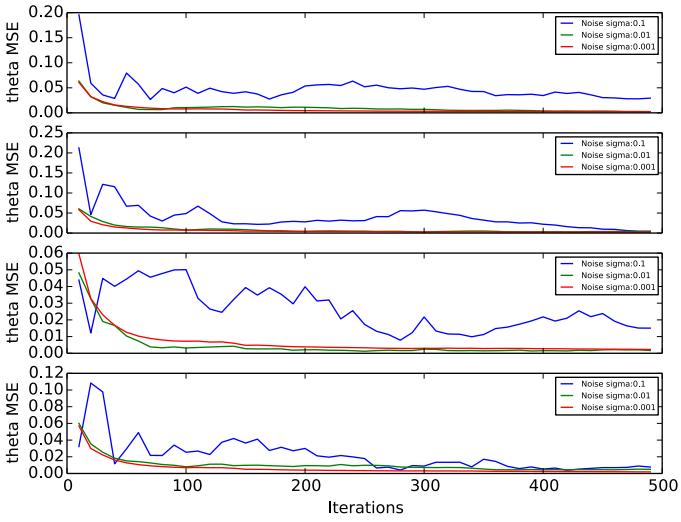

Figure 9: Convergence of θ for smaller noise levels. From top sub-plot the true β is .8, .9, .999 and 1.

comes from LinRel/SupLinUCB decomposition in [9] and the BaseLinUCB/SupLinUCB [10], to ensure this independence.

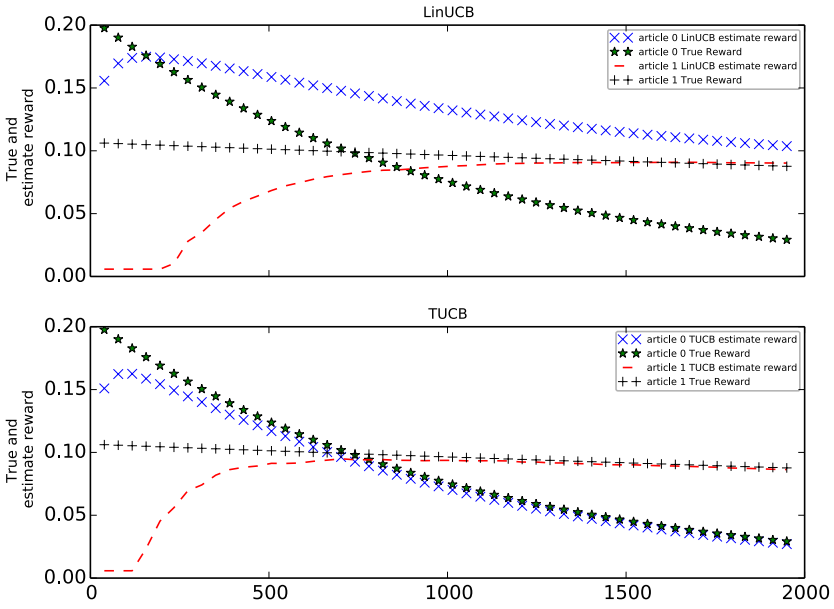

Figure 10: Temporal LinUCB tracks the true reward consciously and converges.

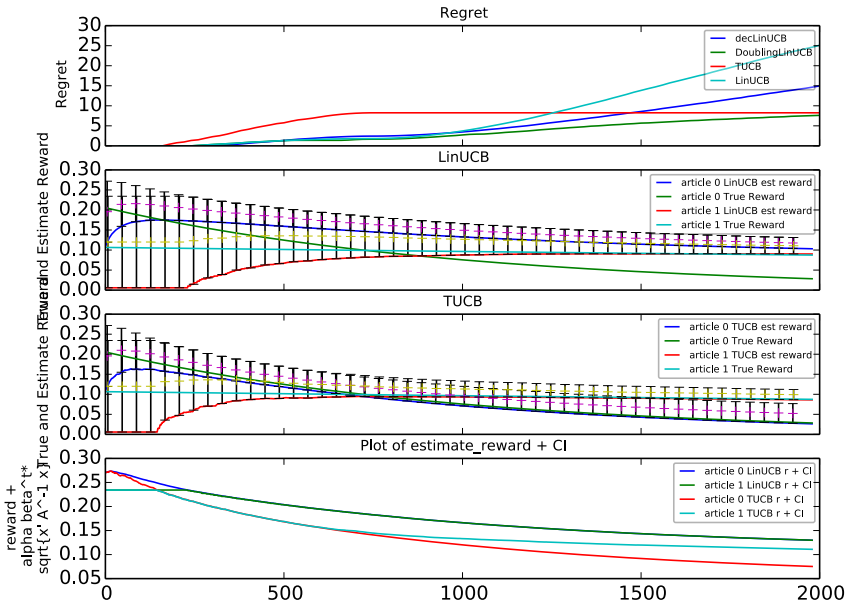

Figure 11: Correctly gets the switch and move on to the better action.

5.1 LinUCB regret

We start off by saying that the regret for LinUCB is linear. We prove that the switch point of the article true rewards and the LinUCB switch (when it realizes that its superior action has become inferior) are far apart and in fact the distance between them is proportional to the actual switch point. Since the switch point could be at $T/2$ it will take LinUCB linear time to recognize the switch. According to our assumption about the environment, $E(r_{a,t}|\mathbf{x}_t) = \beta_a^t \mathbf{x}_t^\top \boldsymbol{\theta}_a$. So for article 1, $E(r_{1,t}|\mathbf{x}_t) = \beta_1^t \mathbf{x}_t^\top \boldsymbol{\theta}_1$; for article 2, $E(r_{a,t}|\mathbf{x}_t) = \beta_2^t \mathbf{x}_t^\top \boldsymbol{\theta}_2$. Suppose $\mathbf{x}_t^\top \boldsymbol{\theta}_1 > \mathbf{x}_t^\top \boldsymbol{\theta}_2$ and $\beta_1 < \beta_2$, and denote the first switch point as t^*,

$$\beta_1^t \mathbf{x}_t^\top \boldsymbol{\theta}_1 \leq \beta_2^t \mathbf{x}_t^\top \boldsymbol{\theta}_2 \tag{18}$$

So we have, $t \geq log_{\frac{\beta_1}{\beta_2}} \frac{\mathbf{x}_{t*}^\top \boldsymbol{\theta}_2}{\mathbf{x}_{t*}^\top \boldsymbol{\theta}_1}$, which means when $t \geq t^* = log_{\frac{\beta_1}{\beta_2}} \frac{\mathbf{x}_{t*}^\top \boldsymbol{\theta}_2}{\mathbf{x}_{t*}^\top \boldsymbol{\theta}_1}$ the optimal article which should be chosen is article 2.

For LinUCB, we have $E(\hat{r}_{1,t}|\mathbf{x}_t) = \mathbf{x}_t^\top \hat{\boldsymbol{\theta}}_1$ and $E(\hat{r}_{a,t}|\mathbf{x}_t) = \mathbf{x}_t^\top \hat{\boldsymbol{\theta}}_2$, in which $\hat{\boldsymbol{\theta}}_1 = \mathbf{A}_{t_1}^{-1}\mathbf{b}_{t_1}$, $\hat{\boldsymbol{\theta}}_2 = \mathbf{A}_{t_2}^{-1}\mathbf{b}_{t_2}$. We denote the switch point for LinUCB as t_0, so $t_1 + t_2 = t_0$. If we want LinUCB to identify article 2 as the best article, we should have,

$$\mathbf{x}_{t_0}^\top \mathbf{A}_{t_1}^{-1} \mathbf{b}_{t_1} \leq \mathbf{x}_{t_0}^\top \mathbf{A}_{t_2}^{-1} \mathbf{b}_{t_2} \tag{19}$$

Putting $\mathbf{A}$ and $\mathbf{b}$ into the equation, we have,

If we assume that for i from 1 to t, $\mathbf{x}_i = \mathbf{c}$, the equation above can be simplified as,

$$\mathbf{c}^\top (\mathbf{I} + t_1 \mathbf{c}\mathbf{c}^\top)^{-1} \mathbf{c}^\top \boldsymbol{\theta}_1 \mathbf{c} \sum_{i_1} \beta_1^{i_1} \leq \mathbf{c}^\top (\mathbf{I} + t_2 \mathbf{c}\mathbf{c}^\top)^{-1} \mathbf{c}^\top \boldsymbol{\theta}_2 \mathbf{c} \sum_{i_2} \beta_2^{i_2} \tag{20}$$

