Survey of multi-armed bandit algorithms applied to recommendation systems

Elena Gangan, Milos Kudus, Eugene Ilyushin

***Abstract*—The main goal of this paper is to introduce the reader to the multi-armed bandit algorithms of different types and to observe how the industry leveraged them in advancing recommendation systems. We present the current state of the art in RecSys and then explain what multi-armed bandits can bring to the table. First, we present the formalization of the multi-armed bandit problem and show the most common bandit algorithms such as upper confidence bound, Thompson Sampling, epsilon greedy, EXP3. Expanding on that knowledge, we review some important contextual bandits and present their benefits and usage in different applications. In this setting, context means side information about the users or the items of the problem. We also survey various techniques in multi-armed bandits that make bandits fit to the task of recommendations better; namely we consider bandits with multiple plays, multiple objective optimizations, clustering and collaborative filtering approaches. We also assess bandit backed recommendation systems implemented in the industry - at Spotify, Netflix, Yahoo and others. At the same time we discuss methods of bandit evaluation and present an empirical evaluation of some notorious algorithms. We conduct short experiments on 2 datasets to show how different policies compare to each other and observe the importance of parameter tuning. This paper is a survey of the multi armed bandit algorithms and their applications to recommendation systems.**

***Ключевые слова*—Recommendation, Multi-armed bandits, Reinforcement learning**

I. Introduction and motivation

Recommender systems are the base of many internet services nowadays, and widely are used by big Internet companies like Netflix [1], Spotify [2], [3], Amazon, Yahoo [4], [5], etc. for their services. The main goal of a recommender system is to maximize the engagement of each user or some other metric like profit or client satisfaction. At the highest possible level, the recommender system should be able to suggest an item with which it is likely for the user to interact with. This is usually done based on a previous history of user/item interaction, and/or user metadata such as age, gender, geographical location, etc. Recommendation systems are essential in user engagement, detention and satisfaction. They could tailor a general experience into a personalized one, profiting both the user and business.

Manuscript received March 15th, 2021.

E. Gangan. Author is a MSc graduate at Babes-Bolyai University, Faculty of Mathematics and computer Science, Cluj-Napoca, (email: lena-gangan2@gmail.com)

Milos Kudus is a Ph.D. canditate at the Faculty of Technical Sciences, Energetics, Electronics, and Telecommunication (Signal processing), Novi Sad, (email: miloskudus@gmail.com)

E. Ilyushin. Author is a PhD candidate at Lomonosov Moscow State University (MSU) The faculty of Computational Mathematics and Cybernetics, (email: eugene.ilyushin@gmail.com)

A. Traditional approaches in RecSys

1) Collaborative filtering: Collaborative filtering (CF) methods usually try to combine users (or items) into groups of similar users (or items). In the user-oriented collaborative filtering if the user comes to interact with the system, the recommendations will be made based on the preferences of similar users. In contrast, in item-oriented collaborative filtering methods the recommendation will be made based on items that are related to ones that the user has already interacted with. As noted in [**?**], CF has 2 approaches: memory-based (a heuristic approach, divided into item-based and user-based categories; based on finding similar items/users) and model-based approach. The model-based approach incorporates Matrix Factorization, Clustering and Deep Learning (auto-encoders, restricted Boltzmann machines).

2) Matrix factorization: The family of matrix factorization algorithms - MF [6], NeuMF [7], BiasedMF [8], etc.) try to represent both users and items in the same low dimensional latent space. Predicted compatibility between user and item is then computed usually by dot product (or some other more complicated function like MLP [7]) of their corresponding low dimensional latent representation. When a user comes to interact with the system, the item with the highest score for that user is recommended. They give good results in practice and are still considered as standard for recommendation systems. *Single Value Decomposition* (SVD) is a widely used technique to perform Matrix Factorization and probably one of the most popular approaches in RecSys. Its idea is to represent users and items as latent vectors and compute the ratings using these vectors. Single Value Decomposition is flexible, but its performance can suffer from item under or over-rate. Also SVD can be computed only for dense matrices. SVD has some extensions, like SVD ++ and Asymmetric SVD ++ (both use *implicit feedback*), and Time SVD++ (incorporates knowledge about time) [9].

