A Linear Bandit for Seasonal Environments

Giuseppe Di Benedetto [1 2] **Vito Bellini** [1] **Giovanni Zappella** [1]

Abstract

Contextual bandit algorithms are extremely popular and widely used in recommendation systems to provide online personalized recommendations. A recurrent assumption is the stationarity of the reward function, which is rather unrealistic in most of the real-world applications. In the music recommendation scenario for instance, people's music taste can abruptly change during certain events, such as Halloween or Christmas, and revert to the previous music taste soon after. We would therefore need an algorithm which can promptly react to these changes. Moreover, we would like to leverage already observed rewards collected during different stationary periods which can potentially reoccur, without the need of restarting the learning process from scratch. Here we present a contextual bandit algorithm which detects and adapts to abrupt changes of the reward function and leverages previous estimations whenever the environment falls back to a previously observed state. We provide experiments to show that the proposed method can outperform state-of-the-art algorithms for non-stationary environments.

1. Introduction

Bandit algorithms are extremely popular and widely used in recommender systems, due to their ability to efficiently deal with the exploration-exploitation trade-off in an online fashion. Moreover, contextual bandits are able to leverage context information (e.g., regarding the user or the device) often available in modern applications.A common assumption which is often postulated is the stationarity of the environment. Under this assumption, the aim of the algorithm is to obtain a precise estimate of the parameter that defines the mapping between contextualized actions and rewards. However, most of the times this assumption is not satisfied in real-world scenarios. For example, in music streaming platforms the taste of the users changes significantly over time and intra-day patterns could be observed, with some contents being extremely successful at certain hours of the day but performing significantly worse in other moments. Change points are extremely hard to correctly estimate a-priori and for this reasons it is important to consider them unknown.This is just one example (see for instance (Wu et al., 2019) for more real-world examples), which in practice is often addressed introducing additional information in the context. While these solution are often effective for well-known scenarios, they often fail in real-world recommenders due to the difficulty of keeping track of specific information about the environment. When a recommendation system is serving customers' requests world-wide, it should have available information about specific days (e.g., local holidays which can be different even within the same country), special offers for each location (e.g., ad-hoc marketing campaigns), endogenous events (e.g., marketing campaigns on TV and other media, music festivals) and many other factors. This is just unfeasible in practice, and even if not unfeasible, it would be hard to learn the effect of each of these factors since some of them are quite rare but still extremely valuable from a business prospective.

The ubiquity of non-stationary data has lead, in the recent years, to several works (see (Luo et al., 2018), (Karnin & Anava, 2016), (Chen et al.)) about online learning with bandit feedback under non-stationarity. Though, when dealing with abruptly changing environments, it is reasonable to assume that stationary configurations can reoccur over time. We will refer to this scenario as *seasonal*, even if stationary configurations might nor occur on a periodic basis.

We propose a novel contextual bandit algorithm that detects abrupt changes and leverages the seasonality of the reward function. It deploys a collection of base bandit instances, each one learning about one of the unique reward stationary states. Each bandit gets assigned a weight reflecting how likely recent observations are to come from the stationary periods described by it. A short-term memory bandit is used to detect reward shifts toward stationary periods that had not occurred in the past, and cannot therefore be described by the existing bandits. In this case, a new base bandit instance is initialized. Experiments on real datasets show that our model can outperform state-of-the-art algorithms.

[1]Amazon, Berlin, Germany [2]University of Oxford, Department of Statistics, Oxford, United Kingdom. This work has been done during an internship at Amazon. Correspondence to: Giuseppe Di Benedetto <bgiusep@amazon.com>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

The paper is organized as follows: Section 2 reviews some of the relevant algorithms present in the literature; Section 3 details the setting and Section 4 contains a description of the proposed algorithm. Sections 5 and 6 show experiments on synthetic and real data, comparing our algorithm against several baselines. Section 7 contains concluding remarks.

2. Related work

In the last decade there has been an increasing interest in relaxing the stationarity assumption in the multi-armed bandits algorithms. While this is a step forward compared to the traditional stationary assumption, as already explained in Section 1, more and more complex scenario arises in real-world applications, and seasonal environments is just one of them. The literature considering more complex scenarios is quite scarce in the bandit domain, so in this section we provide an overview of work studied for related settings.

In the non-stationary bandit problem, a key feature of the algorithms is to forget about past observations which could be outdated and therefore potentially harmful for current predictions. In this setting we can distinguish two main families of algorithms: the passive and the active.

Algorithms passively discarding the past do not perform any data-driven decision to choose what to discard and often rely on hyperparameters to decide how aggressively to forget. (Garivier & Moulines, 2011) first proposed two non-contextual mulit-armed bandit algorithms: the Sliding Window Upper Confidence Bound (SW-UCB) and the Discounted Upper Confidence Bound (D-UCB). The former simply updates the reward parameter estimator by considering a fixed number of past observations, while the latter discounts the past observations with exponentially decreasing weights, with the advantage that there is no need to store a sliding window of records to perform the update of the estimator. A recent work (Russac et al., 2019) extends the D-UCB to the linear case. Among the algorithms that passively address the non-stationarity, (Besbes et al., 2014) proposes a schema which leverages the Exp3 algorithm (Auer et al., 2002) with restart after a fixed and arbitrary number of observations, providing theoretical gurantees for this policy. (Allesiardo et al., 2017) instead presents a variation of the *Successive Elimination with Randomized Round-Robin* (Even-Dar et al., 2006), called *Successive Elimination with Randomized Round-Robin and Resets* (SER3), where the estimators of the algorithm get reseted with a fixed probability to capture the best-arm switching taking place in changing environments. The recent work (Zhao et al., 2020) proposes a simple restarted UCB algorithm to achieve near-optimal dynamic regret for non-stationary linear bandit.