According to Lemma 5.1, Eq 20 can be written as,

$$\mathbf{c}^\top (\mathbf{I} - \frac{t_1}{1 + t_1 \|\mathbf{c}\|^2} \mathbf{c}\mathbf{c}^\top) \mathbf{c}^\top \boldsymbol{\theta}_1 \mathbf{c} \sum_{i_1} \beta_1^{i_1} \leq \mathbf{c}^\top (\mathbf{I} - \frac{t_2}{1 + t_2 \|\mathbf{c}\|^2} \mathbf{c}\mathbf{c}^\top) \mathbf{c}^\top \boldsymbol{\theta}_2 \mathbf{c} \sum_{i_2} \beta_2^{i_2} \tag{21}$$

It can be further simplified as,

$$\mathbf{c}^\top \boldsymbol{\theta}_1 \|\mathbf{c}\|^2 \sum_{i_1} \beta_1^{i_1} - \frac{t_1}{1 + t_1 \|\mathbf{c}\|^2} \|\mathbf{c}\|^4 \mathbf{c}^\top \boldsymbol{\theta}_1 \sum_{i_1} \beta_1^{i_1} \leq \mathbf{c}^\top \boldsymbol{\theta}_2 \|\mathbf{c}\|^2 \sum_{i_2} \beta_2^{i_2} - \frac{t_2}{1 + t_2 \|\mathbf{c}\|^2} \|\mathbf{c}\|^4 \mathbf{c}^\top \boldsymbol{\theta}_2 \sum_{i_2} \beta_2^{i_2} \tag{22}$$

Then we have,

$$\mathbf{c}^\top \boldsymbol{\theta}_1 \sum_{i_1} \beta_1^{i_1} \frac{1}{1+t_1\|\mathbf{c}\|^2} \leq \mathbf{c}^\top \boldsymbol{\theta}_2 \sum_{i_2} \beta_2^{i_2} \frac{1}{1+t_2\|\mathbf{c}\|^2} \tag{23}$$

$$\frac{1+t_1\|\mathbf{c}\|^2}{1+t_2\|\mathbf{c}\|^2} \geq \frac{\mathbf{c}^\top \boldsymbol{\theta}_1 \sum_{i_1} \beta_1^{i_1}}{\mathbf{c}^\top \boldsymbol{\theta}_2 \sum_{i_2} \beta_2^{i_2}} \tag{24}$$

Analysis: When t_1 and t_2 satisfy the inequality in Eq 24, LinUCB will identify article 2 as the better article

Suppose $\|\mathbf{c}\|^2 = 1$ and suppose $\beta_2 = 1$ and $\mathbf{c}^\top \boldsymbol{\theta}_1 = m\mathbf{c}^\top \boldsymbol{\theta}_2$ in which $m > 1$.

Then $t^* = \log_{\frac{1}{\beta_1}} m$

For LinUCB, Eq 24 can be simplified as,

$$t_1 \geq (\frac{m}{t_2} + m) \sum_{i_1} \beta_1^{i_1} - 1 \tag{25}$$

Then,

$$t_0 = t_1 + t_2 \geq (\frac{m}{t_2} + m) \sum_{i_1} \beta_1^{i_1} + t_2 - 1 \tag{26}$$

$$t_0 > m \sum_{i_1} \beta_1^{i_1} \geq (\frac{1}{\ln \frac{1}{\beta_1}} + \log_{\frac{1}{\beta_1}} \ln \frac{1}{\beta_1} + \log_{\frac{1}{\beta_1}} m) \sum_{i_1} \beta_1^{i_1} = t^* \sum_{i_1} \beta_1^{i_1} + (\frac{1}{\ln \frac{1}{\beta_1}} + \log_{\frac{1}{\beta_1}} \ln \frac{1}{\beta_1}) \sum_{i_1} \beta_1^{i_1} \tag{27}$$

Lemma 5.1. *Let* $\mathbf{G}$ *and* $\mathbf{G} + \mathbf{E}$ *be nonsingular matrices where* $\mathbf{E}$ *is a matrix of rank one. Let* $g = tr(\mathbf{E}\mathbf{G}^{-1})$. *Then* $g \neq 1$, *and*

$$(\mathbf{G} + \mathbf{E})^{-1} = \mathbf{G}^{-1} - \frac{1}{1+g}\mathbf{G}^{-1}\mathbf{E}\mathbf{G}^{-1}$$

Proof. This Lemma essentially comes from Sherman–Morrison formula, and details of this Lemma can be found in [30].

5.2 Regret Analysis for Decay LinUCB

The proof of LinUCB relies on the fact that the confidence in the estimate of parameters shrinks monotonically. This condition translates to the fact that the eigen values of $\mathbf{A}$ increase monotonically which in turn controls the extent of the exploration. This is no longer true in case of decay LinUCB where the successive decayed weights on the past observations can sometime decrease the eigen values. This effect is

set by the nature of the decay algorithm which wants to constantly explore since it expects forever changing environment with same velocity. This behavior costs the algorithm very dearly resulting in linear regret due to a constant minimal exploration. None the less we put the proof here for a comprehensive report and how to specifically reach the conclusion of linear regret.

Let decay parameter for algorithm be γ and β be the true decay factor that is dependent on the arm. Also let $\mathbf{G} = diag(\gamma^t, ..., \gamma)$

$$\begin{aligned} r_t - \hat{r}_t &= \mathbf{x}_t^\top \hat{\theta}_t - \mathbf{x}_t^\top \theta_t \\ &= \mathbf{x}_t^\top \mathbf{A}_\gamma^{-1} \mathbf{D}^\top \mathbf{G}\mathbf{y} - \mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}(\mathbf{I} + \mathbf{D}^\top \mathbf{G}\mathbf{D})\theta_t \\ &= \mathbf{x}_t^\top \mathbf{A}_\gamma^{-1} \mathbf{D}^\top \mathbf{G}(\mathbf{y} - \mathbf{D}\theta_t) - \mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}\theta_t \end{aligned} \tag{28}$$

Since $\|\theta_t\| \leq 1$.

$$|r_t - \hat{r}_t| \leq |\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1} \mathbf{D}^\top \mathbf{G}(\mathbf{y} - \mathbf{D}\theta_t)| - \|\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}\|$$

Proving for the second component.

$$\begin{aligned} \|\mathbf{x}^\top \mathbf{A}_\gamma^{-1}\| &= \sqrt{\mathbf{x}^\top \mathbf{A}_\gamma^{-1} \mathbf{I} \mathbf{A}_\gamma^{-1} \mathbf{x}} \\ &\leq \sqrt{\mathbf{x}^\top \mathbf{A}_\gamma^{-1}(\mathbf{I} + \mathbf{D}^\top \mathbf{G}\mathbf{D})\mathbf{A}_\gamma^{-1} \mathbf{x}} \\ &\leq \sqrt{\mathbf{x}^\top \mathbf{A}_\gamma^{-1} \mathbf{x}} = s_{t,a} \end{aligned}$$

For the first component we take the following approach.

$$\begin{aligned} |\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1} \mathbf{D}^\top \mathbf{G}(\mathbf{y} - \mathbf{D}\theta_t)| &= |\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1} \sum_{i=1}^{t} \mathbf{x}_i \gamma^{t-i}(r_i - \mathbf{x}_i^\top \theta \beta^t)| \\ &= |\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1} \sum_{i=1}^{t} \mathbf{x}_i \gamma^{t-i}(r_i - \mathbf{x}_i^\top \theta \beta^i) + \mathbf{x}^\top \mathbf{A}_\gamma^{-1} \sum_{i=1}^{t} \mathbf{x}_i \gamma^{t-i} \mathbf{x}_i^\top \theta(\beta^i - \beta^t)| \\ &\leq |\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1} \sum_{i=1}^{t} \mathbf{x}_i \gamma^{t-i}(r_i - \mathbf{x}_i^\top \theta \beta^i)| + |\mathbf{x}^\top \mathbf{A}_\gamma^{-1} \sum_{i=1}^{t} \mathbf{x}_i \gamma^{t-i} \mathbf{x}_i^\top \theta \beta^i| \end{aligned}$$

Since $E(r_i - \mathbf{x}_i^\top \theta \beta^i) = 0$ we can apply Azuma's inequality on the first component which is stochastic. The second component is the bias, and is to be bounded by algebra tricks.

