Real-world problems for Bandit in recommender system

Garcin Camille & **Lahlali Kenza**
Ecole Centrale Paris
(camille.garcin, kenza.lahlali)@student.ecp.fr

Abstract

This paper presents an overview of the applications of Multi Armed Bandit in Recommender Systems underlying the different problems in real world applications. It describes various limitations of current recommendation methods and possible extensions that can improve recommendation capabilities such as the improvement of understanding of the dynamic market place, the delay conversions or the non transparent environment.

1 Introduction

A recommender system is a system that tries to predict the rating a user would give to a certain item. Typical use cases include movie recommendation and advertisement: showing the user a well-targeted add such as to maximize the generated reward (quantified by the number of clicks for instance). Historical approaches to recommender system are Content-Based approach and Collaborative Filtering.

In the Content-based approach, the idea is to create features for items and to recommend to the user items with similar features to the ones he had previously liked. One of the problems that appears is that we will always propose the same kind of content to the user (this problem is known as the Overspecialization). Also, if the user has not rated any item yet, we have no way of recommending any item to the user (this problem is known as the Cold Start problem).

In the Collaborative setting (CF), we do not create features for items but rather learn from the past interactions of the user with the items. Typically, we have a matrix of ratings where rows represent users and columns represent items. Usually that matrix is very sparse and we would like to fill it up. One way of seeing CF is that it will find similar users and use their ratings of an item to predict how well you might like that item. This is called the "user-user approach". Similarly there is the item-item approach : to predict how a user might like an item, look at how he rated similar items.

However, these two approaches present important shortcomings. First, as mentioned above, these methods don't cope well with the Cold Start problem since they are based on previous ratings of the user. Secondly, these methods don't adapt to changes of the user's behavior: over time, a user might change his preferences or like to be presented with new content that could potentially spark his interest. In the settings described above, the user will only be presented with content similar to what he had already liked. This is an Exploration-Exploitation problem: we would like to take advantage of what we know about the user, while at the same time presenting a new content which could, in the long term, yield higher economic returns. Multi Armed Bandit approach tackle well this trade-off, this is why it is largely used in Recommendation Systems. The arms are the items, and at each time step, we want to pull the arm with the highest expected reward, (ie. the news article he is most likely to read, or the movie he is most likely to watch, or the advertisement for which he is more likely to spend his money). The different Multi Armed Bandit algorithms used in Recommendation systems

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

will be further developed in Part 1 of this report, highlighting for each one how they managed the Exploration/Exploitation problem.

Nonetheless, there are several problems that need to be tackled when using Multi Armed Bandit as an approach to build a real-world Recommender System. First, Multi Armed Bandits have some optimistic assumptions that don't apply to real world. Besides, the growing concern over Privacy jeopardizes Bandit expansion. These problems (and others) are developed in Part 2 as well as some attempts to modify Multi Armed Bandit in order to tackle these real-world problems. Open questions are also developed in this part.

2 Recommender systems with Bandit

The big amount of data of users and items needs an efficient way of recommendation that will constantly interact with the system and get improved rapidly by the received feedback. We need to recommend in the most efficient and rapid way. Multi Armed Bandit give great results because they constantly learn from the user and the environment. Let us present how they work.

The Multi armed Bandit Protocol is as follow:

- The learner has K arms (actions)
- At each step: the environment chooses a vector of rewards $[X_{(i,t)}]_{i=1}^{K}$ (each arm is associated to a probability distribution ν_i) and the learner chooses an arm I_t
- The learner receives a reward $X_{I_t,t}$
- The environment doesn't reveal the reward of the other arms.

We define the regret as: $max_{i=1...K}E[\sum_{t=1}^{n} X_{i,t}] - E[\sum_{t=1}^{n} X_{I_t,t}]$

Imagine a player in a room with different slot machines, with no prior knowledge of the payoffs of any machines. His goal is maximizing his total reward from the slot machines.