3) Downsides of traditional recommendation systems: Collaborative filtering methods have limited capabilities, and are not good with highly sparse data. Matrix factorization methods are static, once trained it is not trivial to add a new user or item, or even update it with new online data. Accurate clustering (in k-means, for example) depends on correctly choosing the initial clustering parameter k. In those cases the best way to update the model is to retrain the whole thing again, which is usually not practical for systems with large amounts of data coming in. The methods suffer from the cold-start problem when there is no data for corresponding user or item. In their plain variant, most methods don't utilize user and item metadata, but there are several modifications which make that possible.

B. Multi-armed bandits

Bandit algorithms are becoming a encouraging approach to creating better recommendations systems [10]. To update recommender systems dynamically we can reformulate the problem as a reinforcement learning task, specifically as a multi-armed bandit(MAB) problem. In a multi-armed bandit setting, the agent pulls an arm (makes a prediction) based on the history of interactions with the environment, user's metadata (if available), and item's metadata (if available). The environment gives feedback about agent's prediction which can be used by agents to make better predictions in the future.

1) Multi-armed bandits as recommender systems: One approach to use MAB as a recommender system would be to consider the user and their actions as the environment, and items (or clusters of items [11]) - as arms. An agent can observe information about the user (if available) and based on that observation and its inner state, the agent should recommend an item (or items) to the user. If the user interacts with the item, positive feedback is given, if not, negative feedback is given. Interaction can be a click, like, share, sell, etc. The agent should maximize those interactions in the long run.

One of the problems that is known to be handled by bandits better than other traditional approaches is the cold-start problem, for example the scenario when the system does not have any information about the user in order to make an informed decision. This is considered one of the most important benefits that MABs can bring to the table. Recommending content to new users was modeled as a cold-start MAB problem in a series of works [12], [13], [14] and seems a promising research direction.

a) : The remained of the paper is divided into the following sections: Section 2 explains the multi-armed bandit problem. Section 3 describes the most popular non-contextual multi-armed bandit algorithms. Section 4 introduces contextual features to the MAB problem. Section 5 explains different MAB strategies in recommendation systems. Section 6 section describes the evaluation methods for MAB algorithms. Section 7 takes a survey of use cases of multi-armed bandit systems across the industry. The last sections include experiments with some of the described algorithms and conclusions to our work.

II. Theoretical overview of multi-armed bandits

Multi-armed bandits have seen an increasing interest in the academia and industry over the last years. Bandits are a special area of Reinforcement Learning that is specialized in making decisions under uncertainty [15].

A. Multi-armed bandit problem description and formalization

Given some number of choices (arms), a MAB policy should choose the arm which gives the greatest average reward. Reward probability distributions are considered static in the classical MAB setting. The problem of probability distributions that are changing over time is known as an adversary bandit problem. In a non-contextual bandit scenario the decisions are made based only on the previous choices and their corresponding rewards. A good policy should find the arm with the highest average reward as fast and as certain as possible and utilize only that arm.

1) Formalization of the problem: The agent and the environment interact in sequence over n rounds.There are k arms to choose from. For each round $t = 1, ..., T$ the agent chooses an action $a_t \in \mathcal{A}$, based on which the environment returns a reward $r_t \in R$. Only the rewards for the actions that were taken are known. Usually the number of actions n is finite, but sometimes this can not be the case and $(n = \infty)$. To formalize:

Algorithm 1 MAB Problem

for each round $t = 1, 2, ...T$ **do**
1. The agent takes action $a_{t,k} \in \mathcal{A}$
2. The environment reveals reward $r_{t,k} \in [0, 1]$ for chosen action

end for

B. Exploration vs exploitation dilemma

The central idea of the MAB algorithm is to correctly decide when to explore (search for new items) or exploit (play the chosen arm). Ideally, the MAB algorithm should always choose the arm with the highest average reward. The problem is that the algorithm's approximation of arms average rewards is never perfect, even with a relatively large number of plays for each arm. In the beginning, when there is no previous data, the algorithm could not know anything about arms' average rewards. If it chooses one arm at random and always pulls that arm, that arm may be suboptimal and thus the algorithm is suboptimal. The approximation of arms average rewards will tend to be more accurate the larger number of times that specific arm is played. It could be useful for the algorithm to sometimes choose a seemingly suboptimal arm to get a more accurate approximation of its average reward. It might turn out that in the fact that arm has a larger average reward value than the seemingly optimal arm.