$$P(|\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1} \sum_{i=1}^{t} \mathbf{x}_i \gamma^{t-i}(r_i - \mathbf{x}_i^\top \theta \beta^i)| < \alpha s_{t,a}) \leq 2\exp\Big(-\frac{2\alpha^2 s_{t,a}^2}{\|\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}\mathbf{D}^\top \mathbf{G}\|^2}\Big)$$
$$\leq 2\exp(-2\alpha^2)$$

$$\begin{aligned}\|\mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}\mathbf{D}^\top \mathbf{G}\|^2 &= \mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}\mathbf{D}^\top \mathbf{G}\mathbf{G}\mathbf{D}\mathbf{A}_\gamma^{-1}\mathbf{x}_t \\ &\leq \mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}\mathbf{D}^\top \mathbf{G}\mathbf{D}\mathbf{A}_\gamma^{-1}\mathbf{x}_t \\ &\leq \mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}(\mathbf{I} + \mathbf{D}^\top \mathbf{G}\mathbf{D})\mathbf{A}_\gamma^{-1}\mathbf{x}_t \\ &= \mathbf{x}_t^\top \mathbf{A}_\gamma^{-1}\mathbf{x}_t = s_{t,a}^2\end{aligned}$$

The most strict bound we can give over the difference $\beta^i - \beta^t = \beta^i(1-\beta^{t-i}) \leq \beta^i$. Further more we will use the fact that $\mathbf{x}^\top\theta \leq 1$ and $\|\mathbf{x}\| \leq 1$.

$$\begin{aligned}|\mathbf{x}^\top \mathbf{A}_\gamma^{-1} \sum_{i=1}^{t} \mathbf{x}_i \gamma^{t-i}\mathbf{x}_i^\top \theta(\beta^i - \beta^t)| &\leq \|\mathbf{x}^\top \mathbf{A}_\gamma^{-1} \sum_{i=1}^{t} \gamma^{t-i}\| \\ &= \|\mathbf{x}^\top \mathbf{A}_\gamma^{-1} \sum_{i=0}^{t-1} \gamma^{i}\| \\ &= \|\mathbf{x}^\top \mathbf{A}_\gamma^{-1} \frac{1-\gamma^t}{1-\gamma}\| \\ &= \|\mathbf{x}^\top \mathbf{A}_\gamma^{-1} \rho(t,\gamma)\| \\ &\leq \rho(t,\gamma) s_{t,a}\end{aligned} \tag{29}$$

Lemma 5.2. *Suppose $\psi_{t+1} = \psi_t \cup \{t\}$ in BaseLinUCB, so the eigenvalues of A_{t+1} can be arranged so that $\gamma\lambda_{t,j} + 1 - \gamma \leq \lambda_{t+1,j}$ for all j and,*

$$s_{t,a}^2 \leq 10 \sum_{j=1}^{d} \frac{\lambda_{t+1,j} - \gamma\lambda_{t,j} - 1 + \gamma}{\gamma\lambda_{t,j} + 1 - \gamma}$$

Lemma 5.3. *Using notation in BaseLinUCB and assuming $\Psi_{T+1} \geq 2$, we have, when $0 < \gamma \leq 1$,*

$$\sum_{t\in\Psi_{T+1}} s_{t,a} \leq \sqrt{25\gamma^{-1}d|\Psi_{T+1}|\ln|\Psi_{T+1}| + 10d|\Psi_{T+1}|^2(\gamma^{-1}-1)}$$

Proof. For convenience, define $\psi = |\Psi_{T+1}|$, Lemma 3.2 implies,

$$\sum_{t\in\Psi_{T+1}} s_{t,a} \leq \sum_{t\in\Psi_{T+1}} \sqrt{10\sum_{j=1}^{d}(\frac{\lambda_{t+1,j}}{\gamma\lambda_{t,j}+1-\gamma}-1)} \tag{30}$$

The function

$$f = \sum_{t\in\Psi}\sqrt{\sum_{j=1}^{d}(h_{tj}-1)} \tag{31}$$

is maximized under the constraints, $h_{tj} \geq 1$ and $\sum_{j=1}^{d}\prod_{t\in\Psi} h_{tj} \leq C$, when

$$h_{tj} = (\frac{C}{d})^{1/|\Psi|} \tag{32}$$

for all $t \in \Psi$ and $j \in [d]$.

In our context, we have,

$$\frac{\lambda_{t+1,j}}{\gamma\lambda_{t,j}+1-\gamma} \geq 1 \tag{33}$$

When $0 < \gamma \leq 1$

$$\begin{aligned}\sum_{j=1}^{d}\prod_{t\in\Psi_{T+1}}\frac{\lambda_{t+1,j}}{\gamma\lambda_{t,j}+1-\gamma} &\leq \sum_{j=1}^{d}\prod_{t\in|\Psi_{T+1}|}\frac{\lambda_{t+1,j}}{\gamma\lambda_{t,j}}\\ &= \gamma^{-|\Psi_{T+1}|}\sum_{j=1}^{d}\lambda_{T+1,j}\\ &= \gamma^{-|\Psi_{T+1}|}(\sum_{t\in|\Psi_{T+1}|}\|x_{t,a_t}\|^2+d)\\ &\leq \gamma^{-\psi}(\psi+d)\end{aligned} \tag{34}$$

and so,

$$\begin{aligned}\sum_{t\in\Psi_{T+1}} s_{t,a} &\leq \psi\sqrt{10d}\sqrt{(\frac{\gamma^{-\psi}(\psi+d)}{d})^{1/\psi}-1}\\ &\leq \psi\sqrt{10d}\sqrt{\gamma^{-1}(\psi+1)^{1/\psi}-1}\end{aligned} \tag{35}$$

When $\psi \geq 2$, according to the fact that,

$$(\psi+1)^{1/\psi} - 1 \leq \frac{2.5}{\psi}\ln\psi \tag{36}$$

we have,

$$\gamma^{-1}(\psi+1)^{1/\psi}-1 \leq \gamma^{-1}\frac{2.5}{\psi}\ln\psi+\gamma^{-1}-1 \tag{37}$$

and hence, when $0<\gamma\leq 1$,

$$\begin{aligned}\sum_{t\in\Psi_{T+1}} s_{t,a} &\leq \sqrt{10d\psi^2(\gamma^{-1}\frac{2.5}{\psi}\ln\psi+\gamma^{-1}-1)} \\ &= \sqrt{25\gamma^{-1}d\psi\ln\psi+10d\psi^2(\gamma^{-1}-1)}\end{aligned} \tag{38}$$

Notice due to ψ^2 is $\mathcal{O}(T)$ the variance could not be bounded by anything smaller than T. Hence the upper bound is linear.

5.3 Regret Analysis for Restart LinUCB

In this section we provide detailed regret analysis of Restart LinUCB algorithm. Restart LinUCB is restarting LinUCB with doubling intervals or phase lengths. Phase is indexed by j and length of phase j is $t_j = 2^j$. Let decay parameters for algorithm β be the true decay factor that is dependent on the arm. According to Problem formulation in Section 3, and following the proof of Lemma 9 in [9] for LinRel algorithm and Lemma 1 in [10] for LinUCB algorithm, Lemma 1 for Restart LinUCB can be proved.

Theorem 5.4. *Suppose the input index set Ψ_t for the j-th interval in Restart BaseLinUCB is constructed so that for fixed x_τ with $\tau\in\Psi_t$, the reward r_τ are independent random variables with means $E(r_\tau)=\beta^\tau\mathbf{x}_\tau^\top\boldsymbol{\theta}^*$. Then with probability at least $1-\delta/T$, we have,*

$$|r_{\tau,a_\tau}-\hat{r}_{\tau,a_\tau}| \leq (\alpha+\beta^\tau+d^{\frac{3}{2}}\sum_{i=t_j}^{t}\beta^i)s_{\tau,a_\tau}$$

in which, $s_{\tau,a_\tau}=\sqrt{\mathbf{x}_\tau^\top\mathbf{A}_\tau^{-1}\mathbf{x}_\tau}$

Proof. Using notations from previous sections , for the j-th interval in Restart-LinUCB,

$$\begin{aligned}\hat{r}_{\tau,a_\tau}-r_{\tau,a_\tau} &= \mathbf{x}_\tau^\top\mathbf{A}_\tau^{-1}\mathbf{D}_\tau^\top\mathbf{y}-\beta^\tau\mathbf{x}_\tau^\top\mathbf{A}_\tau^{-1}(\mathbf{I}+\mathbf{D}_\tau^\top\mathbf{D}_\tau)\boldsymbol{\theta}_\tau \\ &= \mathbf{x}_\tau^\top\mathbf{A}_\tau^{-1}\mathbf{D}_\tau^\top(\mathbf{y}-\beta^\tau\mathbf{D}\boldsymbol{\theta}_\tau)-\beta^\tau\mathbf{x}_\tau^\top\mathbf{A}_\tau^{-1}\boldsymbol{\theta}_\tau\end{aligned} \tag{39}$$