Exploration-Exploitation Challenge: The learner doesn't know all the rewards of the other slot machines (arms). By trying more slot machines, he will gain more information. Nonetheless, the more the learner tries "bad slot machines", the higher the regret. Therefore, the learner needs to solve the exploration-exploitation dilemma. In recommendation systems, the Bandit algorithm needs to maximize user satisfaction in the long run alongside gathering information about goodness of match between user interests and content. In the recommendation of news, the learner has to explore news by choosing suboptimal news to gather more information about them. Even though exploitation can decrease short-term user's satisfaction (as some suboptimal news may be chosen) it enables getting information about the news' average rewards and therefore refining the user's estimate of the news' rewards. Consequently, it increases long-term user's satisfaction (as explained in [3]).

Bandit algorithms tackle this dilemma. They can be classified into Context-free or Contextual bandit algorithms.

Context-free algorithms: ϵ-greedy and UCB-algorithms are examples of context-free algorithms. In the ϵ-greedy algorithm, at each trial t, the algorithm estimates the average payoff $\mu_{t,a}$ of each arm a. Then, with probability $1-\epsilon$ it chooses the arm with highest payoff estimate, and with probability ϵ it chooses a random arm. The parameter ϵ decreases with the number of trials (if it was fixed, the average regret incurred would grow linearly). By increasing this parameter, it adds randomness the algorithm will explore other options more frequently. Therefore, when the number of trials goes to infinity, each arm will be tried infinitely often and $\mu_{t,a}$ will converge to its true value with a probability 1. Besides, with ϵ decaying at a certain speed, the regret per step will converge to 0 with probability 1.
One drawback of this algorithm is the difficulty to decide in advance the optimal value of parameter ϵ (this point will be developed in Part 2 with Dynamic content). Besides, we may need to dynamically adjust the Exploration/Exploitation trade off at different stages of the experiment ([4]). A smarter way to handle exploration and exploitation is to use UCB-type algorithms (according to [1]) which estimate the mean payoff $\mu_{t,a}$ of each arm a and the confidence interval $C_{t,a}$ such that we have with a high probability $|\mu_{t,a} - \mu_a| < C_{t,a}$. The $argmax_a(\mu_{t,a} + C_{t,a})$ is the chosen arm. The regret is logarithmic in the total number of trials T, which turns out to be optimal.

However Context-free algorithms present an important shortcoming when applied to recommender system : no information about users is used and thus we can't have personalized recommendations. Hence the need for contextual bandits.

Contextual bandit algorithm: In Contextual bandit, the learner uses context vectors along side rewards of the arms played in the past to make the choice of the arm to play in the current iteration. Over time, the learner's aim is to collect enough information about how the context vectors and rewards relate to each other, so that it can predict the next best arm to play by looking at the feature vectors.
Contextual bandit algorithms tackle the following issues: different preferences for users, different characteristics for news and changing news set. At time t, a user (with some features) arrives on a website, several items (with some features) could be presented to him. For each of these items, a context x_t is formed based on the user features, the items features as well as the website features. The contexts $(x_t)_t$ will form the set D_t. If x_t is chosen, then this item will be presented to the user. A feedback will be received, which depends on x_t: $r_t = f(x_t) + \epsilon_t$. f could be the time spent on a website, the amount of money spent, the number of clicks, etc. EXP4 or epoch-greedy are contextual bandit algorithms but they seem to face still unsolved challenging problems. EXP4 has a $O(\sqrt{T})$ regret but may experience high computational complexity; epoch-greedy in the other hand, has low computational complexity but high regret $O(T^{2/3})$. EXP4 and epoch-greedy algorithms are developed more in details in the second part.

In more details, the Contextual Bandit Approach and Protocol is as follow:

- Preliminaries: Finite number arms, the reward varies linearly with the context $E[r(x,a)] = \phi_{x,a}^T \theta_a^*$
- Procedure: at time t, user x_t arrives and a new set of articles A_t is provided. x_t and a are described by a feature vector $\phi_{x_t,a}$ (with a in A_t). The learner chooses a news a_t in A_t and receives a reward $r_t(x_t, a_t)$.
- The optimal news is $a_t^* = argmax_a E[r_t(x_t, a_t)]$ in A_t.

3 "Real-world" recommendation problems with bandit

In this section, we will develop the different problems encountered by Bandit algorithms in real-world and some attempts to encounter them. The problems encountered are either related to a lack of information from the environment or to an optimistic assumption of Multi Armed Bandit. Indeed, the dynamic content, the diverse forms of digital content, the non exhaustive user/items features, the delayed conversion as well as the Privacy Constraints are real-environmental problems that face Multi Armed Bandit.

