From Routine Bandits to Non-Stationary Bandits

Mustapha Ajeghrir
mustapha.ajeghrir@student-cs.fr

Asmae Khald
asmae.khald@student-cs.fr

Nouamane Tazi
nouamane.tazi@student-cs.fr

Abstract

When a recommender system is deployed on multiple users, one does not typically assume that the best recommendation is the same for all users. The naive strategy in this situation is to consider each user as being a different bandit instance and learning from scratch for each user. When users can be recognized (e.g., characterized by features), this information can be leveraged to speed up the learning process by sharing observations across users. The resulting setting is known as contextual bandit [1, 2]. In the routine bandit setting [3], the users cannot be or do not want to be identified (e.g., for privacy reasons), and it is assumed that there exists a (unknown) finite set of possible user profiles (bandit instances), such that information may be shared between the current user and some previously encountered users. More specifically, at each period $h \in [1, H]$, the same bandit b^h is considered during $T > 1$ consecutive time steps, but the identity b^h is unknown to the learner. The KLUCB-RB algorithm [3] solves the routine bandit problem with asymptotically optimal regret minimization guarantees. The goal of this work is to consider a variant of the routine bandit problem where the number T_h of consecutive time steps of each period h is unknown to the learner and to propose adapted versions of KLUCB-RB to tackle this problem, which appears as a variant of the non-stationary multi-armed bandit problem where a learner faces abrupt changes between a small set of bandit instances.

1 Introduction

Online algorithms are widely applicable in real life and over many applications based on fast growing corpus of information. Multi-armed bandit is one such algorithm and is the focus of our study. 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, online 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, which means it will require a training dataset to build a model. In online learning this dataset 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 a very tricky 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) very few algorithms that takes on the non-stationary environment while considering the contextual information (user history etc.).

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.

2 Related work

In literature there are multiple types of explore and exploit strategies that differ in the kind of environments they are dealing with [4]. 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 [5]. 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.

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

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

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 [5]. The following equation gives the formula for UCB1 strategy given by [5] 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,

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

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 [6], 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 [7]. 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 [8] is called Exp3.

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 [4]. Gittens gave a computationally efficient greedy policy for calculating the optimal policy in his seminal paper [9].

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 [4]. 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 [10]. 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 [11] 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 [12]. 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].

2.6 Non-stationary bandits

2.6.1 Abrupt changing environments

In [13] 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.

2.6.2 Brownian restless bandits

Upfal etal in [14] 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.

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

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

[15] 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 [8] 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 $\mathcal{O}\left(T^{2/3}\right)$ 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 [8]. 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.

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

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

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

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 [13]. 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 $O\left(k + d_F\right)\left(\frac{n}{\Delta}\log T\right)$ 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" ($\Delta = \min_t \min_{i \neq i^{*t}} p_t\left(i_t^*\right) - p_t(i)$) 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 Routine Bandits

In this section we turn to the more relevant part of the report. We describe the research relevant to the routine bandits.

2.7.1 Initiation

The stochastic multi-armed bandit , is a popular framework to model a decision-making problem where a learning agent (learner) must repeatedly choose between several real-valued unknown sources of random observations (arms) to sample from in order to maximize the cumulative values (rewards) generated by these choices in expectation. This framework is commonly applied to recommender systems where arms correspond to items (e.g. ads, products) that can be recommended and rewards correspond to the success of the recommendation (e.g. click, buy). An optimal strategy to choose actions would be to always play an arm with highest expected reward. Since the distribution of rewards and in particular their mean are unknown, in practice a learner needs to trade off exploiting arms that have shown good rewards until now with exploring arms to acquire information about the reward distributions. The stochastic multi-armed bandit framework has been well-studied in the literature and optimal algorithms have been proposed . We study a variant of the multi-armed bandit problem in which a learner faces every day one of $\mathcal{B}$ many bandit instances, and call it a routine bandit. More specifically, at each period $h \in 1, H$, the same bandit $b_\star^h$ is considered during $T > 1$ consecutive time steps, but the identity $b_\star^h$ is unknown to the learner. We assume all rewards distribution are Gaussian standard. Such a situation typically occurs in recommender systems when a learner may repeatedly serve the same user whose identity is unknown due to privacy issues.