And since $\|\boldsymbol{\theta}_\tau\| \leq 1$,

$$|r_{\tau,a_\tau} - \hat{r}_{\tau,a_\tau}| \leq \|\beta^\tau \mathbf{x}_\tau^\top \mathbf{A}_\tau^{-1}\| + |\mathbf{x}_\tau^\top \mathbf{A}_\tau^{-1} \mathbf{D}_\tau^\top (\mathbf{y} - \beta^t \mathbf{D}_\tau \boldsymbol{\theta}_\tau)| \tag{40}$$

For the first component of Equ. 40,

$$\begin{aligned} \beta^\tau \|\mathbf{x}_\tau^\top \mathbf{A}_\tau^{-1}\| &\leq \beta^\tau \sqrt{\mathbf{x}_\tau^\top \mathbf{A}_\tau^{-1} (\mathbf{I} + \mathbf{D}_\tau^\top \mathbf{D}_\tau) \mathbf{A}_\tau \mathbf{x}_\tau} \\ &\leq \beta^\tau \sqrt{\mathbf{x}_\tau^\top \mathbf{A}_\tau^{-1} \mathbf{x}_\tau} = \beta^\tau s_{\tau,a_\tau} \end{aligned} \tag{41}$$

For the second component of Equ. 40 we take the following approach.

$$\begin{aligned} &|\mathbf{x}_\tau^\top \mathbf{A}_\tau^{-1} \mathbf{D}_\tau^\top (\mathbf{y}_\tau - \beta^\tau \mathbf{D}_\tau \boldsymbol{\theta}_\tau)| \\ &\leq |\mathbf{x}_t^\top \mathbf{A}_\tau^{-1} \sum_{i=t_j}^{\tau} \mathbf{x}_i (r_i - \mathbf{x}_i^\top \boldsymbol{\theta} \beta^i)| + |\mathbf{x}_\tau^\top \mathbf{A}_\tau^{-1} \sum_{i=t_j}^{\tau} \mathbf{x}_i \mathbf{x}_i^\top \boldsymbol{\theta} (\beta^i - \beta^\tau)| \end{aligned} \tag{42}$$

Since $E(r_i - \mathbf{x}_i^\top \boldsymbol{\theta} \beta^i) = 0$ we can apply Azuma's inequality on the first component of Equ. 42, which is stochastic. The second component is the bias, and is to be bounded by algebra tricks.

$$\begin{aligned} &P(|\mathbf{x}_\tau^\top \mathbf{A}^{-1} \sum_{i=1}^{\tau} \mathbf{x}_i (r_i - \mathbf{x}_i^\top \boldsymbol{\theta} \beta^i)| < \alpha s_{\tau,a_\tau}) \\ &\leq 2 \exp\Big(- \frac{2\alpha^2 s_{\tau,a_\tau}^2}{\|\mathbf{x}_\tau^\top \mathbf{A}^{-1} \mathbf{D}^\top\|^2} \Big) \leq 2 \exp(-2\alpha^2) \end{aligned} \tag{43}$$

in which the last inequality is based on the following fact,

$$\begin{aligned} \|\mathbf{x}_\tau^\top \mathbf{A}^{-1} \mathbf{D}^\top\|^2 &\leq \mathbf{x}_\tau^\top \mathbf{A}^{-1} (\mathbf{I} + \mathbf{D}^\top \mathbf{D}) \mathbf{A}^{-1} \mathbf{x}_\tau \\ &= \mathbf{x}_\tau^\top \mathbf{A}^{-1} \mathbf{x}_\tau = s_{\tau,a_\tau}^2 \end{aligned} \tag{44}$$

Using limits on l_2 norm on $\mathbf{x}$ and $\boldsymbol{\theta}$, if $0 \leq \beta < 1$, and since $t_j \leq \tau \leq 2t_j$ and $\sum_{i=t_j}^{\tau} (\beta^i - \beta^\tau)$ is an increasing function with t,

$$\begin{aligned} |\mathbf{x}^\top \mathbf{A}^{-1} \sum_{i=t_j}^{\tau} \mathbf{x}_i \mathbf{x}_i^\top \boldsymbol{\theta} (\beta^i - \beta^\tau)| &\leq d^{\frac{3}{2}} \|\mathbf{x}^\top \mathbf{A}_\tau^{-1} \sum_{i=t_j}^{\tau} \beta^i\| \\ &\leq d^{\frac{3}{2}} s_{\tau,a_\tau} \sum_{i=t_j}^{\tau} \beta^i \end{aligned} \tag{45}$$

Combining Equ.41, 42, 43 and 45 finishes the proof. □

Define, $|\Psi_{T+1}| = \psi$. According to Lemma 40, summing regret from all the intervals, we have,

$$\begin{aligned}\sum_{t=1}^{\Psi_{T+1}} (\hat{r}_{t,a_t} - r^*_{t,a_t}) &\leq \sum_{j=0}^{log_2\psi} (1+\alpha+d^{\frac{3}{2}} \sum_{t_0=t_j}^{t_{j+1}} \beta^{t_0}) \sum_{t=t_j}^{t_{j+1}} s_{t,a_t} \\ &\leq \sum_{j=0}^{log_2\psi} (1+\alpha+d^{\frac{3}{2}} \sum_{t_0=t_j}^{t_{j+1}} \beta^{t_0}) \sqrt{dt_j \ln t_j}\end{aligned} \tag{46}$$

According to the decomposition of BaseLinUCB/SupLinUCB in [10], if SupLinUCB is run with $\alpha = \sqrt{\frac{1}{2}\ln\frac{2Kt_j}{\delta}}$ at every interval j, then with probability at least $1-\delta$, the regret Restart-LinUCB algorithm is,

$$\mathcal{O}(\sum_{j=0}^{log_2 T} \sum_{t_0=t_j}^{t_{j+1}} \beta^{t_0} \sqrt{t_j d \ln^3(Kt_j \ln(t_j))/\delta)}) \tag{47}$$

or

$$O(\sum_{j=0}^{log_2 T} \sqrt{t_j d \ln^3(Kt_j \ln(t_j))/\delta} + 5d^{\frac{3}{2}} \frac{\beta(1-\beta^{\psi})}{1-\beta} \sqrt{Td \ln^3(KT \ln(T))/\delta}) \tag{48}$$

Which is simply:

$$O\left(log_2(T) \sqrt{\frac{Td \ln^3(KT \ln(T))}{\delta}}\right) \tag{49}$$

5.4 Regret Analysis for Temporal LinUCB

In this section we provide detailed regret analysis of our proposed Temporal LinUCB algorithm using similar tricks as in the regret analysis for Restart LinUCB.

Suppose the input index set Ψ_t in Temporal LinUCB is constructed such that for fixed x_{τ,a_τ} with $\tau \in \Psi_t$, the reward r_τ are independent random variables with means $E(r_\tau) = \beta^\tau \mathbf{x}_\tau^\top \boldsymbol{\theta}^*$. Define $s_{\tau,a_\tau} = \sqrt{\mathbf{x}_{\tau,a}^T \mathbf{A}_\tau^{-1} \mathbf{x}_{\tau,a}}$ Then we have,

Using the notion in Section 1, we have,

$$\begin{aligned}&r^*_{\tau,a} - \hat{r}_{\tau,a} = \beta^{*\tau} \mathbf{x}_{\tau,a}^T \boldsymbol{\theta}^* - \hat{\beta}^t \mathbf{x}_{\tau,a}^T \hat{\boldsymbol{\theta}}_\tau \\ &= \beta^{*\tau} \mathbf{x}_{\tau,a}^T \mathbf{A}_\tau^{-1} \boldsymbol{\theta}^* \\ &+ \mathbf{x}_{\tau,a}^T \mathbf{A}_\tau^{-1} (\beta^{*t} \sum_{i=1}^{\tau} \mathbf{x}_{i,a} \mathbf{x}_{i,a}^T \boldsymbol{\theta}^* - \hat{\beta}^\tau \sum_{i=1}^{\tau} \mathbf{x}_{i,a} \hat{\beta}^{-i} y_{i,a})\end{aligned} \tag{50}$$