C. Concept of reward and regret

If the average reward for each arm is known before the first play of the algorithm, the algorithm could choose the optimal arm from the start and get a corresponding reward from the environment. In the case where average rewards are unknown before the first play, the policy will sometimes, especially at the beginning, choose suboptimal arms and observe suboptimal rewards.

1) Defining reward: The policy's goal is to maximize the total cumulative reward:

$$S_n = \sum_{t=1}^{n} r_t \tag{1}$$

which depends on the actions of the policy and the rewards returned by the environment. The reward is a random value with unknown distribution to the policy, it is dependent on the previous action of the policy towards the environment. The policy's goal is to approximate this distribution in order to make better predictions.

Based on the nature of the rewards, we differentiate between the adversarial and stochastic bandits. The schema of gathering rewards under in the stochastic and adversarial fashions is exemplified by [16], stating that the adversarial reward model does not make any restrictions about the sequence of rewards,

while the stochastic model assumes the rewards as being an independent and identically distributed (i.i.d from now on) sequence of random variables. The adversarial model simultaneously selects rewards, with the goal of minimizing the regret defined as the sum of differences of the drawn arm versus optimal arm in the adversarial model; while in the stochastic model each reward is drawn independently from a probability distribution and the regret is defined as the sum of differences of the respective means of the probability distribution [16].

2) Defining regret: Regret can be defined as deficit suffered by the policy relative to the optimal policy. In other words, regret is the difference between the total expected reward of a policy over n rounds and the real reward gained over the course of n rounds [17]. The regret is the result when the policy does not play all the optimal arms. Suppose we have $i = 1, 2...k$ arms and a sequence of unknown rewards for each time step $t = 1, 2, ...n$ such as $r_{i,1}, r_{i,2}....$ at each time t the policy selects an action $a_{i,t}$ with $r_{a_{i,t}}$ associated reward. Then, the regret of policy π can be defined as:

$$R_n(\pi) = \max_{i=1..k} \sum_{t=1}^{n} r_{i,t} - \sum_{t=1}^{n} r_{a_{i,t}} \tag{2}$$

Minimizing the regret is equivalent to maximizing the total reward. Depending on algorithm, different regret bounds can be defined.

D. Example

Imagine a scenario where 3 one-armed bandit gambling machines are available to the player. Let's assume that they differ in their average rewards where the first one has an average reward of 0.1, the second reward of 0.7, and third 0.6 and that those rewards are unknown for the player. The player should choose the second machine to have a maximal gain (minimal loss), but at the start, he doesn't know anything about their average rewards. If he chooses one of them, let's say third, he might be stuck at it thinking that the reward of that machine is the greatest - not knowing that in the fact the second machine has a better reward. The best way to figure out which machine has the best reward is to play on all 3 of them and approximate their rewards. More times the player chooses one specific machine, becomes more certain of its average approximation. The player should soon realize that the first machine has a substantially smaller reward than the second and third machine and he can quit playing them. It will take more plays of the second and third machines for the player to figure out that the second machine has a slightly better reward then third.

III. Popular non-contextual multi-armed bandit algorithms

In the case where contextual features are not available, we can use non-contextual MABs. If we can assume i.i.d. (independent and identically distributed) rewards, we have a stochastic bandit problem, if not then we have an adversarial bandit problem. Finding the best arm while trading exploiting vs exploration is he main aspect of MAB problems; exploiting only the best arms chosen at one moment could give bad results in a long run because the optimal arms can change over time, that's why it is important to identify certain heuristics to continuously find th best actions (arms). Here we present these classical policies. ϵ-greedy, Upper Confidence Bound (UCB), Thompson sampling (TS), and Softmax as solutions to the stochastic bandit problem, and EXP3 is an example of a solution to an adversarial bandit problem.