2.7.2 Routine bandit formulation

A routine bandit problem is specified by a time horizon $T1$ and a finite set of distributions $\nu = (\nu_b)_{b\in\mathcal{B}}$ with means $(\mu_{a,b})_{a\in\mathcal{A},b\in\mathcal{B}}$, where $\mathcal{A}$ is a finite set of arms and $\mathcal{B}$ is a finite set of bandit configurations. Each $b \in \mathcal{B}$ can be seen as a classical multi-armed bandit problem defined by $\nu_b = (\nu_{a,b})_{a\in\mathcal{A}}$. At each period $h1$ and for all time step $t \in 1, T$, the learner deals with a bandit $b_\star^h \in \mathcal{B}$ and chooses an arm $a_t^h \in \mathcal{A}$, based only on the past. The learner then receives and observes a reward $X_t^h \sim \nu_{a_t^h, b_\star^h}$. The goal of the learner is to maximize the expected sum of rewards received over time (up to some unknown number of periods $H1$). The distributions are unknown, which makes the problem non-trivial. The optimal strategy therefore consists in playing repeatedly on each period h, an optimal arm $a_\star^h \in \mathrm{argmax}_{a\in\mathcal{A}}\, \mu_{a,b_\star^h}$, which has mean $\mu_\star^h = \mu_{a_\star^h, b_\star^h}$. The goal of the learner is equivalent to minimizing the cumulative regret with respect to an optimal strategy:

$$R(\nu, H, T) = E_\nu \left[\sum_{h=1}^{H} \sum_{t=1}^{T} \left(\mu_\star^h - X_t^h \right) \right] \tag{2}$$

2.7.3 Related work : latent bandits

A latent bandit is a bandit problem where the learning agent knows reward distributions of arms conditioned on an unknown discrete latent state. The goal of the agent is to identify the latent state, after which it can act optimally. This setting is a natural midpoint between online and offline learning, where complex models can be learned offline and the agent identifies the latent state online. This is of high practical relevance, for instance in recommender systems. In this work [17], man proposes general algorithms for latent bandits, based on both upper confidence bounds and Thompson sampling. The algorithms are contextual, and aware of model uncertainty and misspecification. man provides a unified theoretical analysis of the algorithms, which have lower regret than classic bandit policies when the number of latent states is smaller than actions. A comprehensive empirical study showcases the advantages of that approach.