Since $\|\boldsymbol{\theta}^*\| \leq 1$

$$
\begin{aligned}
&|r^*_{\tau,a} - \hat{r}_{\tau,a}| \leq \|\beta^{*\tau}\mathbf{x}^T_{\tau,a}\mathbf{A}^{-1}_{\tau}\| \\
&+ |\mathbf{x}^T_{\tau,a}\mathbf{A}^{-1}_{\tau}(\beta^{*\tau}\sum_{i=1}^{\tau}\mathbf{x}_{i,a}\mathbf{x}^T_{i,a}\boldsymbol{\theta}^* - \hat{\beta}^{\tau}\sum_{i=1}^{\tau}\mathbf{x}_{i,a}\hat{\beta}^{-i}y_{i,a})|
\end{aligned} \tag{51}
$$

Similar to Equ. 41 we can bound the first term in Equ.51:

$$
\|\beta^{*\tau}\mathbf{x}^T_{\tau,a}\mathbf{A}^{-1}_{\tau}\| \leq \beta^{*\tau}s_{\tau,a_\tau} \tag{52}
$$

For the second term in Equ.51, we can rewrite it as,

$$
\begin{aligned}
&|\mathbf{x}^T_{\tau,a}\mathbf{A}^{-1}_{\tau}(\beta^{*\tau}\sum_{i=1}^{\tau}\mathbf{x}_{i,a}\mathbf{x}^T_{i,a}\boldsymbol{\theta}^* - \hat{\beta}^{\tau}\sum_{i=1}^{\tau}\mathbf{x}_{i,a}\hat{\beta}^{-i}y_{i,a})| \\
&= |\mathbf{x}^T_{\tau,a}\mathbf{A}^{-1}_{\tau}\sum_{i=1}^{\tau}\mathbf{x}_{i,a}\Big[\beta^{*(\tau-i)}(\beta^{*i}\mathbf{x}^T_{i,a}\boldsymbol{\theta}^* - y_{i,a}) \\
&+ (\beta^{*(\tau-i)} - \hat{\beta}^{\tau-i})y_{i,a}\Big]| \\
&\leq |\mathbf{x}^T_{\tau,a}\mathbf{A}^{-1}_{\tau}\sum_{i=1}^{\tau}\mathbf{x}_{i,a}(\beta^{*i}\mathbf{x}^T_{i,a}\boldsymbol{\theta}^* - y_{i,a})| \\
&+ |\mathbf{x}^T_{\tau,a}\mathbf{A}^{-1}_{\tau}\sum_{i=1}^{\tau}\mathbf{x}_{i,a}(\beta^{*(\tau-i)} - \hat{\beta}^{\tau-i})|
\end{aligned} \tag{53}
$$

The inequality is because of $0 < \beta^* < 1$ and for $\forall a, i$, $|y_{i,a}| \leq 1$.

The first part of Equ.53 can be bounded with s_{t,a_t} with high probability according to Azuma's Inequality. Therefore, from theoretical point of view, to bound the regret on TUCB, it requires us to find the rate of convergence for such an estimator, which can guarantee the second component of Equ.53 to be bounded with s_{t,a_t}. Unfortunately we could not find any statistical properties that establish the rate of convergence for such an estimator except for the asymptotic results due to MLE. We think this is the branch of theory which has yet to be explored. Since the proof is not attainable, the optimal index design could not be inferred and hence the TUCB is not the optimal version of the idea of parametric non-stationary bandits. For this reason the implementation of TUCB approximates the factor $\rho_{t,a} = 1$ in step 4, the true value of which depends on the rate of convergence of β estimation, of algorithm 3. This factor is pertinent to achieve the right amount of exploration to get a tight upper bound on the regret, with this factor assumed to be 1, exploration is enhanced giving rise to more regret. Apart from theoretical difficulties, TUCB is also implementation

heavy and naturally not online. Due to the non-linear factor requiring numerical optimization methods, the algorithm has to store the entire history which is also indicated by the fact that the sufficient statistic is all the contexts observed, arms played and the time stamps when the action was played. Despite all these drawbacks, TUCB is significant since it is the best possible MAB strategy; a consequence of its explicit assumption.

5.4.1 Asymptotic results

Although these results did not prove useful, another idea is to use the MLE asymptotic results. We state these results here just in case they could be useful in other domains.

Starting off, first lets evaluate the fisher score.

$$I(\beta) = -E(\frac{\partial^2 \log f(\mathbf{D}, \mathbf{y}|\beta)}{\partial \beta^2})$$

$$\log f(\mathbf{D}, \mathbf{y}|\beta) = -\frac{t}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=0}^{t}(\frac{y_i}{\beta^{t_i}} - \theta_t^\top \mathbf{x}_t)^2$$

$$\begin{aligned} \frac{\partial \log f(\mathbf{D}, \mathbf{y}|\beta)}{\partial \beta} &= \frac{1}{\sigma^2}\sum_{i=0}^{t}(\frac{y_i}{\beta^{t_i}} - \theta_t^\top \mathbf{x}_t)\frac{y_i t_i}{\beta^{t_i+1}} \\ &= \frac{1}{\sigma^2}\sum_{i=0}^{t}(\frac{y_i^2 t_i}{\beta^{2t_i+1}} - \frac{\theta_t^\top \mathbf{x}_t y_i t_i}{\beta^{t_i+1}}) \end{aligned} \tag{54}$$

$$\frac{\partial^2 \log f(\mathbf{D}, \mathbf{y}|\beta)}{\partial \beta^2} = \frac{1}{\sigma^2}\sum_{i=0}^{t}(\frac{-y_i^2 t_i(2t_i+1)}{\beta^{2t_i+2}} + \frac{\theta_t^\top \mathbf{x}_t y_i t_i(t_i+1)}{\beta^{t_i+2}}) \tag{55}$$

$$\begin{aligned} -E(\frac{\partial^2 \log f(\mathbf{D}, \mathbf{y}|\beta)}{\partial \beta^2}) &= \frac{1}{\sigma^2}\sum_{i=0}^{t}\left[\frac{E(y_i^2)t_i(2t_i+1)}{\beta^{2t_i+2}} - \frac{\theta_t^\top \mathbf{x}_t E(y_i)t_i(t_i+1)}{\beta^{t_i+2}}\right] \\ &= \frac{1}{\sigma^2}\sum_{i=0}^{t}\left[\frac{(\beta^{2t_i}\sigma^2 + \beta^{2t_i}(\theta^\top \mathbf{x}_t)^2)t_i(2t_i+1)}{\beta^{2t_i+2}} - \frac{\theta_t^\top \mathbf{x}_t \beta^{t_i}\theta^\top \mathbf{x}_t t_i(t_i+1)}{\beta^{t_i+2}}\right] \\ &= \frac{1}{\sigma^2}\sum_{i=0}^{t}\left[\frac{(\sigma^2 + (\theta^\top \mathbf{x}_t)^2)t_i(2t_i+1)}{\beta^2} - \frac{(\theta_t^\top \mathbf{x}_t)^2 t_i(t_i+1)}{\beta^2}\right] \\ &= \frac{1}{\sigma^2}\sum_{i=0}^{t}\left[\frac{\sigma^2 t_i(2t_i+1) + (\theta^\top \mathbf{x}_t)^2(t_i^2)}{\beta^2}\right] \end{aligned} \tag{56}$$

Using the MLE property,

$$\begin{aligned}\sqrt{t}(\widehat{\beta}_t - \beta^*) &\rightarrow^D \mathcal{N}(0, I(\beta)^{-1}) \\ &\rightarrow^D \mathcal{N}(0, \frac{\sigma^2\beta^2}{\sum_{i=0}^{t}\Big[\sigma^2 t_i(2t_i+1) + (\theta^\top \mathbf{x}_t)^2(t_i^2)\Big]})\end{aligned} \tag{57}$$

In general MLE property gives more.

$$\begin{aligned}\sqrt{t}(\tau(\widehat{\beta}_t) - \tau(\beta)) &\rightarrow^D \mathcal{N}(0, \frac{[\tau'(\beta)]^2}{I(\beta)}) \\ \sqrt{t}(\widehat{\beta}_t^c - \beta^c) &\rightarrow^D \mathcal{N}(0, \frac{(\sigma c \beta^c)^2}{\sum_{i=0}^{t}\Big[\sigma^2 t_i(2t_i+1) + (\theta^\top \mathbf{x}_t)^2(t_i^2)\Big]})\end{aligned} \tag{58}$$

This is the asymptotic result in variance of MLE estimate. However in our proof we need non-asymptotic result although a confidence bound could be inspired from this equation but we did not make any progress in this direction.

5.4.2 Variance of Non-stationary Oracle

Another method to figure out the form of the confidence bound is to calculate the variance of the process directly. This can provide us with the form of the confidence bound for which just the right constant has to be figured out.