Non-stationarity can be alternatively dealt with in an active fashion, by detecting when a change in the reward function occurs and propose a strategy to quickly adapt to the new environment. In the recent years several authors focused on this approach. In the non-contextual bandit setting, (Cao et al., 2019) presents the *Monitored-UCB* algorithm, a UCB instance with a change detector algorithm. Allesiardo et al. (Allesiardo et al., 2017) propose a variation of the Exp3 algorithm with a drift detection test. Mellor and Shapiro (Mellor & Shapiro, 2013) follow a Bayesian approach to tackle abrupt reward changes by linking the Thompson Sampling algorithm (Thompson, 1933) with the Bayesian Online Change-point Detection (BOCPD). These algorithms are all designed for the non-contextual bandit setting, so can hardly be leveraged in a real-world application and extending them to contextual bandit algorithms is often non-trivial.

Within the class of bandit algorithms that actively detect changes in the enviroment, there are a few methods proposing to involve multiple multi-armed bandits in a hierarchical way. One of the first was *Adapt-EvE*, proposed in (Hartland et al., 2006), which adopts a UCB instance together with the Page-Hinkley statistics to test for an abrupt change. A so called Meta-Bandit decides whether to accept the change detection or not, and the parameter of the change detection test is adjusted over time. The method described in (Cheung et al., 2019), called *Bandit over Bandits* (BoB), involves a SW-UCB bandit whose window size is selected by a Exp3 bandit. (Wu et al., 2018) adopt a hierarchical bandit algorithm, called *Dynamic Linear UCB* (dLinUCB), where a *Master Bandit* decides which of the *Slave Bandits* has to play based on the accuracy of the Slave bandits' reward estimations. Moreover, the Master bandit can discard out-of-date Slave bandits and create new ones. The algorithm we propose differs in both the bandit selection strategy and in the pruning scheme for discarding bandit instances. A recent development of this algorithm, *Dynamic Ensemble of Bandit Experts* (DenBand), is proposed by the same authors in (Wu et al., 2019), and addresses context dependent reward changes. DenBand is also suitable for the seasonal bandit scenario and we consider it as our main competitor in the experimental section. An additional mention should be made about the works in (Bousquet & Warmuth, 2002), (Adamskiy et al., 2012), and (Kolter & Maloof, 2007). While the algorithms described in those papers do not fit our setting but work in the prediction with expert advice setting, leveraging the full information feedback, they provides some interesting concepts, such as the mixing of past posteriors ((Bousquet & Warmuth, 2002), (Adamskiy et al., 2012)) and the ensemble method for drifting concept (Kolter & Maloof, 2007), that resemble our use of a collection of base bandits to make predictions (see Section 4 for details).

3. Setting

We present a novel contextual bandit algorithm for non-stationary reward functions with seasonality. In particular, the focus is on settings where the reward function abruptly changes, shifting to a brand new one or reverting to an already observed configuration. We assume that both the number of points in which the reward function changes (later called change-points) and the number of unique reward functions are unknown. More in detail, we assume that at each time t there is a set of possible actions to chose from $\mathcal{A}_t = \{x_1(t), \ldots, x_{n_t}(t)\}$. In a music recommender system for example, an action could correspond to a track to be proposed to the user. Each action is represented by a contextualized action vector $x_i(t) \in \mathbb{R}^d$ which contains information about the context at time t and about the action i, which in the music recommendation example can incorporate information about the user, the device, music genre of the track, etc. The aim of a contextual bandit algorithm is to choose the action which maximizes the expected reward in a sequential fashion. In the "seasonal setting" we consider, the reward function that the bandit tries to learn is not unique but changes over time. Given a sequence of steps from 1 to T, there are a number of so-called stationary periods $S = \{s_0, \ldots, s_C\}$ in which a single reward function is used. Namely we assume that there are C change-points determining stationary periods as a set of consecutive time steps. We assume $C \ll T$ and the change-points to be unknown to the learner, which therefore has to detect the change-points in order to quickly react, learning the parameters of the new reward function or selecting the appropriate one among the ones which it previously learnt. Moreover, we assume a linear structure in the function of the rewards

$$r_t(x_i(t)) = \langle \theta_t^*, x_i(t) \rangle + \epsilon_t \tag{1}$$

where $\theta_t^* \in \mathbb{R}^d$ is the parameter of interest. Hereafter we assume the perturbations to be sampled from independent Gaussian distrubutions $\epsilon_t \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$.
Under the seasonal assumption described above, the true value of the parameter θ^* is not constant over time, but it is selected from the set $\Theta = \{\widetilde{\theta}_1, \ldots, \widetilde{\theta}_k\}$ and each of the stationary periods $s \in S$ is associated with a unique value of the reward parameter $\theta_t^* = \widetilde{\theta}_s \in \Theta$ for all $t \in s$. It is also to be noted that in practice most often $k \ll C$.