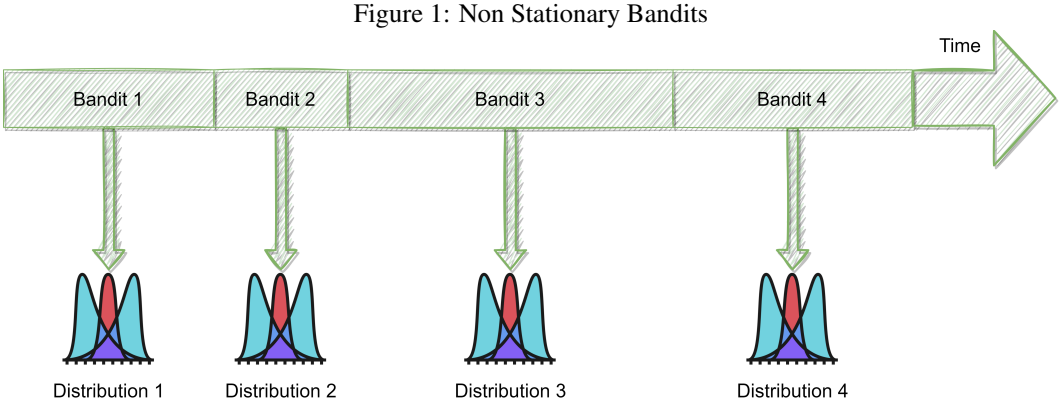
2.7.4 Related work : clustering bandits

Cluster-of-bandit policy leverages contextual bandits in a collaborative filtering manner and aids personalized services in the online recommendation system (RecSys). When facing insufficient observations, the cluster-of-bandit policy could achieve more outstanding performance because of knowledge sharing. Cluster-of-bandit policy aims to maximize the cumulative feedback, e.g., clicks, from users. Nevertheless, in the way of their goal exist two kinds of uncertainties. First, cluster-of-bandit algorithms make recommendations according to their uncertain estimation of user interests. Second, cluster-of-bandit algorithms transfer relevant knowledge upon uncertain and noisy user clusters. Existing algorithms only consider the first one, while leaving the latter one untouched. [18]

3 Problem Description

The Non Stationary bandits problem is similar to the Routine bandits problem described in 2.7.2, except that the Horizon of each bandit is unknown. As we can see in the figure 1, each bandit is free to take a different horizon than others. In addition, bandits don't share the same distributions in contrast with Routine bandits.

Figure 1: Non Stationary Bandits



In this work, we chose to work with Bernoulli distributions, each bandit generates randomly the profiles for its Arms, after reaching its horizon, a new bandit is generated with new distributions.

Our main focus throughout this work is to minimize the pseudo regret by trying out several popular algorithms. We finally propose a new algorithm **KL_NS_UCB** inspired from the **KLUCB-RB** algorithm [3] which outperforms the rest on the task at hand.

4 Implementation

4.1 UCB

The Unstructured Stochastic Bandits, this is the classical implementation with $\delta_t = t^{-2} \times (t+1)^{-1}$, thus the upper bounds becomes:

$$U_a(t) = \hat{\mu}_a(t) + \sqrt{\frac{\log(1/\delta_t)}{2N_a(t)}}\,, \tag{3}$$
$$a_{t+1} = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, U_a(t)\,.$$

with :

N_a : number of times a is drawn
$\hat{\mu}_a$: empirical mean for arm a

4.2 KL_UCB

The Kullback Leibler Unstructured Stochastic Bandits. This is the same as the **UCB** but, it is using the kullback leibler divergence to determine the best arm to sample :

$$U_a(t) = \sup\left\{\mu \in \bar{I} : d\left(\widehat{\mu}_a(t), \mu\right) \leq \frac{\ln(t)}{N_a(t)}\right\} \tag{4}$$

d : is the kullback leibler divergence
$\bar{I}$: μ's interval, for Bernoulli it is equal to $[0, 1]$

This version of **UCB** takes more execution time since it should find the $\sup$ value (with a tolerance of 10^{-4}), but it is known to give better results than the classical **UCB**.

4.3 R_UCB and R_KL_UCB

The Refreshed (Kullback Leibler) Unstructured Stochastic Bandits. This is a simple updated version of **UCB** or **KL_UCB**, it consists of re-initialising the internal parameters of the learner for each $T := period$ steps. For the non stationary bandits problem, the optimal period is not known as it is depending on the horizons of each bandit, so we will study the effect of choosing a different period than the real one.

4.4 SW_UCB and SW_KL_UCB

The Sliding Window (Kullback Leibler) Unstructured Stochastic Bandits. This is based on the implementations of **UCB** or **KL_UCB** with the difference being the existence of a memory buffer. So the learner only remembers what has been seen in the previous `buffer_size` steps. This helps the learners to stop relying on the deep past, which enables the learned distributions to be updated over time. But it is also harmful since it becomes difficult to estimate the distributions with a low `buffer_size`.