Dynamic content $\&$ **Fast Changing Market Place.** As explained in [4], in a dynamic marketplace, ϵ-greedy algorithm approach may not be suitable (because it decreases the parameter ϵ monotonically). Instead, it is more suitable to adaptively balance the trade off between exploitation and exploration with the dynamic content. The authors updated ϵ dynamically (not necessarily decreasing), it is optimally chosen at each step from a sampled finite set of values $(\epsilon_1, ..., \epsilon_T)$. A probability vector $\mathbf{p} = (p_1, ..., p_T)$ is introduced where p_i is the probability of using ϵ_i. First, $p_i = \frac{1}{T}$ for all i. p_i is updated throughout the recommendation, the idea is to increase p_i if we receive a click from using ϵ_i. The results with this algorithm outperform the ϵ-greedy algorithm, the varying ϵ is more adapted to real-world dynamic changing market place. Nonetheless, even though ϵ is updated considering the context at each time t, it is still a context-free algorithm which doesn't provide a personalized recommendation per user. Contextual Bandit Approach tackle better the dynamic changing environment.

Non-stationnarity and regret. In real life applications of recommender systems there are a lot of sources of non-stationarity : for instance in movies recommendation, the interest of users for movies might change over time, and so on... In the MAB setting where reward distributions are assumed to be stationary, we can derive bounds for the regret for UCB and ϵ-greedy algorithms that are logarithmic in the number of trials T, which is shown to be optimal [16]. However when we introduce non-stationarity in our bandit problem we can no longer derive such bounds. Several algorithms have been proposed to tackle the contextual bandit problem. An extensive comparison of these algorithms can be found in [17]. Exp4 [18] algorithm solves the contextual problem by updating weights given to experts at each time step. The regret is bounded by $O(\sqrt{KTln(N)})$ where K is the number of arms and N is the number of experts. Not surprisingly, we notice that the higher the number of expert and the number of arms, the higher the bound, hence the motivation to keep the number of items to propose relatively small.

Algorithm 1 EXP4

```
parameter: Real $\gamma \in (0, 1]$.
Initialization: $\omega_i(1) = 1 for i = 1, ..., N$
for each t=1,2,...
  1.Get advice vectors $\xi^1(t), ..., \xi^n(t)$
  2.Set $W_t = \sum_{i=1}^{N} \omega_i(t)$ and for j =1,...,N set $p_j(t) = (1-\gamma) \sum_{i=1}^{N} \frac{\omega_i(t)\xi_j^i}{W_t}$
  3.Draw action $i_t$ randomly according to the probabilities $p_1(t), ..., p_K(t)$
  4. Receive reward $x_{it}$ ∈[0,1].
  5. For j=1,...,K set
        $\hat{x}_j(t) = \begin{cases} x_j(t)/p_j(t), & \text{if j} = i_t \quad (1) \\ 0, & \text{otherwise} \quad (2) \end{cases}$
  6. For i=1,...,N set
     $\hat{y}_i(t) = \xi^i(t).\hat{x}(t)$,
     $\omega_i(t+1) = \omega_i(t) exp(\gamma \hat{y}_i(t)/K)$
```

The Epoch greedy algorithm [23] which alternates at every epoch between exploration steps first (pull the arms uniformly at random) and then exploitation steps has a worse regret bound in T ($O(T^{2/3})$) but has a much better dependence on the size of the set of predictors and has reduced computational complexity. Also note that epoch-greedy does not need to know T in advance and converges in probability rather than expectation.
In [1] the authors propose the LinUCB algorithm to solve the contextual bandit problem defined in

the second part. The algorithm applies Ridge Regression to find estimates of the parameters of the arms and uses a UCB-like selection to select the arm at time step t.