4. All-season bandit

The algorithm we propose addresses non-stationary rewards with abrupt and possibly seasonal changes. The seasonality of the reward function can be favourable since predictions about recurrent stationary states can be reused and further updated. This suggests the use of a collection of contextual bandits, each one targeting a stationary state of the reward function. The number of change-points is of course unknown a priori, therefore the algorithm has to detect these changes and recognize whether such a state had already occurred. In such case, the bandit which had learned about that value of the reward parameter needs to be chosen to interact with the environment. Alternatively, a brand new bandit needs to be created. In order to perform the change-point detection, at each step every bandit gets assigned a score indicating how likely the last observation is to come from one of the stationary periods represented by the available bandits. The normalized scores are used to sample the bandit which will play in the current time step (see line 6 in algorithm 1). The collection of bandits used in the algorithm consists of a set of "long-term memory" instances, which are called *base bandits* (denoted by L_i in algorithm 1), and a "short-term memory" one (denoted by S in algorithm 1), which is referred to as *shadow bandit*. Each base bandit has to learn about one of the true values $\theta_s^* \in \Theta$ of the reward parameter. The shadow bandit, which only considers the most recent observations, is responsible for triggering the detection of changes to stationary states which had not been previously observed.
In real-world applications, it is common to receive observations gathered into batches with potentially inhomogeneous sizes. Recommender systems usually perform updates of their models according to a time schedule, and between each consecutive updates the system collects different numbers of feedbacks. Algorithm 1 details the batch version of the proposed algorithm, with the shadow bandit being a sliding-window bandit instance. Three hyperparameters are given as input: λ is a regularization parameter which depends on the bandit prototype used for the base and shadow instances; τ tunes the short-term memory of the shadow bandit[1]; N_{max} is the maximum number of base bandits the learner is willing to maintain. We denote with B the number of batches. For each observation in the current batch, a bandit is sampled from the set of available bandits (line 6 in Algorithm 1) with probabilities proportional to the weights ω_j, and interacts with the environment. If the shadow bandit has been selected at least once in the batch, then a new long-term memory instance is initialized with the parameters of the shadow bandit. In the update step, the shadow bandit gets updated with the last τ observation, regardless of which bandit has played, while the base instances are updated on the set of observations they have been assigned in the batch. A maximum number N_{max} of base bandits can be fixed a priori. If the number of base bandits exceeds this constraint, a pruning scheme is called to discard one of them. Before starting a new batch, the weights of each bandit are computed.
Reward shift detection. The changes in the reward function are detected by looking at the weights assigned to each bandit, which depend on the last observed reward —or the rewards observed in the last batch— and on the reward parameter estimate provided by the bandit. When the reward function shifts to an already observed state, the base bandit which had learned about that stationary configuration

[1]The parameter τ could be the window size or the discount factor if the shadow bandit is a sliding-window or discounted bandit instance respectively

should get assigned a high weight. Instead, when the reward parameter shifts to a new state which had not been previously observed, none of the base bandits would be eligible to choose the action to play. In this case a brand new bandit should be initialized. The creation of a new bandit is triggered by the shadow bandit. Being it a short-term memory instance, it does not take into account the whole history but, in the case of the sliding window base algorith, a fixed number τ of past observations (other base algorithms will adopt a different way to "forget" the past). Moreover, the shadow bandit gets updated at each time step, regardless of which bandit is playing (see line 16 of Algorithm 1), allowing it to track changes in the environment. When the shadow bandit is selected to play, the new base instance is initialized as a copy of the shadow bandit (see line 13 of algorithm 1). diction are constantly based on the last τ observations.

Algorithm 1 All-Season bandit (batch update)

```
1:  Inputs:
        $\lambda,\ \tau,\ N_{\max}$
2:  Initialize:
        Create base and shadow bandits: $L_1, S = \text{INIT}(\emptyset)$
        Set of base bandits: $\mathcal{B} = \{L_1\},\ N_L = 1$
        Initialize bandits' weights: $\omega_S = 0.5,\ \omega_1 = 0.5$
3:  for b = 1 to B do
4:      Set $I_j = \emptyset$ for all $j \in \mathcal{B} \cup \{S\}$
5:      for t in b do
6:          Bandit selection: $j \sim \text{Discrete}(\omega_j : j \in \mathcal{B} \cup \{S\})$
7:          $I_j = I_j \cup \{t\}$
8:          $x_t = \text{PlayBandit}(j, \mathcal{A}_t)$
9:          Observe reward $r_t$
10:     end for
11:     if $I_S \neq \emptyset$ then
12:         $N_L = N_L + 1$
13:         Create bandit $L_{N_L} = \text{INIT}(S),\ I_{L_{N_L}} = \emptyset$
14:         $\mathcal{B} = \mathcal{B} \cup \{L_{N_L}\}$
15:     end if
16:     $S = \text{Init}(\emptyset)$ and Update($S$, $(x_s, r_s)_{s=\min(t-\tau+1,0):t}$)
17:     for $j \in \mathcal{B}$ do
18:         Update base bandit $j$: Update($L_j$, $(x_s, r_s)_{s \in I_j}$)
19:     end for
20:     if $N_L > N_{\max}$ then
21:         Prune($\mathcal{B}$) (see alg. 4)
22:     end if
23:     $(\omega_j)_{j \in \mathcal{B} \cup \{S\}} = \text{UpdateWeights}(\mathcal{B} \cup \{S\}, (x_s, r_s)_{s \in b})$
24: end for
```