4.5 NS_UCB and KL_NS_UCB

The (Kullback Leibler) Non Stationary Upper Confidence Bound. This is our proposed adaptation of the **KLUCB-RB** algorithm [3] seen above. Since we don't try to recognise bandits, we only want to be aware when a new bandit is coming, so we don't need the full potential of the **KLUCB-RB** algorithm. For our implementation, we used a revision buffer to continuously test if the learned distributions in the last `buffer_size` resembles the ones we have seen so far.

The main loop is similar to the **UCB** or **KL_UCB**, the data is additionally saved in a second container of length `buffer_size`.

The revision process is only starting after `buffer_size*2` for each 10 steps. It is based on the following test :

First lets consider : $b := buffer\_size$

The test statistics resembles the one in [3], but some changes have been adopted :

$$Z_a(t) = \left|\widehat{\mu}_a^{buffer}(t) - \widehat{\mu}_a(t-b)\right| - \mathrm{d}\left(N_a^{buffer}(t), \delta\right) - \mathrm{d}\left(N_a(t-b), \delta\right), \tag{5}$$

With :

$$\mathrm{d}(n, \delta) = \sqrt{2\left(1 + \frac{1}{n}\right)\frac{\log(\sqrt{n+1}/\delta)}{n}} \qquad \delta = 1$$

The test is considered positive if :

$$\mathrm{T}(t) := \max_{a \in \mathcal{A}} Z_a(t) \geq 0.$$

When the test is positive, the learner will be reinitialized. This means that the learner detected that it should learn a new distribution for the new arms.

5 Experiments

The configuration of the following experiments is :

```
timeHorizon = 20000
N_exp       = 100
nbArms      = 5
ts          = [5000]
```

We have the total horizon is equal to 20000 steps. For each case, the experiment will be conducted 100 times. The number of arms in our Non Stationary bandits problem is equal to 5. The horizon is equal to 5000 for all bandits (unknown to the learner).

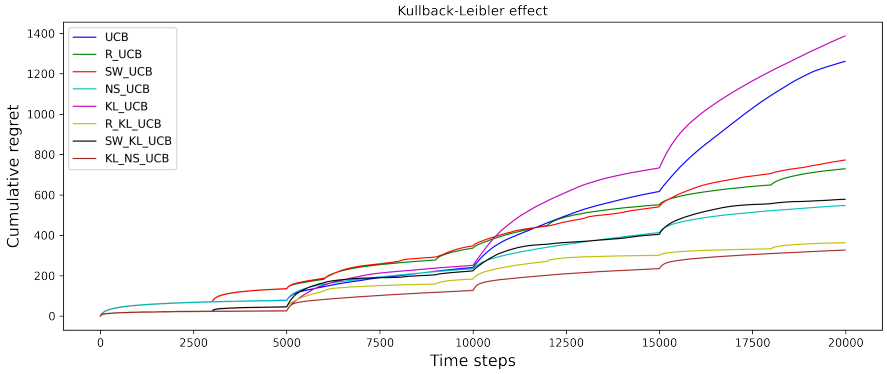
5.1 The effect of Kullback Leibler's divergence

The used configuration for tested algorithms is listed below, the goal of this experiment is to discover the effect of using the Kullback Leibler's divergence

Algorithm	Period	Buffer_size
UCB + KL_UCB	-	-
R_UCB + R_KL_UCB	3000	-
SW_UCB + SW_KL_UCB	-	3000
NS_UCB + KL_NS_UCB	-	200