A. ϵ-greedy

ϵ-greedy [18], [17] is the simplest, and most obvious solution to the exportation-exploitation dilemma. The policy explores a random arm with ϵ probability. Accordingly, with a probability of $1-\epsilon$, the policy exploits the solution with the highest average reward. In the vanilla version of the ϵ-greedy algorithm, ϵ is constant, but this is not necessary. It makes sense to make ϵ iteration dependent (linearly decreasing, exponentially decreasing, explore with probability ϵ exploit with probability 1-ϵ for some number of iterations, no exploration afterward)

$$arm = \begin{cases} randint(1,k), & \text{if } n \leq \epsilon \\ \underset{i}{\operatorname{argmax}}\, \widehat{r_i}, & \text{otherwise} \end{cases} \tag{3}$$

Where n is a random value drawn from a uniform distribution in the range from 0 to 1. Randint is a function that returns a random integer in a specified range, k is the number of arms; $\widehat{r_i}$ is the predicted average reward of *i*-th arm. This policy is simple and there are policies that give smaller regret. The reason is that there are smarter ways to explore than choosing an arm at random. Soon after the algorithm starts running, it is clear that some arms are sub-optimal, and there is no need to explore them further, that exploration could be utilized for arms that are better candidates for an optimal arm.

B. UCB

The main idea of the upper confidence bound algorithm [19], [17] is to always choose an arm with the highest upper bound. This method is also called *optimism in the face of uncertainty*. The predicted upper bound consists of two elements: predicted average reward and uncertainty given in equation:

$$arm = \underset{i}{\operatorname{argmax}}\, \widehat{r_i} + \sqrt{\frac{2 \ln n}{n_i}} \tag{4}$$

Where $\widehat{r_i}$ is the predicted average reward of *i*-th arm, n is the number of arms pulled so far, and n_i is the number of pulls of *i*-th arm. Arms with higher average rewards will have higher scores. Unexplored arms will tend to have higher scores because of uncertainty estimation. This will give lower regret compared to the ϵ-greedy algorithm because less exploration will be wasted on substantially sub-optimal arms.

C. Thompson Sampling

Thompson Sampling is one oldest and most popular heuristics that dates back to 1933 and has received a lot of attention recently in works on multi-armed bandits. Thompson sampling policy [20], [21], [22], [23] samples value from a beta distribution with different parameters for each arm. The arm with the highest sampled value will be pulled. Beta distribution has 2 parameters a and b and they are set to the

number of success and failure rewards for the corresponding arm.

$$arm = \underset{i}{\operatorname{argmax}}\, beta(success(i), failure(i)) \quad (5)$$

A higher the number of pulls that concluded in success compared to failure means a higher number will be drawn and the chance for that arm to be pulled increases. Arms with sub-optimal success/failure ratio still have a chance to be pulled and in that manner, exploration is performed.

D. Softmax (Boltzmann Exploration)

Softmax policy [24] does exploration and exploitation simultaneously by assigning probability for each sample and at each iteration pulls one of the assigned probabilities. Those probabilities are calculated by the formula:

$$p_i(t+1) = \frac{e^{\frac{\hat{r}_i}{\tau}}}{\sum_{j=1}^{k} e^{\frac{\hat{r}_j}{\tau}}} \quad (6)$$

Where $i = 1...k$ is the number of arms, μ_i represents the reward return average of each arm at the current round, and τ is a hyperparameter. When τ is large randomness of pulls is greater, and thus model explores more. When τ is a small policy is more prone to exploiting solution with the highest reward average.