Algorithm 1 LinUCB with disjoint linear models.

```
0: Inputs: $\alpha \in \mathbb{R}_+$
1: for $t = 1, 2, 3, \ldots, T$ do
2:     Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
3:     for all $a \in \mathcal{A}_t$ do
4:         if $a$ is new then
5:             $\mathbf{A}_a \leftarrow \mathbf{I}_d$ ($d$-dimensional identity matrix)
6:             $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ ($d$-dimensional zero vector)
7:         end if
8:         $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
9:         $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}}$
10:    end for
11:    Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbi-
       trarily, and observe a real-valued payoff $r_t$
12:    $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
13:    $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
14: end for
```

Under certain conditions it can be proven that this algorithm exhibits a strong regret bound in $O((KT)^{1/2})$. As expected, the regret of all of these algorithms increases with the number of arms K. For a number of applications, however, the number of items remains small and stable (such as news recommendation, where articles are frequently removed). To our knowledge, no Bandit algorithm applied to recommendation systems has an independent bound regret from K and T. This constitutes another problem for Bandits application on real-world recommendation systems.

Nontransparent Game. Contextual Bandit deems that the game is transparent ([8]). Indeed, it assumes that the features used are known and uses them to generate the payoff of each action. This assumption is false in the reality and will consequently introduce bias during online learning. Take news recommendation as an example. If an information is not given to the learner it will constantly recommend suboptimal arms. Moreover, it is impossible to learn all the relevant attributes before developing a recommender system, (and the huge number of features have a high computational complexity). It is also misleading to learn the missing features when the recommendation system is deployed. Besides, many very important factors are unknown to the recommender system (such as income) due to privacy constraints.
[8] proposes a method to improve the recommender system using hidden features. Indeed, they proposed a Contextual Bandit with Hidden Features where they introduced the concept of hidden features of item a_t, denoted ψ_{a_t} in addition to the observed features ϕ_{a_t}, the reward becomes : $(\phi_{a_t}, \psi_{a_t})^T(\theta_u^x, \theta_u^h) + \eta_t$, where $\phi_{a_t} \in \mathbb{R}^d$ and $\psi_{a_t} \in \mathbb{R}^l$. η_t is the random noise drawn from a zero-mean Gaussian distribution. We denote $\theta_u = (\theta_u^x, \theta_u^h)$, θ_u^x is the observed bandit parameter for each user and θ_u^h is the hidden bandit parameter for each user. They assume that l the dimension of hidden features is known to the learner ahead of the experience. The observed features are the only features revealed to the learner.

They used Ridge Regression to estimate the unknown bandit parameter θ_u for each user and ψ_a for each item:
$min_{\theta_u,\psi_a} \sum_{t=1}^{T}((\phi_{a_t}, \psi_{a_t})^T\theta_u - r_{a_t,u})^2 + \lambda_1||\theta_u||_2 + \lambda_2||\psi_a||_2$,

where λ_1 and λ_2 are the trade-off parameters for L2-regularization. We note $**$ for the estimated parameters and $*$ for the true parameter. Therefore, we have a closed forme: $r^{**}_{a_t,u} = (\phi_{a_t}, \psi^{**}_{a,t})^T\theta^{**}_{u,t}$ with:

- $\theta^{**}_{u,t} = A^{-1}_{u,t}b_{u,t}$, estimated parameter with $\theta^{**}_{u,t} = (\theta^{**,x}_{u,t}, \theta^{**,h}_{u,t})$.
- $\psi^{**}_{a_t} = C^{-1}_{a_t}d_{a_t}$, estimated hidden feature.

  with :
- $A_{u,t} = \lambda_1 I_1 + \sum_{t'=1}^{t}(\phi_{a_{t'}}, \psi^{**}_{a_{t'}})(\phi_{a_{t'}}, \psi^{**}_{a_{t'}})^T$
- $C_{a_t} = \lambda_2 I_2 + \sum_{t'=1}^{t}\theta^{**,h}_{u,t'}\theta^{**,hT}_{u,t'}$
- $b_{u,t} = \sum_{t'=1}^{t}(\phi_{a_{t'}}, \psi^{**}_{a_{t'}})r_{a_{t'},u}$
- $d_{a_t} = \sum_{t'=1}^{t}\theta^{**,h}_{u,t'}(r_{a_{t'},u} - \phi^T_{a_{t'}}\theta^{**,x}_{u,t'})$