Shadow bandit and base bandits. In algorithm 1 the choice of the bandit instances and of the bandits' weighting strategy are not specified. The weights assigned to each bandit should reflect the likelihood of describing the last observation. Any contextual bandit algorithm which can provide such scores can be used as prototype instance for base and shadow bandits in the proposed scheme. In the Bayesian framework, such weights are naturally provided by the posterior predictive probabilities of the observation under each bandit. Therefore in what follows, all the bandits instances involved in the algorithm are linear Thompson Sampling (linTS) bandits. Algorithm 2 recalls the procedures involved in the linear Thompson Sampling. A conjugate model is adopted in order to have analytical form of the posterior distribution of the reward parameter. In particular we place a Gaussian prior on the parameter of interest and assume the reward likelihood to be Gaussian:

$$\theta \sim \mathcal{N}\left(0_d,\, \lambda^{-1}\mathbb{1}_d\right) \qquad r_t \,|\, \theta, x \sim \mathcal{N}\left(\langle\theta, x\rangle,\, \sigma^2\right) \tag{2}$$

Denoting by $D_t = \{(x_s, r_s)\}_{s=1:t}$ the history of chosen actions x_s and corresponding observed rewards r_s up to time t, the posterior distribution of the reward parameter, given D_t, is a Gaussian distribution with precision matrix $M = \lambda\mathbb{1}_d + \sum_{s=1}^{t} x_s x_s^T$ and mean vector $\mu = M^{-1}\sum_{s=1}^{t} r_s x_s$. The posterior predictive distribution of the observed reward at time t given the previous observations is Gaussian distributed as well

$$r_t \,|\, x_t, D_{t-1} \sim \mathcal{N}\left(\langle\mu, x\rangle,\, \sigma^2 + x_t\, M^{-1}\, x_t^T\right) \tag{3}$$

with μ and M being the mean vector and precision matrix of the posterior distribution of $\theta \,|\, D_{t-1}$ respectively.
Every base linTS bandit gets updated only when it plays, therefore its posterior distribution takes into account only the observations that have been assigned to it. Let $\mathcal{I}$ be the set of time indices one of the base bandits has played overall. The posterior distribution of the reward parameter provided by that base bandit is a Gaussian distribution with precision matrix $M = \lambda\mathbb{1}_d + \sum_{s\in\mathcal{I}} x_s x_s^T$ and mean vector $\mu = M^{-1}\sum_{s\in\mathcal{I}} r_s x_s$.
The changes in the reward function are detected by looking at the posterior predictive probabilities of the last observed reward. In the case the rewards are observed in batches, the scores assigned to each bandit take into account all the records in the last batch (see algorithm 3). Denoting by $p_j(r_t \,|\, x_t)$ the posterior predictive probability that the bandit j gives to the reward r_t, given the action x_t, we have that the weight assigned to the bandit after collecting a batch b of observations is

$$\omega_j = \prod_{t\in b} p_j(r_t \,|\, x_t)$$

After computing these scores for each bandit, the algorithm samples the reward parameter estimate $\hat{\theta}$ from the mixture of posterior distributions provided by the available bandits, with weights being the predictive posterior probabilities defined above (line 6 of Algorithm 1).

Pruning schemes. Allowing the algorithm to create an unbounded number of base bandits could lead to computational issues, both in terms of memory and time. As a matter of fact, outliers might trigger false change-point detections which would imply the creation of additional long-term memory bandits targeting a stationary configurations which had already been learnt by another long-term

Algorithm 2 Linear Thompson Sampling

```
1:  procedure INIT(L)
2:      if L = ∅ then
3:          return $M = \lambda \mathbb{1}_d,\ b = 0_d,\ \mu = 0_d$
4:      else
5:          return $M = M_L,\ b = b_L,\ \mu = \mu_L$
6:      end if
7:  end procedure
8:  ________________________________
9:  procedure PLAYBANDIT(L, $\mathcal{A}_t$)
10:     Sample $\hat{\theta}_j \sim \mathcal{N}(\mu_L,\ M_L^{-1})$
11:     return $x_t = \arg\max_{x \in \mathcal{A}_t} \langle \hat{\theta}_j, x \rangle$
12: end procedure
13: ________________________________
14: procedure UPDATE(L, $(x_s, r_s)_{s \in I}$)
15:     $M_L = M_L + \sum_{i \in I} x_i x_i^T$,
16:      $b_L = b_L + \sum_{i \in I} x_i r_i,\ \mu_L = M_L^{-1} b_L$
17: end procedure
```

Algorithm 3 Bandit's weights update

```
1: procedure UPDATEWEIGHTS($\mathcal{I}$, $(x_s, r_s)_{s \in b}$)
2:     $\omega_j = \prod_{t \in b} p_j(r_t \mid x_t)$ for all $j \in \mathcal{I}$
3:     return $(\omega_j)_{j \in \mathcal{I}}$
4: end procedure
```

memory bandit previously created. The problem of having spurious base bandits can be addressed by imposing the maximum number of long-term memory instances we are willing to maintain. This number can be chosen using some prior information about the number of unique stationary configurations $|\Theta|$. However, in the case such knowledge is missing, it is recommended to be conservative about this constraint. Having multiple base bandits learning the same stationary period would result in a slow convergence of their estimators since they get updated less frequently. In order to satisfy the constraint, the algorithm needs to have a strategy to prune the least useful base bandit whenever the maximum allowed number is exceeded. We propose two different pruning schemes to control the number of available bandits at each time step (see Algorithm 4). The rationale behind both schemes is to find the pair of closest bandits, and among the two, discard the one which is less certain about its estimate, measured with the trace of the associated posterior covariance matrix. A quick way to find the pair of closest bandits would be to compare the pairwise distances between the posterior mean parameters of the base bandits, resulting in a time complexity of $\mathcal{O}(N_{\max}^2 d)$. However, this would not take into account the whole distribution associated to each bandit. Therefore an alternative approach, used in the experiments presented below, consists in comparing the symmetric Kullback-Leibler divergence between pairs of base bandits. The symmetric Kullback-Leibler divergence between two measures p and q is defined as $\mathrm{KL}_{\text{sym}}(p, q) = \mathrm{KL}(p, q) + \mathrm{KL}(q, p)$, and it can be computed in closed form if the distributions of interest are Gaussian:

$$\mathrm{KL}_{\text{sym}}(\mathcal{N}(\mu_1,\ \Sigma_1),\ \mathcal{N}(\mu_1,\ \Sigma_2)) = \frac{1}{2}\mathrm{tr}(\Sigma_2^{-1}\Sigma_1 + \Sigma_1^{-1}\Sigma_2) \tag{4}$$

$$+ \frac{1}{2}(\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) + (\mu_2 - \mu_1)^T \Sigma_1^{-1}(\mu_2 - \mu_1)$$

The time complexity of this scheme is $\mathcal{O}(N_{\max}^2 d^3)$.

Algorithm 4 KL pruning

```
    Inputs:
        Long-memory bandits $j \in \mathcal{I}$
    $(i^*, j^*) = \underset{(i,j) \in \mathcal{I}^2, i<j}{\arg\min} \mathrm{KL}_{\text{sym}}(\mathcal{N}(\mu_j,\ M_j^{-1}),\ \mathcal{N}(\mu_j,\ M_j^{-1}))$
3:  $i_{\text{delete}} = \underset{k=i^*, j^*}{\arg\min} \mathrm{tr}(M_k^{-1})$
    $\mathcal{I} = \mathcal{I} \setminus \{L_{i_{\text{delete}}}\}$ and $N_L = N_L - 1$
```

5. Synthetic data experiment

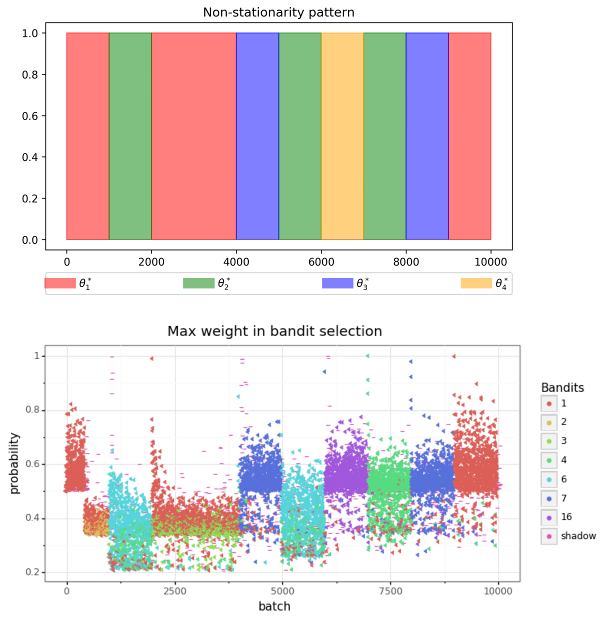

Figure 1. Top figure: Non-stationary pattern of the reward function. Bottom figure: weight of the bandit with highest posterior predictive weight, base bandits depicted with coloured triangles and shadow bandit with pink dashes.

The first experiment is on synthetic data with abruptly changing and seasonal rewards. The aim is to show empirically how the proposed model functions. The experiment consists of $n = 10000$ observations. At each time, the action set consists of five unit-norm vectors in $\mathbb{R}^5$ obtained by randomly sampling from a centred Gaussian distribution with independent coordinates and normalizing. The rewards are sampled from a Gaussian distribution as described in eq 1 with $\sigma^2 = 0.1$ and reward parameter $\theta^* \in \mathbb{R}^5$ changing over time. We consider four different values for the reward parameter, representing the unique stationary periods, alternating over time according to the pattern depicted in Figure 1, which contains eight change-points. In this example each batch corresponds to one observation, and the window-size of the Shadow bandit is set to $\tau = 10$. Under correct specification of the model, the proposed algorithm manages to recover the stationary periods. The second plot in Figure 1 shows the posterior predictive weights of the most likely bandit to be chosen at each time step. The base bandits indexed by 1, 2, 3 learn about the stationary period represented by θ_1^* (see first plot in Figure 1); θ_2^* is learned by base bandits 4 and 6; while θ_3^* and θ_4^* by bandits 7 and 16 respectively. It is worth noting that the shadow bandit has probability of being chosen very close to 1 only when there is a change-point followed by a stationary period which had not been previously observed, namely in the first, third and fifth change-points. In contrast, when an alredy observed stationary period reoccurs, the long-term memory bandit which had been learning about it gets assigned a probability very close to 1, meaning that the reward seasonality is correctly captured. Even if the shadow bandit is assigned the highest weight very often, the constraint on the maximum allowed number of long-term memory bandits ensures that the spurious long-term memory bandits, that were created when not necessary, are discarded by the pruning scheme.