E. EXP3

Exponential-weight algorithm for Exploration and Exploitation [25] is constructed to deal with adversarial environment. It memorizes list of weights w corresponding to arms. Based on these weights, the policy calculates probabilities of pulls p for each arm i at time-step t based on a equation:

$$p_{i,t} = (1-\gamma)\frac{w_i}{\sum_{j=1}^{K} w_i} + \frac{\gamma}{K} \quad (7)$$

γ is additional parameter which controls the randomness of arm pulling, higher γ means higher randomness and ranges from 0 to 1. K is the number of arms. We estimate reward $\widehat{r_{i,t}}$ with equation:

$$\widehat{r_{i,t}} = \frac{r_{i,t}}{p_{i,t}} \quad (8)$$

$r_{i,t}$ is the actual reward returned by the environment. This step ensures that the conditional expectation of the estimated reward is the actual reward. Based on estimated reward we update weight of the pulled arm:

$$w_i(t+1) = w_i(t+1)e^{\frac{\gamma \widehat{r_{i,t}}}{K}} \quad (9)$$

Un-pulled arms are not updated. Because of introduced randomness update strategy, the algorithm could explore as long as the session lasts. Older rewards hold smaller significance, thus old optimal arm could be forgotten when it is no longer optimal.

IV. Contextual multi-armed bandits

A more advanced type of bandits that make use of the contextual information in order to compute the most optimal arm are called *contextual bandits*. They have different names in the literature: bandits problems with side observations, associative reinforcement learning, reinforcement learning with immediate reward, bandit problems with covariates; but the term *contextual bandits* introduced by Langford and Zhang [26] became the industry standard. The main difference between non-contextual and contextual bandits is that in the latter the environment reveals a context that becomes relevant in choosing the pulled arm.

The context is the side information that the system might have. If we take as an example a platform of movie recommendations, this context can be the information about the *user* (location, browser, watching habits, previous likes etc) or about the *content* (movie category, ratings, actors etc). In such a setting, the reward can be described as some sort of function or mapping of the context information and past rewards.

To formalize, the contextual bandit problem can be viewed as the repeated interaction of the agent with the environment that reveals a context x_t at every time t; the agent performs an action $a_{t,k}$ over T rounds and accumulates the reward $r_{t,k}$ for each round, where k represents the pulled arm from the $\{1, ..., k\}$ arms to be pulled.

Algorithm 2 Contextual MAB

for each round $t = 1, 2, ...T$ **do**
1. The environment reveals the context $x_t \in \mathcal{X}$
2. The agent takes action $a_{t,k} \in \mathcal{A}$
3. The environment reveals reward $r_{t,k} \in [0, 1]$ for chosen action

end for

Cortes [27] formalises the problem in a similar way, but he assumes a stochastic binary reward for each arm through a function of the covariates which differs for every arm but remains the same throughout all rounds.

A. Guided exploration approaches - UCB

As described in the previous chapter, the Upper Confidence Bound model is based on the idea of choosing the best option considering the arm with the highest UCB. This is a very popular exploration strategy and many famous algorithm are based on it. Introduced in 2002 by Auer [19], LinRel is a notable algorithm that pioneered the idea of the contextual bandit problem with linear payoffs. It assumes that the rewards are some kind of the linear combination of the previous rewards of the arm. Auer calles his model "associative reinforcement learning with linear value functions". It is the precursor of the now famous LinUCB. The concept behind the algorithm is similar, except that LinUCB adds an identity matrix in the ridge regression, while LinRel does the regularization by setting the matrix's small eigenvalues to zero [28]. We will not go in detail of explaining this algorithm, but will explain the LinUCB instead, considering it a simplified and easier to implement form of LinRel. However it is important to notice that LinUCB does not have the same regret bound as LinRel [29].

1) LinUCB: LinUCB or linear upper confidence bound algorithm was introduced by Li et al in 2010 and is one of the most popular linear stochastic algorithms. In their work [4] the authors tried to solve the problem of personalized news article recommendations at Yahoo! from the perspective of contextual bandits. The scope was to maximize the user click using the existing contextual information about the user-click

history. They have applied the algorithm on the data from Yahoo! Front Page Today Module. The dataset is constructed of over 33 million events. The algorithm achieved a 12.5% click lift compared to a context-free policy.