Variance in Ridge Regression

Let us drive the variance for the Ridge regression and then follow the same procedure to drive variance for our non-stationary oracle. Let $\mathbf{r}_t$ be the vector of rewards till time t. Let $\widehat{\boldsymbol{\theta}}_t$ be the estimate of θ at time t.

$$\widehat{\boldsymbol{\theta}} = \mathbf{A}_t^{-1}\mathbf{D}_t^\top \mathbf{r}_t$$

$$p(\theta) = \mathcal{N}(\theta|0, \sigma^2\mathbf{I})$$

$$p(\mathbf{r}_t|\theta) = \mathcal{N}(\mathbf{r}_t|\mathbf{D}_t\theta, \sigma^2\mathbf{I})$$

Using Bayes theorem for Gaussian variables, we can figure out $p(\theta)$.

$$\begin{aligned} p(\theta|\mathbf{r}) &= \mathcal{N}(\theta|\widehat{\boldsymbol{\theta}}, \boldsymbol{\Sigma}) \\ \boldsymbol{\Sigma} &= (\frac{\mathbf{I}}{\sigma^2} + \frac{\mathbf{D}^\top \mathbf{D}}{\sigma^2})^{-1} \\ \boldsymbol{\Sigma} &= \sigma^2(\mathbf{D}^\top \mathbf{D} + \mathbf{I})^{-1} = \sigma^2 \mathbf{A}^{-1} \end{aligned}$$

Bayesian point estimate

Using Bayes theorem we get a different answer. Let the prior on the θ be $\pi(\theta)$.

$$\begin{aligned} \pi(\theta) &= \mathcal{N}(\theta|0, \sigma^2) \\ \mathcal{L}(\theta|\mathbf{r}_t) &= \prod_{i=1}^{t} \mathcal{N}(r_i|\theta^\top \mathbf{x}_i, \sigma^2) \end{aligned} \tag{59}$$

Here we calculate the posterior probability.

$$\begin{aligned} p(\theta|\mathbf{D}) &\propto \pi(\theta)\mathcal{L}(\theta|\mathbf{r}_t) \\ \pi(\theta)\mathcal{L}(\theta|\mathbf{r}_t) &= \frac{1}{(2\pi\sigma^2)^{\frac{t+1}{2}}} \exp(-\frac{(\mathbf{r}_t - \mathbf{D}\theta)^\top(\mathbf{r}_t - \mathbf{D}\theta)}{2\sigma^2} - \frac{\theta^\top \theta}{2\sigma^2}) \end{aligned} \tag{60}$$

We will look at the expression inside the exponential, since it determines the nature of the probability distribution. Since the Conjugate prior of Gaussian likelihood is also Gaussian, we can analyze the expression inside exponential to drive the variance formula.

$$\begin{aligned} \frac{(\mathbf{r}_t - \mathbf{D}\theta)^\top(\mathbf{r}_t - \mathbf{D}\theta)}{2\sigma^2} + \frac{\theta^\top \theta}{2\sigma^2} &= \frac{\mathbf{r}_t^\top \mathbf{r}_t - 2\mathbf{r}_t^\top \mathbf{D}\theta + \theta^\top \mathbf{D}^\top \mathbf{D}\theta + \theta^\top \theta}{2\sigma^2} \\ &= \frac{const - 2\mathbf{r}_t^\top \mathbf{D}\theta + \theta^\top(\mathbf{D}^\top \mathbf{D} + \mathbf{I})\theta}{2\sigma^2} \end{aligned} \tag{61}$$

The second order term in the above expression determines the variance of the Gaussian posterior. It is the inverse of the term in the product. It is clear that the variance is $2\sigma^2\mathbf{A}^{-1}$.

Variance of Non-stationary Oracle $\beta\theta$

As a warm up let us start with a simple case where we know the decaying factor β. This case essentially does not account for the variance in β which may not play any significant role since we observe an exponential decay. Again we define $\mathbf{G}_t = diag(\beta, \beta^2, ..., \beta^t)$.

$$\begin{aligned} p(\theta) &= \mathcal{N}(\theta|0, \sigma^2\mathbf{I}) \\ p(\mathbf{r}_t|\theta) &= \mathcal{N}(\mathbf{r}_t|\mathbf{G}_t\mathbf{D}_t\theta, \sigma^2\mathbf{G}_t^2) \end{aligned}$$

Following the same procedure as for the

$$
\begin{aligned}
p(\theta|\mathbf{r}_t) &= \mathcal{N}(\theta|\widehat{\boldsymbol{\theta}}, \boldsymbol{\Sigma}) \\
\boldsymbol{\Sigma} &= (\frac{\mathbf{I}}{\sigma^2} + \frac{\mathbf{D}_t^\top \mathbf{G}_t \mathbf{G}^{-2} \mathbf{G}_t \mathbf{D}_t}{\sigma^2})^{-1} \\
\boldsymbol{\Sigma} &= \sigma^2 (\mathbf{D}_t^\top \mathbf{D}_t + \mathbf{I})^{-1} = \sigma^2 \mathbf{A}^{-1}
\end{aligned}
$$

Like LinUCB we use predictive variance for $\mathbf{x}_t$ which will be given by.

$$
\mathbf{w} = \beta^t \sqrt{\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x}}
$$

This bound gives us promise that the similar looking confidence bounds are good. Notice the lingering β leading the bound formula however it requires the true value of the parameter which makes it useless. For the lack of better bound we use this formula without the β expression.

6 Results

In [6] authors experiment with Yahoo! dataset of Yahoo! front page module where the recommendations are made from an article pool, articles stay in the pool for a day or so and then are taken off the pool. This is also non-stationary in a sense to which LinUCB is very accommodating, since at the entrance of a new article due to its large confidence width it gets thoroughly explored. Our setting can be considered a more relaxed and realistic version of the scenario faced where it is not clear when to take the article off the website, or in general action from the pool, since its CTR is non-stationary. In the experiment below we simulate the user and articles such that the CTR of articles decay with time and the optimal action can also change with it.

We carry out testing in simulation controlled environment. We design actions such that optimal arms switch during the experiment hence strengthening the effect of the non-stationary behavior. In this experiment we use two dimensional feature vectors $\mathbf{x}$ and linear parameters $\boldsymbol{\theta}$ to represent users, articles and the linear mapping. Further more the reward is linear in the context feature vector and decays exponentially with time, following the dynamics stated in equ. 8. The feature vectors are uniformly chosen from the space $\|\mathbf{x}\| \leq 1$ and β is chosen based on the $\boldsymbol{\theta}$ such that higher starting rewards are more likely to get lower decay parameters. This is done in order to promote the environment where there are more switches between the optimal arms. This is done to evaluate the algorithms when non-stationary in the environment affects

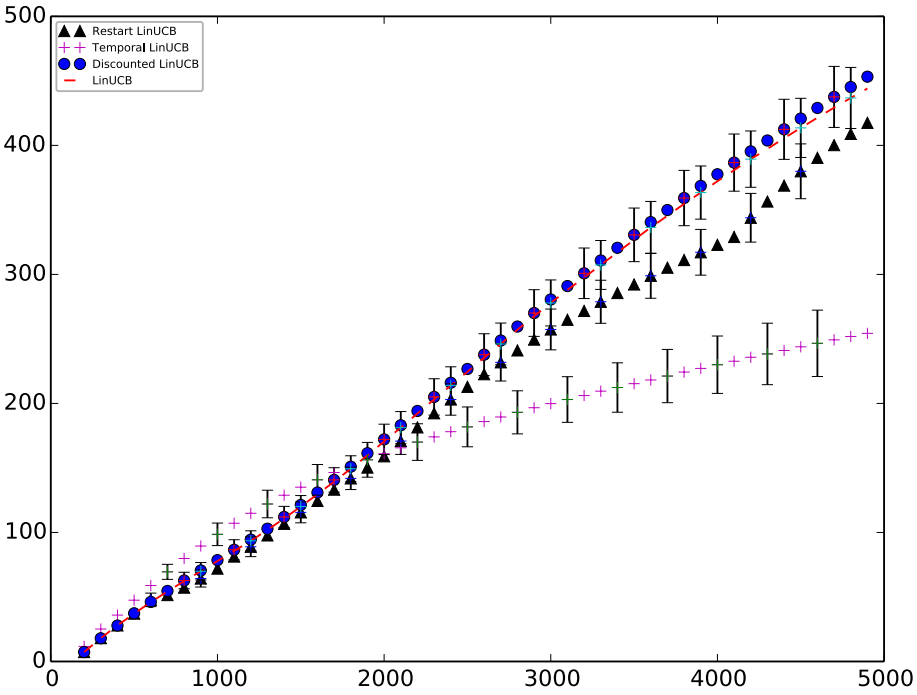

Figure 12: Regret curves for the three methods as compared to the LinUCB algorithm.