The adapted contextual bandit approach with hidden features gives a reasonable prediction. Nonetheless, this approach may not be efficient when we do not have sufficient observations at early stage. It is necessary to explore less promising items to reduce the regret. Upper Confidence Bound (UCB) has showed good results while tackling with this problem. It uses the estimated confidence of predicted rewards on the selected arms. Based on a lemma developed in [8], it can be shown that by noting α^u_t and α^a_t as the upper bound of respectively $||\theta^{**}_{u,t} - \theta^*_u||_{A_{u_t}}$ and $||\psi^{**}_{a_t} - \psi^*_{a_t}||_{C_{a_t}}$, therefore the optimal arm is:
$a^{**}_t = argmax_{a_t \in A}((\phi_{a_t}, \psi^{**}_{a_t})^T\theta^{**}_{u,t} + \alpha^u_t\sqrt{(\phi_{a_t}, \psi^{**}_{a_t})A^{-1}_{u,t}(\phi_{a_t}, \psi^{**T}_{a_t})} + \alpha^a_t\sqrt{\theta^{**}_{u,t}C^{-1}_{a_t}\theta^{h,T}_{u,t}}$ (1).
The first term of (1) is the predicted payoff of arm a_t to user u based on the current estimation of bandit parameter. The second and thirds term reflect the uncertainty over θ_u and ψ_a which show the exploitation of less promising arms. As there are more observations, the exploration terms shrink, this is the balance between Exploration and Exploitation. They called this algorithm Hidden LinUCB or hLinUCB. The advantages of this algorithm are: First, linear computational complexity with respect to the arms and the users. Second, the more observations are available, the lower the exploration parameters are. It was shown in the paper that when the reward is governed by both observed and hidden features then LinUCB suffers a cumulative regret that grows linearly, while hLinUCB converges quickly. This verifies their motivations to learn the hidden features in contextual bandit. Nonetheless, as the hLinUCB requires the dimension l of the hidden features as input, it makes it quite hard to estimate this parameter in the beginning of the experiment. Furthermore, over-estimating the number of hidden features has dramatic consequences on the regret. The reason explaining this behavior is that the upper bound was computed for hidden feature dimension close to the ground-truth. Therefore, an idea would be to calibrate the number of features such as to minimize the regret, but it may take a lot of time to try several values of l. We may consider that the number of hidden features for a given experiment stay constant. This approach is quite interesting as it proved efficiency on real data. We can therefore minimize the regret on real data even though we don't have an exhaustive set of features.

Delayed Conversion. In web ads, there is a delay between received clicks and their display, as for the buying decision it may take hours, or even longer to happen([9]). Many advertisers would prefer not to pay for an ad impression unless that impression leads the user to the advertiser's website. Chapelle ([10]) came with a a solution based on the idea that each action may engender a future reward that will occur within a stochastic delay. He applied his solution to Criteo data and discovered that only 35% of the conversions occur within the first hour, 50% of the conversions occur after 24 hours. Moreover, approximately about 13% of the conversions could be attributed to marketing actions (clicks) displayed more than two weeks before. This remarks are very related to Criteo data and may change with the kind of data we consider. Chapelle discovered that the delay distribution is well fitted by an Exponential distribution especially when conditioned on context variables given by the advertiser. [9] proposes a complete analysis of delay impact under the hypothesis that its distribution is known:

- At each round $t \in \mathbb{N}^*$, the learner chooses an arm $a_t \in (1, ..., K)$, a conversion indicator $C_t \in (0, 1)$ is generated with value 1 when there is a conversion in the future otherwise 0, $D_t \in \mathbb{N}$ is the delay between the click and the conversion (undefined if C_t is equal to 0). The agent will receive $Y_t = \sum_{s=1}^{t} X_{s,t}$ which is the sum of the contributions of action in time s before time t, $X_{s,t} = C_s \mathbb{1}_{D_s = t-s}$. $X_{s,t}$ is the possible contribution of the action taken at time s to the conversions observed at time t. $X_{s,t}$ is equal to 1 if and only if, C_s is equal to 1 and $D_s = t - s$, which means that the action at time s has a conversion of 1 and the delay is $t - s$.