Table 1: Used configuration for the different algorithms

Figure 2: Effect of the Kullback Leibler's divergence



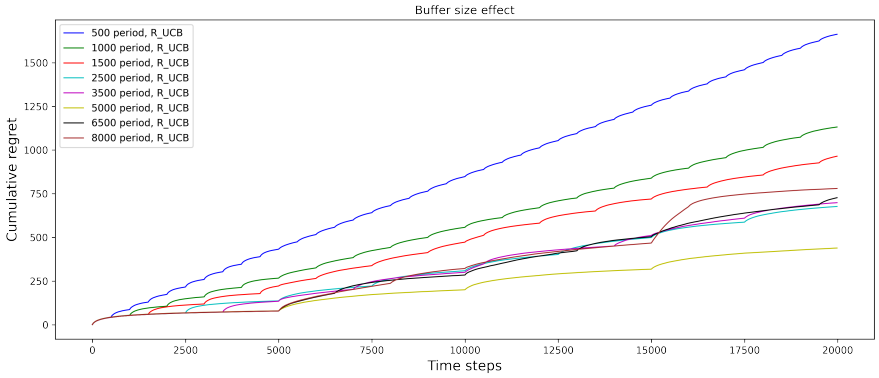
In the figure 2, we clearly see the advantage of using the Kullback Leibler's divergence. For example, it decreased the cumulative regret by 350 for the **R_UCB** algorithm, this is because the KL divergence converges faster towards the best arm.

On the other hand, we see that the KL divergence is not always positive for simple learners like **UCB**. But this stays a marginal observation as the classic **UCB** is not meant to be used directly for non stationary bandits.

5.2 The effect of refresh period for R_UCB algorithm

In this experiment, the period of the **R_UCB** is being studied. In fact, the optimal value is obvious, it should simply be equal to the horizon of each bandit. But this value is unknown, which makes this study interesting.

Figure 3: Effect of the Refresh period for the **R_UCB**



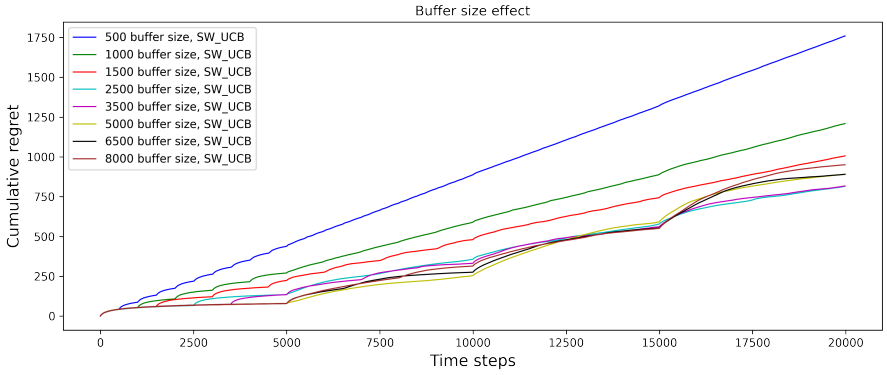
In the figure 3, the optimal period is equal to the bandits horizon, this simply means that the optimal configuration is when the algorithms re-initializes itself after the arrival of a new bandit.

It's also worth noting that it is not very dramatic to get the wrong value of the period. Other than the first 3 values, the regret is bounded by the double of the optimal value. Depending on the application, it might be sufficient to only know the order of magnitude of the non stationary bandits' horizons to be able to use this algorithm. As is it very hardware efficient.

5.3 The effect of Buffer size for SW_UCB

In this experiment, the buffer size of the **SW_UCB** is being studied.