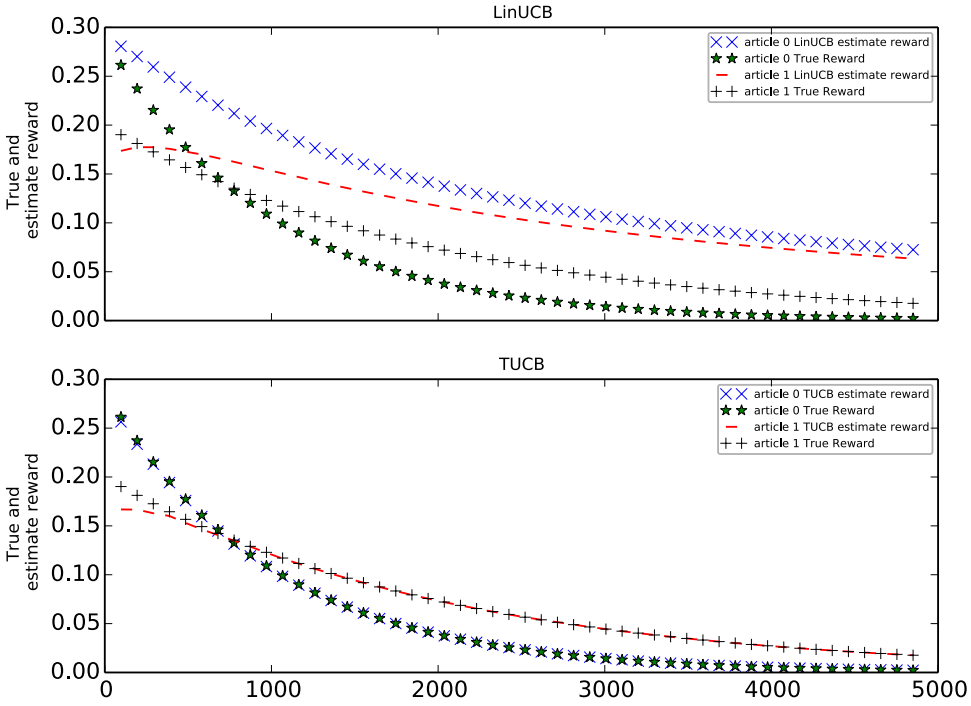

Figure 13: Optimal arm switch.

the order of the recommendations. This experiment highlights a situation where there are articles in the pool that are undergoing gradual loss in popularity after going viral mixed with articles that draw moderate attention but are relatively long lived and their popularity decays much slower until they are taken off the site. In this dynamic environment we test our given approaches and vet them against LinUCB. We use 50 articles, 1000 users and run the experiment for 5 thousand iterations. We repeat the experiment 20 times and average the curves, we also plot vertical bars representing the standard deviation. Also since TUCB is implementation heavy we run decay estimation only in batches after the action has accumulated a good number of observations. We found that with the right setting of parameter it does not make much of a difference in the final results. We used threshold of 300 and ran optimization every 10th iteration after that.

Figure 12 shows the result of running the proposed algorithms. In this environment the order of performance is clear. Temporal LinUCB beats all which is validation of our idea. Unlike other algorithms, TUCB knows the setting and goes after estimating the decay directly. This assumption is very specific but will pay off if the environment really holds exponentially decaying rewards. The downside is when the assumption does not hold or the decay is not strictly exponential, it will hurt TUCB the most. In our empirical results we found TUCB to perform well specifically in the estimation of the decay parameter β. This is illustrated in the bottom plot of figure 13 which shows how well TUCB track the true reward of an action. The figure draws the true reward for an action with a single user, a context free setting, and highlights the non-stationary aspect of the environment. The purpose of the figure is to explore how different algorithms track the true reward. In this case the curves of true reward cross each other and we can observe that TUCB is doing a good job at converging its estimates to the true rewards. Since it models the reward function exactly, it can predict when two actions will be switching and hence will incur very minimal opportunity loss or regret.

We see LinUCB and DecayLinUCB perform very similar, this is due to the reason that we set the decay to a very small value of $1 - 10^{-4}$. We also tried the algorithm with smaller decay however the results are worse in that case. The prime reason for poor results of decay LinUCB is its continuous exploration which accumulates regret linearly. LinUCB follows the same regret bound but owes to different reasons. The main difficulty for LinUCB is its inappropriate estimator that gives equal weight to the examples in the history. This leads the algorithm to over estimate the true reward which has in fact decayed. Due to this over estimation if the optimal arm has switched, LinUCB will not realize it immediately and will keep playing the action that it still thinks is optimal. LinUCB does great on stationary environments because it plays the action it thinks is optimal. This over estimation compels LinUCB to choose the previously optimal action, and now sub-optimal, even more, hence it harbors highly weighted history which delays its realization of

the optimal action switch by a great deal, as shown in figure 13. This delay intuitively makes the LinUCB linear; the delay in LinUCB switch to the actual true switching point will be proportional to the length of the history before the true switching point. Since the switch can happen at time $\frac{T}{2}$ LinUCB in that case will not recognize it until after the end of the experiment and will incur linear regret $\mathcal{O}(T)$. The LinUCB's tendency to over estimate has an awkward upside; if despite decaying, one action remains optimal through out the experiment, then LinUCB due to its overestimation property will choose that arm even more frequently than any other algorithm. This gives LinUCB an unparalleled advantage, but since this case is not an interesting one we discard it in our experiments by controlling the environment to prefer situations where actions true rewards cross each other.

Restart LinUCB does slightly better than LinUCB as depicted in the figure 12. In practice we observe that restarting LinUCB has both pros and cons. While restarting gives a fresh start, it also makes the algorithm to completely discard the history. In the figure we observe initial restart has helped this algorithm to produce lower regret until at iteration 4096 the restart makes it forget everything followed by an immediate rise in regret which can be explained by newly started exploration required by the algorithm to re-learn the parameters. Again this shows that just by relieving the algorithm of its weight of its history can lead to performance boost as shown in the figure.

From the figures and discussion we conclude that TUCB is the superior algorithm however to tune it even further requires us to finish the proof and is an open problem. If it is known that rewards are decaying exponentially TUCB is the clear choice. For decay LinUCB we find that it is not appropriate for our setting and does as worse as LinUCB due to its constant exploration of all actions. We also find that restarting the algorithm has some value and shows off in the boosted performance of the restart LinUCB.

6.1 Experiment on Yahoo! data

Granularity was set to 500 observations i.e. statistics were aggregated for every 500 observations. DecayLinUCB was run with decay parameter which was multiplied with the articles design matrix D and the weighted reward vector b at every time stamp. (For D and b refer to [6]). There are only 289 distinct timestamps for 1st day of Yahoo! dataset. This means that Yahoo buckets large number of observations together and give it the same time stamp. This means that we can not have granularity as small as in the simulations. Also in Yahoo! dataset only 20 or so new articles are added every day where we have hundred thousands of observations on each article. Black vertical lines represent new articles incoming. Results for the first day of Yahoo! dataset are given. CTR is aggregated over 289 distinct time stamps. Decay to the article D and b