6. Experiments with real data

In this section we present a collection of experiments to compare the All-Season bandit against non-stationary baselines (SW-LinTS, D-LinTS, BoB (Cheung et al., 2019), DenBand (Wu et al., 2019)), showing that our model can provide better performance on abruptly changing environments with seasonality. Even though the best way to compare bandit algorithms would be to test them online with A/B tests, online experiments in recommedation systems are usually expensive and risky since they can negatively affect the customer experience. Moreover, they are impossible to reproduce and unaccessible to academic community. Methods to perform offline evaluation have been designed in order to compare different algortihms on randomized logged data (e.g., (Li et al., 2011), (Li et al., 2010)). The authors of (Dudík et al., 2012) propose an evaluation scheme for non-stationary policies, although the assumption of iid contexts is violated in our case. Moreover, we are considering the case where both the logging policy and the environment are non-stationary, and up to our knowledge, there is not any unbiased offline evaluation proceduce already known for this setting. In order to offer a fair comparison, we have artificially induced non-stationarity to classification problems using two real datasets, trying to mimic plausible non-stationary patterns that could occur in real-world applications.

Change-points patterns. In the following experiments we compare non-stationary bandit algorithms on datasets with three different types of reward seasonality. In all these settings the reward function is piece-wise constant, therefore the changes are abrupt, and the number of unique stationary configurations is the same across them. However, the number of change-points differs, and therefore the length of each stationary period (see Figure 2).

The first scenario is characterized by long stationary periods and few change-points. This setting should favour the algorithms which passively address non-stationarity, such as those which discount past observations or base their prediction on a sliding window of records. Although such algorithms are not suited for abrupt changes, the long duration of a stationary period allows them to learn about the reward parameter before encountering the following change-point; The second scenario contains a mix of short and long

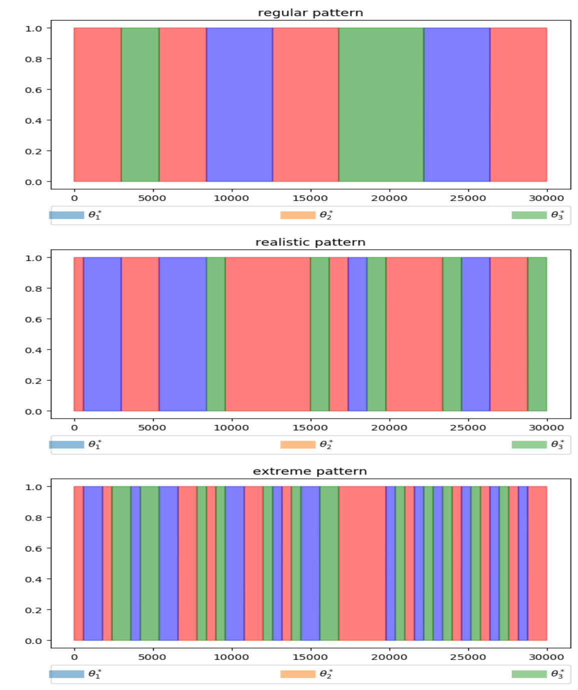

Figure 2. Non-stationary patterns

stationary periods. The aim is to mimic abrupt-changes which might occur in real-world scenarios. In the music recommendation paradigm, very short stationary periods could represent the occurrence of short events (Halloween or Father's Day for instance) which abruptly change the music taste of the users for a fairly short time. Longer stationary periods could correspond for instance to Summer or Christmas, when users might be more willing to listen to Summer hits and Christmas songs respectively. Very long periods instead could correspond to the usual music taste of the user, without being affected by external events. The third scenario contains several very short stationary periods. This scenario can occur when looking at a finer time grid, describing for example variation in the music listened during daytime and nighttime. For the SW-linTS the learner should pick a window wider than the learning time of the LinTS, since this would ensure a good performance in the stationary setting. However, it might be the case that the stationary periods are much shorter than the convergence time. In such case, the bandits that passively tackle the non-stationarity, e.g. SW-linTS and D-linTS, would not be able to quickly react and converge to such short stationarity periods.

Datasets. Two real datasets are considered in the following experiments: MNIST and Fashion-MNIST. Both datasets are used for benchmarking algorithms on image classification. Each observation consists of a vector in $\mathbb{R}^{748}$ with entries from 0 to 255 representing the pixel values of a grey-scale image of a digit (MNIST) or a fashion item (Fashion-MNIST). PCA is used to reduce the dimensionality of the context vectors down to 44 principal components for MNIST and 43 for Fashion-MNIST, describing 80% and 85% of the variance respectively. Across all the experiments we run all the algorithms on the first 30000 observations gathered into batches of size 10. Each stationary period lasts at least for 600 observations and the convergence time of the linTS on the MNIST digit recognition task is of about 2000 observations The contextualized action vectors are given by the outer product between the one-hot encoded action vector and the context vector containing the image features. Two different experiments are presented for each dataset.

Experimental settings.
Two Arms Experiment. The first setting consists of a two-arm problem. In the MNIST dataset, three different tasks are considered: parity, divisibility by three, and primality of the digit. In particular the the two arms (arm 1, arm 2) represent the actions (even, odd), (divisible by three, non divisible by three), (non prime, prime) for each task respectively. Reward 1 is given for correct classification, and 0 otherwise. Analogously for Fashion-MNIST, the arms describe the following classification tasks: (upper-body, lower-body), (winter clothes, summer clothes), (shoes, clothes).
Arm-shift Experiment. In this setting the standard classification of the digits and fashion items is performed. The non-stationarity is induced by shuffling the labels of the arms, therefore each stationary period is represented by a permutation of the labels.