Figure 4: Effect of the Buffer size for the **SW_UCB**



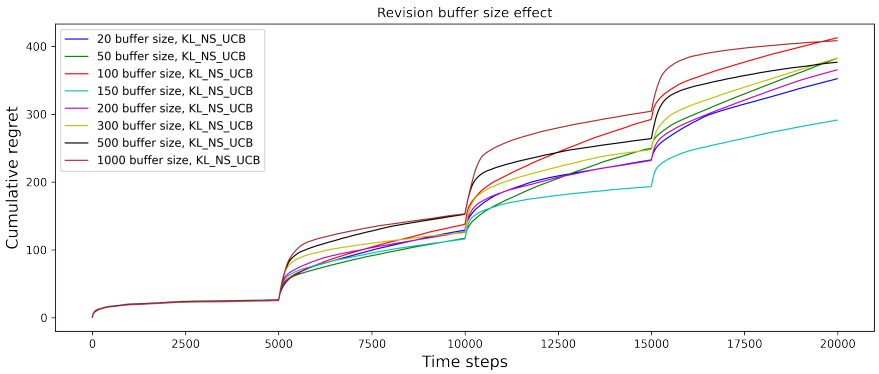
Based on the figure 4, the **SW_UCB** is more resilient than the **R_UCB**, the cumulative regret is less dispersed when changing the buffer size. This resilience is explained by the freedom given by the buffer to the algorithm, as opposed to the re-initialization.

It is also important to notice that the optimal buffer sizes are lower than the horizon. This is explained by the tradeoff between getting higher amount of data-points and the importance of the new points.

5.4 The effect of Revision Buffer size for KL_NS_UCB

This experiment focuses on the most promising algorithm in this list. We study the effect of changing its revision buffer size, as it is explained above, this buffer is only used to test if the arms have changed their distribution but the full history of pulls is kept in memory until a positive test.

Figure 5: Effect of the Revision Buffer size for the **KL_NS_UCB**



In the figure 5, the **KL_NS_UCB** algorithm reveals its promising resilience to the buffer size value. The algorithm kept good results nearly for all the tested values. We can also notice that higher buffer size implies higher regret when the bandit is changed, as the buffer needs more time to be filled with a significant amount of points sampled from the new distribution.

5.5 Global comparison

For this experiment, the following configuration is used :

```
timeHorizon = 30000
N_exp       = 100
nbArms      = 5
ts          = [10000, 5000, 7000, 2000]
```

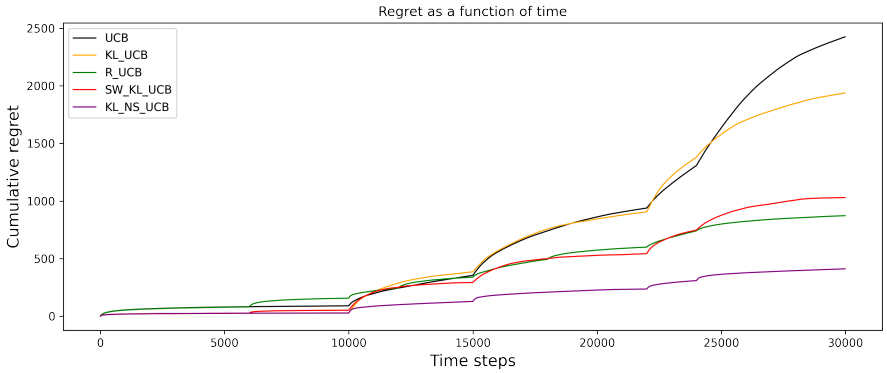
This means that the total horizon is equal to 30000 steps. For each case, the experiment will be done 100 times. The number of arms is equal to 5. The horizon is equal to 10000 for the 1st bandit, then 5000, 7000 and 2000 for the next bandits.

The configuration used for algorithms is as follows :

Algorithm	Period	Buffer_size
UCB	-	-
KL_UCB	-	-
R_KL_UCB	6000	-
SW_KL_UCB	-	6000
KL_NS_UCB	-	150

Table 2: Used configuration for the different algorithms

Figure 6: Global overview for the main algorithms in the non stationary bandits problem



the figure 6 show a practical example of the Non Stationary bandits, classical **UCB** gets very bad results as is it impossible for it to forget the past.

In the other hand **KL_NS_UCB** successfully managed to merge good aggregation of past data points, and also quick responses to any distribution change.

5.6 More depth in KL_NS_UCB

In order to get a better understanding of the **KL_NS_UCB** algorithm, we have studied the effect of the number of bandits over the cumulative regret.