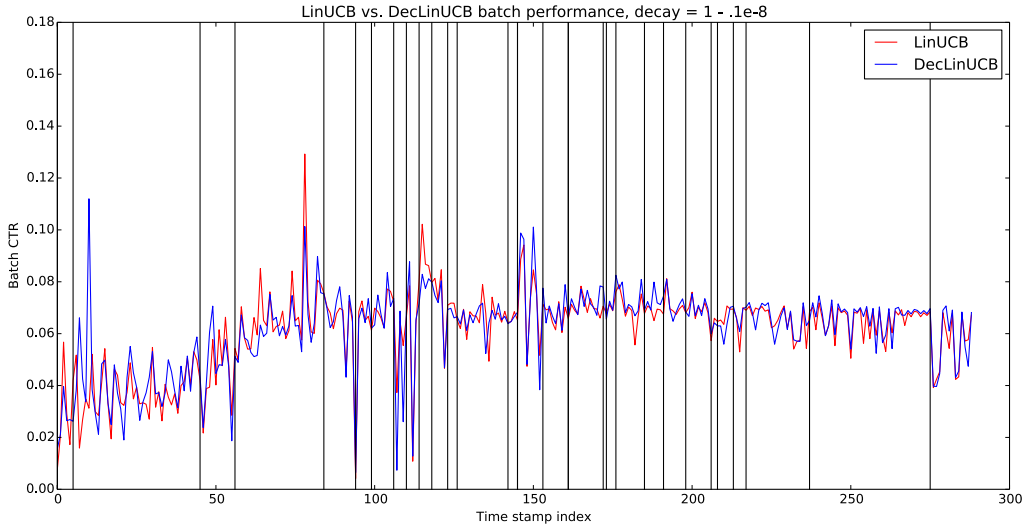

Figure 14: DecayLinUCB with decay=$1 - .1^8$

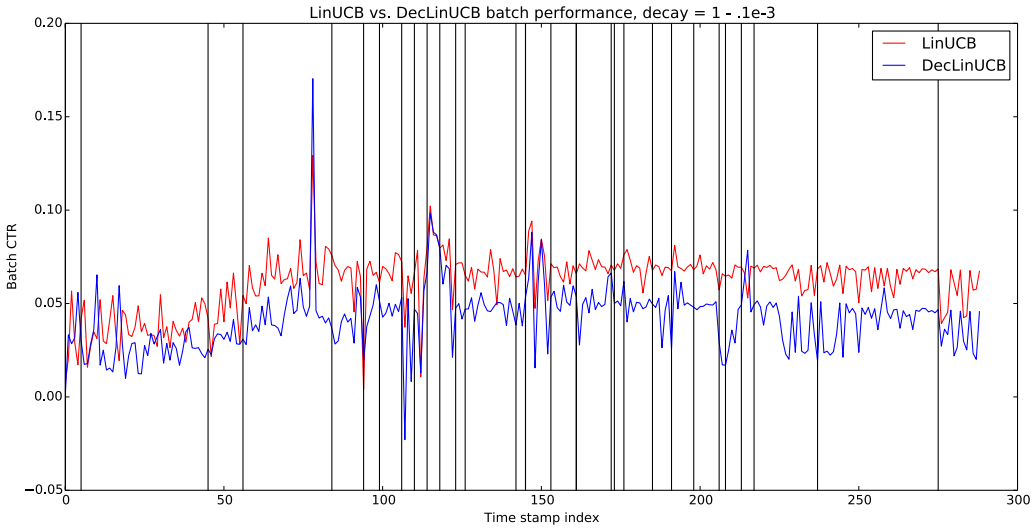

Figure 15: DecayLinUCB does worse with decay .999

is only applied when the time stamp for the articles changes. Hence in the case of first day, decay is applied only 289 times.

In the final CTR we see that decay-LinUCB does very slightly better than LinUCB as for settings in fig. 14.

7 Conclusion and Discussion

In this study we took on the problem of non-stationary meta-trends in user preferences and we proposed three approaches following the promising ideas and popular frameworks. We conclude with an approach which is not provable but shows promise empirically. This approach, called TUCB, follows the UCB style algorithm and constructs the most strict assumption to learn the parameters of the environment. We conclude that if the environment can be parametrized then UCB type framework with explicit estimation of those parameters can give the best results although it may not be provable in all cases. We illustrated this by framing an exponentially decaying rewards setting where TUCB approximated a linear and a non-linear decay parameter simultaneously. With this advancement we also came across an open problem of convergence analysis on estimation of such non-linear parameters.

We also showed that in cases where TUCB type specification is not achievable one can take advantage of the side effects of setting which leads to an algorithm with relatively mild assumptions. This has indeed shown relatively better results in theory and practice. Lastly we see that the most general idea of decaying the past is not always better, since it did not output any better results than the base line of LinUCB. We both confirm this in proof and simulation. Although in the world of non-stationary behaviors there are many cases where decay type idea is the best one can do however, it does not fit our example.

References

[1] Thompson, William R. "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples." Biometrika (1933): 285-294.

[2] Zelen, Marvin. "Play the winner rule and the controlled clinical trial." Journal of the American Statistical Association 64.325 (1969): 131-146.

[3] Bergemann, Dirk, and Ulrich Hege. "The financing of innovation: learning and stopping." RAND Journal of Economics (2005): 719-752.

[4] Awerbuch, Baruch, and Robert D. Kleinberg. "Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches." Proceedings of the thirty-sixth annual ACM symposium on Theory of computing. ACM, 2004.

[5] Kleinberg, Robert, and Tom Leighton. "The value of knowing a demand curve: Bounds on regret for online posted-price auctions." Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on. IEEE, 2003.

[6] Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." Proceedings of the 19th international conference on World wide web. ACM, 2010.

[7] Alaya-Feki, A., E. Moulines, and A. LeCornec. "Dynamic spectrum access with non-stationary multi-armed bandit." Signal Processing Advances in Wireless Communications, 2008. SPAWC 2008. IEEE 9th Workshop on. IEEE, 2008.

[8] Dani, Varsha, Thomas P. Hayes, and Sham M. Kakade. "Stochastic Linear Optimization under Bandit Feedback." COLT. 2008.

[9] Auer, Peter. "Using confidence bounds for exploitation-exploration trade-offs." The Journal of Machine Learning Research 3 (2003): 397-422.

[10] Chu, Wei, et al. "Contextual bandits with linear payoff functions." International Conference on Artificial Intelligence and Statistics. 2011.

[11] Filippi, Sarah, et al. "Parametric bandits: The generalized linear case." Advances in Neural Information Processing Systems. 2010.

[12] Auer, Peter, et al. "The nonstochastic multiarmed bandit problem." SIAM Journal on Computing 32.1 (2002): 48-77.

[13] Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem." Machine learning 47.2-3 (2002): 235-256.

[14] Besbes, Omar, Yonatan Gur, and Assaf Zeevi. "Optimal Exploration-Exploitation in a Multi-Armed-Bandit Problem with Non-stationary Rewards." Available at SSRN 2436629 (2014).

[15] Bubeck, Sébastien, and Nicolo Cesa-Bianchi. "Regret analysis of stochastic and nonstochastic multi-armed bandit problems." arXiv preprint arXiv:1204.5721 (2012).

[16] Granmo, Ole-Christoffer, and Stian Berg. "Solving non-stationary bandit problems by random sampling from Sibling Kalman filters." Trends in Applied Intelligent Systems. Springer Berlin Heidelberg, 2010. 199-208.

[17] Garivier, Aurélien, and Eric Moulines. "On upper-confidence bound policies for non-stationary bandit problems." arXiv preprint arXiv:0805.3415 (2008).

[18] Kun, Jeremy. "Math ∩ Programming." n.p. Web. 16 Nov. 2014.

[19] Lai, Tze Leung, and Herbert Robbins. "Asymptotically efficient adaptive allocation rules." Advances in applied mathematics 6.1 (1985): 4-22.

[20] Langford, John, and Tong Zhang. "The epoch-greedy algorithm for multi-armed bandits with side information." Advances in neural information processing systems. 2008.

[21] Li, Lihong, et al. "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms." Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011.

[22] Robbins, Herbert. "Some aspects of the sequential design of experiments." Herbert Robbins Selected Papers. Springer New York, 1985. 169-177.

[23] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." Proceedings of the 10th international conference on World Wide Web. ACM, 2001.

[24] Slivkins, Aleksandrs, and Eli Upfal. "Adapting to a Changing Environment: the Brownian Restless Bandits." COLT. 2008.

[25] Wang, Ziyu, and Nando de Freitas. "Predictive adaptation of hybrid Monte Carlo with Bayesian parametric bandits." NIPS Deep Learning and Unsupervised Feature Learning Workshop. 2011.

[26] Whittle, Peter. "Multi-armed bandits and the Gittins index." Journal of the Royal Statistical Society. Series B (Methodological) (1980): 143-149.

[27] Syed, Umar, Aleksandrs Slivkins, and Nina Mishra. "Adapting to the shifting intent of search queries." Advances in Neural Information Processing Systems. 2009.

[28] Gur, Yonatan, Assaf Zeevi, and Omar Besbes. "Stochastic Multi-Armed-Bandit Problem with Non-stationary Rewards." Advances in Neural Information Processing Systems. 2014.

[29] Radunovic, Bozidar, et al. "Dynamic channel, rate selection and scheduling for white spaces." Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies. ACM, 2011.

[30] Kenneth S. Miller, "On the Inverse of the Sum of Matrices", Mathematics Magazine Vol. 54, No. 2 (Mar., 1981), pp. 67-72.