6.1. Baselines and hyperparameters' setting

Here we list the baselines used in the experiments, with the relative hyperparameters' setting. Hyperparameter optimization is performed by grid search using the first 10% of the data as validation set (the initial part of the dataset). The regularization hyperparameter is present in all the baselines, hence it has been set to $\lambda = 1$ in all the experiments.
SW-linTS: Sliding-window LinTS with window size τ optimized in the set $\{50,\ 100,\ 500,\ 1000,\ 5000\}$.
D-linTS: Discounted LinTS, the discount parameter $\gamma = 1 - 10^{-\kappa}$ with $\kappa \in \{1, 3, 5, 10\}$.
BoB: *Bandit Over Bandit* algorithm (Cheung et al., 2019) The time horizon is provided a priori, since it is needed to define the set of allowed window-sizes according to equation (23) in (Cheung et al., 2019). We optimize for the hyperparameters denoted by S and L in (Cheung et al., 2019) in the set $\{0.1,\ 1,\ 10\}$.
DenBand: *Dynamic Ensemble of Bandit* (Wu et al., 2019) can be considered as the main competitor of our algorithm, since it uses a collection of bandits to address non-stationarity and in particular context-dependent reward changes. A sliding window size τ described in the paper is optimized in the set $\{500,\ 1000,\ 2000\}$, while the hyperparameter Δ_L in the set $\{0.1,\ 0.25,\ 0.5\}$.
LinTS: Linear Thompson Sampling, for which no hyperparameter optimization is needed, this baseline is included to have a comparison against a bandit algorithm which does not take into account non-stationarity.
Random: picks an action uniformly at random.

All-Season bandit has two hyperparameters: the maximum allowed number N_{max} of base bandits and the window-size τ of the shadow bandit. They are optimized over the sets $N_{\text{max}} \in \{3,\ 4,\ 5\}$ and $\tau \in \{50,\ 100,\ 500,\ 1000,\ 5000\}$.

6.2. Results

Two arms experiment. In the first experiment with only two arms the reward function partially changes between two distinct stationary periods. For instance, in the MNIST dataset, when the task switches from classifying the parity to classifying the divisibility by three, the best arm does not change for the context vector representing the digit six. In both datasets All-Season bandit (both versions) are almost always better than the base algorithms employed, with a single dataset in which there is a substantial parity. Moreover, we see that All-Season (Disc) is mostly outperforming the All-Season (SW). In addition, both the SW-linTS and D-linTS outperform the other more complex baselines BoB and DenBand on both datasets.

Results on arm shifts experiment. The second experiment is a ten-arm bandit problem, and in this case the reward parameter changes completely between two distinct stationary periods. The All-Season bandits constantly ouperform all the competitors, in some case with large margin. Generally, SW-linTS is the best baseline followed by the D-linTS, while the remaining baselines are often underperforming by large margin.