From figure 7, we notice the high value of regret when number of arms is high and the buffer size is low. This is expected because high number of arms requires higher steps to reach good estimations for the distributions.

It is also noticeable that with low number of arms, high buffer size becomes interesting because in those cases, there is a 50% the previous best arm stays the current best arm.

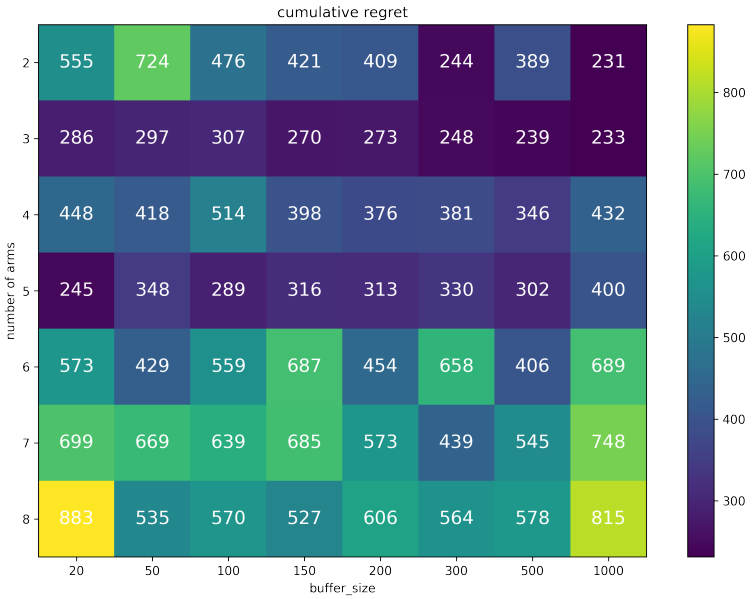
6 Conclusion

We studied a contextual bandit problem for personalized recommendation in a non-stationary environment. We considered a variant of the routine bandit problem where the number T_h of consecutive time steps of each period h is unknown to the learner and to propose adapted versions of KLUCB-RB to tackle this problem, which appears as a variant of the non-stationary multi-armed bandit problem where a learner faces abrupt changes between a small set of bandit instances. Our experiments show that our proposed KL_NS_UCB outperformed its previous peers in catching the quick distribution changes. It also successfully managed to merge good aggregation of past data points, while proving its resilience to the buffer size. Several issues in this work ask for future studies. Further theoretical regret analysis is still lacking, as well as a rigorous comparison to the original KLUCB-RB. Moreover, we can consider applying the algorithm to a wider scope of problems.

Acknowledgments

We would like to thank Hassan Saber for supervising this work.

Figure 7: Effect of the number of bandits over the buffer size



References

[1] Langford, J., Zhang, T.: The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T., Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) NIPS. MIT Press (2007)

[2] Lu, T., Pál, D., Pál, M.: Contextual multi-armed bandits. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the 13th international conference on Artificial Intelligence and Statistics. vol. 9, pp. 485–492 (2010)

[3] `https://hal.archives-ouvertes.fr/hal-03286539`

[4] Bubeck, S´ebastien, and Nicolo Cesa-Bianchi. "Regret analysis of stochastic and nonstochastic multiarmed bandit problems." arXiv preprint arXiv:1204.5721 (2012).

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

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

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

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

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

[10] 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.

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

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

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

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

[15] 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.

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

[17] Hong, Joey and Kveton, Branislav and Zaheer, Manzil and Chow, Yinlam and Ahmed, Amr and Boutilier, Craig : Latent bandits revisited 2020, pages 13423-13433

[18] Liu Yang, Bo Liu, Leyu Lin, Feng Xia, Kai Chen, and Qiang Yang. 2020. Exploring Clustering of Bandits for Online Recommendation System. In Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil. ACM, New York, NY, USA, 10 pages.