- Let X be a set of features, the conditional probability of delay of $t - s$ for action in s is $P(D_s = t - s|X = x, C_s = 1) = \lambda(x)exp(-\lambda(x) * (t - s))$. Some other distributions can be used to model time between events such as Weibull, Gamma, Log-Normal, but the exponential distribution showed a good fit to the empirical delay distribution as well as a practicality of use.

- Let's note a^* the optimal arm and define $r(T) = \sum_{t=1}^{T} Y_t$ is the cumulative reward of the learner and $r^*(T)$ the cumulative reward of the optimal arm. They showed that the regret is consequently equal to : $L(T) = E[r(T) - r^*(T)] = \sum_{s=1}^{T} E[\theta_{a^*} - \theta_{a_s}] * P(D_s \leq T - s)$.

They based their model on the above to model the delayed feedback. Nonetheless, the modelization of delay is an ongoing process and the results are not convincing. Delay is an important problem for bandits in recommendation system. A similar challenge is encountered in many other applications such as in personalized treatment planning where the effect of treatment on a patient's health may take time without saying that it may be difficult to determine which of several past treatments caused the change in the patient's health. Likewise, when an add for a clothe brand is displayed on many websites it becomes difficult to know which website originated the purchase act. All this real-world problems make the recommendation follow-up much harder, there is no efficient algorithm (up to our knowledge) that showed efficiency while tackling these problems.

Independence of the arms : Another problem that arises with bandit is the Independence of arms. In a simplistic model we might assume that arms are independent but this is hypothesis doesn't hold in reality and we have to model the correlation between arms. For instance, considering a movie recommendation system : the movie Cinderella shares more features with the movie Sleeping Beauty than with the movie Rambo and we would like to model this such that when an arm is pulled we gain insight on similar arms.

In our above formulation of the contextual linear bandit, there is an unknown vector parameter for every arm θ_a^*, the expected reward: $E[r(x, a)] = x_{t,a}^T \theta_a^*$. In such a setting, there is no shared parameters across arms. To tackle this problem, in [1] the authors introduced features that are shared by all arms β. To model that another term is added in the expression of the expected reward:
$E[r(x, a)] = x_{t,a}^T \theta_a^* + z_{t,a}^T \beta^*$. The model obtained is called a "hybrid model" This new formulation leads to a slightly more complex formulation of LinUCB than the one presented previously but enables arms to share common features in an elegant way.

However, contextual linear bandit assume that we are in possession of the context (information about users and items). In fact, sometimes we don't have access to that data for practical reasons (security, etc. . .), therefore we cannot use the technique mentioned above and we have to account for the dependencies in another way.

A means of doing that is to perform clustering of the arms. Thus pulling an arm yields information about the other arms belonging to that cluster. In [22], a generative model (Latent Dirichlet Allocation) used in topic modeling is adopted to infer the cluster of arms. Inference of latent variables is performed with particle learning. The algorithm used for choosing which arm to pick can be either Thompson strategy or UCB. The proposed algorithm outperforms classical algorithms on the “CTR” metric for the Yahoo and MovieLense dataset.

Another approach adopted in [13] is the "co-clustering". The idea is that each item induces a clustering of users, each user cluster regrouping users that react similarly to that item. To further reduce the complexity of the algorithm, the authors introduce clustering of the items, where items belonging to the same item cluster induce the same clusters among users. The whole algorithm relies on edge deletion which provides sparse structures and therefore lower computational needs. At each time step, a new user arrives, and an item is proposed to the user based on upper-confidence-based trade-offs between exploration and exploitation. The reward is then observed and used to update the clusters on the user side and on the item side. The results of COFIBA are very promising in terms of the CTR metric compared to other bandit algorithms (LinUCB and its variants such as LINUCB-IND and LINUCB-V) and proves that the clusters provides useful information for recommendation.