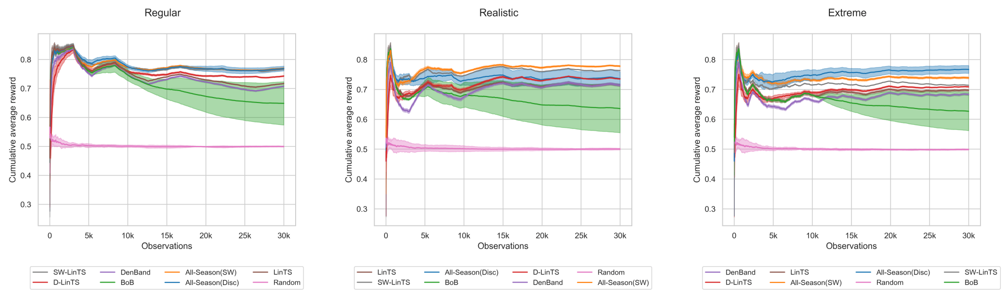

Figure 3. Mnist two-arm: Regular, Realistic and Extreme settings

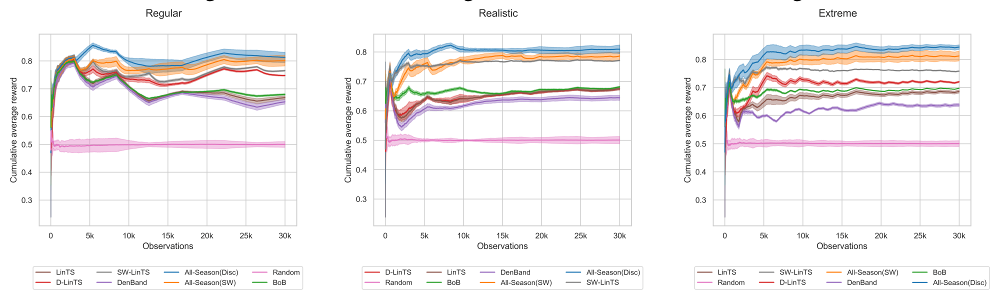

Figure 4. Fashion-Mnist two-arm: Regular, Realistic and Extreme settings

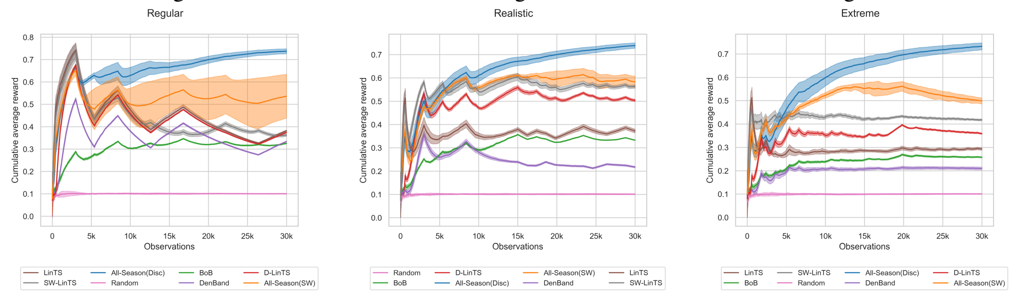

Figure 5. Mnist digit-recognition: Regular, Realistic and Extreme settings

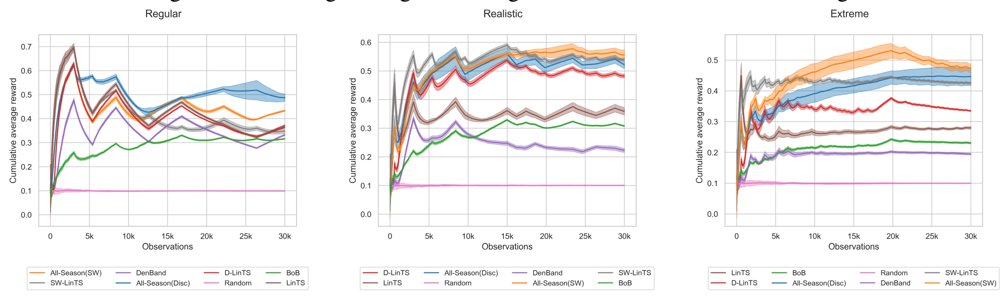

Figure 6. Fashion-Mnist item-recognition: Regular, Realistic and Extreme settings

7. Conclusion and future work

We proposed a new bandit algorithm which addresses the problem of learning in non-stationary environments. The algorithm proved itself able to adapt to different non-stationarity patterns and outperform state-of-the-art solutions by large margin. Moreover, All-Season is a significantly simpler and more robust algorithm than its competitors, suitable to be maintained in industrial applications with little effort.

We conjecture that there is still room for improvement in dealing with the model misspecification problem and in selecting the models to be pruned. In particular, we believe that the General Bayes approach (Knoblauch et al., 2019) could provide a more robust solution for defining the posterior predictive weights.

References

Adamskiy, D., Warmuth, M. K., and Koolen, W. M. Putting bayes to sleep. In *Advances in neural information processing systems*, pp. 135–143, 2012.

Allesiardo, R., Féraud, R., and Maillard, O.-A. The non-stationary stochastic multi-armed bandit problem. *International Journal of Data Science and Analytics*, 2017.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Besbes, O., Gur, Y., and Zeevi, A. Stochastic multi-armed-bandit problem with non-stationary rewards. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 199–207. 2014.

Bousquet, O. and Warmuth, M. K. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3(Nov):363–396, 2002.

Cao, Y., Wen, Z., Kveton, B., and Xie, Y. Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit. In *AISTATS*, 2019.

Chen, Y., Lee, C.-W., Luo, H., and Wei, C.-Y. A new algorithm for non-stationary contextual bandits: Efficient, optimal and parameter-free. In *Conference on Learning Theory, COLT 2019*. PMLR.

Cheung, W. C., Simchi-Levi, D., and Zhu, R. Hedging the drift: Learning to optimize under non-stationarity. *CoRR*, abs/1903.01461, 2019. URL http://arxiv.org/abs/1903.01461.

Dudík, M., Erhan, D., Langford, J., and Li, L. Sample-efficient nonstationary policy evaluation for contextual bandits. *arXiv preprint arXiv:1210.4862*, 2012.

Even-Dar, E., Mannor, S., and Mansour, Y. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(Jun):1079–1105, 2006.

Garivier, A. and Moulines, E. On upper-confidence bound policies for switching bandit problems. In *ALT*, 2011.

Hartland, C., Gelly, S., Baskiotis, N., Teytaud, O., and Sebag, M. Multi-armed bandit, dynamic environments and meta-bandits. 2006.

Karnin, Z. S. and Anava, O. Multi-armed bandits: Competing with optimal sequences. In *Advances in Neural Information Processing Systems*, pp. 199–207, 2016.

Knoblauch, J., Jewson, J., and Damoulas, T. Generalized variational inference. *arXiv preprint arXiv:1904.02063*, 2019.

Kolter, J. Z. and Maloof, M. A. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8(Dec):2755–2790, 2007.

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

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

Luo, H., Wei, C., Agarwal, A., and Langford, J. Efficient contextual bandits in non-stationary worlds. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*. PMLR, 2018.

Mellor, J. and Shapiro, J. Thompson sampling in switching environments with bayesian online change detection. In *Artificial Intelligence and Statistics*, pp. 442–450, 2013.

Russac, Y., Vernade, C., and Cappé, O. Weighted linear bandits for non-stationary environments. In *Advances in Neural Information Processing Systems*, pp. 12040–12049, 2019.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Wu, Q., Iyer, N., and Wang, H. Learning contextual bandits in a non-stationary environment. In *The 41st International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, SIGIR '18, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5657-2.

Wu, Q., Wang, H., Li, Y., and Wang, H. Dynamic ensemble of contextual bandits to satisfy users' changing interests. In *The World Wide Web Conference*, WWW '19, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6674-8.

Zhao, P., Zhang, L., Jiang, Y., and Zhou, Z.-H. A simple approach for non-stationary linear bandits. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 2020, 2020.