Evaluation. One problem that comes up with bandits is how to evaluate our algorithm. That is, we want to know how well a policy π is doing on the data. One way of doing that would be to evaluate our policy on live data. However this is infeasible for practical reasons. Usually what we have at hand is offline collected data that corresponds to a completely different policy than our algorithm π, which we’ll call the logging policy. Therefore we only observe the rewards of the arms that were chosen by the logging policy. So how to evaluate π on this offline data ? One solution would be to create a generator to build a model from the logged data and then to evaluate π on that simulator. However this would introduce bias. Alternatively, in [1] the authors proposed a simple algorithm and easy to implement to evaluate π on offline data. The algorithm relies on two assumptions, namely that the episodes are drawn iid from a distribution D and that at each time step the arm pulled from the logging policy is pulled uniformly at random. Then the idea is to keep only the events for which the arm pulled by the logging policy and π are the same : if this is the case we add that event to our history to refine our future choices and we update the cumulative rewards. Intuitively each episode is only retained with probability $1/K$ where K is the number of arms so the bigger K is, the more events we need.

Cold Start Situation. How can we effectively recommend items to a user about whom we have no information? Different approaches were proposed, one of them is a random recommendation of rate-plan but the chance that the user accepts it is low considering and there is too much randomness. Another approach is to create some typical profiles of users and to propose to the new users the best item of each profile [19]. Therefore, by knowing the new user’s taste we can classify him/her in one of the profiles and then refine more on his/her profile afterwards with new information. According to [14], new users are best served by models already built in system: by selecting a prediction model from a set of strong linked users, it might offer better results than building a personalized model for full cold-start users. Another approach when the new user is admitted into the system in exchange of his social information, the system collects enough preference data from the user and builds an initial prediction model. It is assumed that users who share a common background also share a common taste in products. Later, as the user provides ratings, the system improves the model. Nonetheless, it is very hard to tackle Cold Start Situation, the results are much more accurate if we provide information about the user (such as gender, preference) while he/she registers in the website because it will help categorizing the user into an already-created profile and then recommend an item based on this profile. However, we know that it’s quite inconvenient (and time consuming) to answer questions when you register in "Medium" website for instance.

Multiple arms: In many practical cases, we can encounter situations with a large number of arms for instance in the case of clothes or job offers. Some papers approached the problem of infinite number of arms for multi-armed bandit. Berry, in [21], proposes a *k-failure strategy*: for each positive integer k, the strategy calls for using the same arm until that arm produces k failures, if so, it calls for switching to a new arm (never returning to arms that have yielded failures). For the 1-failure strategy, the failure proportion of this strategy in n trials is asymptotically $\frac{1}{ln(n)}$. They showed that the 1-failure strategy has the smallest asymptotic expected failure rate among k-failure strategies. The 1-failure strategy indicates a switch to a new arm whenever the current arm produces a failure. Which means that it may discard an arm that has given a very large number of successes and a single failure seems wasteful. Therefore, tackling a multiple arm bandit problem is an open problem with no solution found yet.

Intrusiveness and Data Protection: In [5], the author explains that the recommendation systems are intrusive as they require a feedback from the user which is usually the number of clicks or the amount of money sent. Multi Arm Bandit require feedback in order to decrease the regret and are closely dependent on this problem. What is more, in contextual linear bandit, features of the user are needed to produce the context. To do that companies use cookies and private information of the user without him being aware of it. With the rising concern over the data protection, one can question the future of Multi Arm Bandit if the feedback is restricted in the future.

4 Conclusion

In this report, we tackled the different algorithms of Multi Arm Bandit used in recommendation systems. They are used mainly because they present a good trade-off between Exploration and Exploitation. Nonetheless, these algorithms face several real-world problems. Indeed, they have to face a Dynamic content because of an ever changing context (items are constantly updated for e.g). The contextual linear bandit has been extensively studied in the literature as an approach to recommendation system. In this paper, we reviewed different algorithms, each of them has an interesting property, be it in an interesting regret bound or an advantageous computational complexity. However, as far as we know, there is no algorithm that performs better that the others at every level and that is commonly used in industry. The privacy constraints also hinder the development of Multi Arm Bandit in recommendation systems. Data Protection is becoming a thorny issue. Besides, in Context Based algorithms where context features are used, the features are not exhaustive and many important features are unknown. An additional problem is dealing with a new user whom we have no previous information (Cold Start Problem). Delay is another problem, because there is always a delay between the buying decision and the add display what makes recommendation hard. Furthermore, the arms may not be independent, contrarily to what Multi Arm Bandit presuppose, therefore we have to adapt the formulation of the classic problem to take that correlation into account. Two clustering algorithms were presented concerning this issue, and these approaches seem to be promising. Even though, many attempts were made to improve the "classical" MAB algorithms to tackle these problems, it is an open research subject. Indeed, many papers that we encountered dealing with these problems are very recent. Besides, many of the improvements that were made on the Bandit algorithms, to apply to real-world use, require intrusive information (to improve the knowledge about the user) which can face issues with the growing concern over Data Privacy.

References

[1] Lihong Wei Chu, John Langford and Robert E. Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation. *the Nineteenth International Conference on World Wide Web* 2010.

[2] Jonathan Louedec, Max Chevalier, Josiane Mothe, Aurélien Garivier and Sébastien Gerchinovitz. A Multiple-Play Bandit Algorithm Applied to Recommender Systems. *FLAIRS Conference* 2015.

[3] Djallel Bouneffouf, Amel Bouzeghoub and Alda Lopes Gançarski. A Contextual-bandit Algorithm for Mobile Context Aware Recommender System. *ICONIP'12 Proceedings of the 19th international conference on Neural Information Processing* 2012.

[4] Wei Li, Xuerui Wang, Ruofei Zhang, Ying Cui, Jianchang Mao and Ron Jin. Exploitation and Exploration in a Performance based Contextual Advertising System. *KDD '10* 2010.

[5] Georgia Koutrika. Recent Advances in Recommender Systems: Matrices, Bandits, and Blender. *EDBT* 2018.

[6] Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering* 2005.

[7] Mark Collier and Hector Urdiales Llorens. Deep Contextual Multi-armed Bandits. 2018.

[8] Huazheng Wang, Qingyun Wuan and Hongning Wang. Learning Hidden Features for Contextual Bandits. *CIKM '16* 2016.

[9] Claire Vernade, Olivier Cappé and Vianney Perchet. Stochastic Bandit Models for Delayed Conversions. 2017.

[10] Olivier Chapelle. Modeling Delayed Feedback in Display Advertising. *KDD* 2014.

[11] Ciara Pike-Burke, Shipra Agrawal, Csaba Szepesvári and Steffen Grünewälder. Bandits with Delayed, Aggregated Anonymous Feedback. 2017.

[12] Lihong Li, Wei Chu, John Langford and Xuanhui Wang. Unbiased Ofine Evaluation of Contextual-bandit-based News Article Recommendation Algorithms. *WSDM* 2011.

[13] Shuai Li, Alexandros Karatzoglou, Claudio Gentile. Collaborative Filtering Bandits.

[14] Cricia Z. Felicio, Philippe Preux. Preference-like Score to Cope with Cold-Start User in Recommender Systems. *SIGIR* 2016.

[15] Omar Besbes, Yonatan Gur, Assaf Zeevi. Stochastic Multi-Armed-Bandit Problem with Non-stationary Rewards. *NIPS* 2014.

[16] T.L.Lai, Herbert Robbins. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics* 1985.

[17] Li Zhou. A Survey on Contextual Multi-armed Bandits. *ArXiv* 2015.

[18] P.Auer, N.Cesa-Bianchi, Y.Freund, R.Schapire. The non-stochastic multi-armed bandit problem. *Society for Industrial and Applied Mathematics* 2002.

[19] Hai Thanh Nguye, Anders Kofod-Petersen. Using Multi-armed Bandit to Solve Cold-start Problems in Recommender Systems at Telco. *MIKE* 2014.

[20] Yizao Wang, Jean-Yves Audibert, Rémi Munos. Algorithms for Innitely Many-Armed Bandits. *NIPS* 2009.

[21] Donald A. Berry, Robert W. Chen, Alan Zame, David C. Heath, Larry A. Shepp. Bandit Problems With Infinitely Many Arm. *The Annals of Statistics 1997, Vol. 25, No. 5, 2103-2116* 1997.

[22] Qing Wang, Chunqiu Zeng, Wubai Zhou, Tao Li, Larisa Shwartz, Genady Ya. Grabarnik. Online Interactive Collaborative Filtering Using Multi-Armed Bandit with Dependent Arms. *IEEE Transactions on Knowledge Data Engineering* 2017.

[23] John Langford, Tong Zhang. The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits. *NIPS* 2007.