LinUCB is an extenstion of the UCB algorithm. The algorithm comes in 2 flavours: LinUCB with Disjoint Linear Models and LinUCB with Hybrid Linear Models.

a) LinUCB with Disjoint Linear Models.: The algorithm assumes that the expected reward on an arm a is linear combination of a d-dimensional feature vector x_t and some unknown coefficient vector $\boldsymbol{\theta}_a^*$ [4]:

$$\mathbf{E}[r_{t,a}|x_{t,a}] = x_{t,a}^{\mathsf{T}}\boldsymbol{\theta}_a^* \tag{10}$$

The coefficient vector $\boldsymbol{\theta}_a^*$ is unknown, but we can have $\hat{\boldsymbol{\theta}}_a$ as the estimate of coefficients using ridge regression on training data at each time t; $\mathbf{A}_a$ is the history matrix of dimension $d * d$, where d is the dimension of the of the features x_t; which put together leads to the chosen arm being the one with the maximum UCB calculated as sum of expected value and the standard deviation with a hyperparameter α:

$$a_t \overset{\text{def}}{=} \arg\max_{a \in \mathcal{A}_t} \left(x_{t,a}^{\mathsf{T}}\hat{\boldsymbol{\theta}}_a + \alpha\sqrt{x_{t,a}^{\mathsf{T}}\mathbf{A}_a^{-1}x_{t,a}} \right) \tag{11}$$

The algorithm is explained in detail in the paper, but to summarize: At each time t a context vector x_t is generated; for each new arm they generate a $\mathbf{A}_a$ matrix (history) of dimension $d * d$ and a response vector(feedback from user) $\mathbf{b}_a$ of dimension d. The vector of coefficients $\hat{\boldsymbol{\theta}}_a$ is calculated using ridge regression and then optimal arm with the higher confidence bound is selected as shown in (11). The matrix and vector of the chosen arm are updated and the algorithm restarts. This approach is called *disjoint* because the parameters are not shared between the arms.

b) LinUCB with Hybrid Linear Models.: This model assumes that the features can either be shared or non-shared between the arms of the model. The expected payoff of the arm thus translated to:

$$\mathbf{E}[r_{t,a}|x_{t,a}] = z_{t,a}^{\mathsf{T}}\boldsymbol{\beta}_a^* + x_{t,a}^{\mathsf{T}}\boldsymbol{\theta}_a^* \tag{12}$$

where $z_{t,a}^{\mathsf{T}}$ represents the encoded features of the current user/article combination and β_a^* is an unknown coefficient vector that is recurrent for all arms. The main difference between the models is that in the *disjoint* we update only the features of the chosen arms, while in the *hybrid* model we update the reward corresponding on the joint feature's interaction. The confidence intervals are computed a little different in the case of the hybrid model and the algorithm can be found in the paper [4].

2) Similar Approaches and extensions of LinUCB: We would like to mention some of the similar approaches to LinUCB found in the literature. Even if a vast series of algorithms use the upper confident bound, here we will try to briefly describe those that are similar with the LinUCB approach.

a) Latent Contextual Bandits: LCB or Latent Contextual Bandit introduced by Zhou and Brunskill in 2016. The approach is based on *partial personalization*. It does not rely on user's features to capture variability, but instead it leverages the latent class structure. The LCB algorithm is composed of 2 phases: the first phase runs LinUCB on *J* users and in phase 2 uses this data to learn *N* latent models to construct policies and create an instance of a bandit for each user in *J*. Evaluated on a news feed dataset provided by Yahoo!, the algorithm outperforms Population LinUCB (runs a single LinUCB model for all arms), giving a 5% higher CTR under specific experiment settings [30].

b) Contextual Combinatorial Cascading Bandit: The Contextual Combination Cascading Bandit (C^3UCB) is designed like an online learning game where the action is combinatorial (it assumes a *super-arm* formed by *base arms*) and the reward model is cascading (the bandit chooses a sequential list of top best arms). At every time step t, the learner is presented a set of contextual information and has to select a subgroup of items under combinatorial constraints (super arm); it then observes the reward for the super-arm and, because of the cascading feedback model, the weights of the base arms in the super arm [31]. The algorithm stops when the user selects an item at a position o down the list of chosen arms. C^3UCB algorithm first computes the upper confidence bounds of the base arms, then selects an action based on it and after updating certain statistics gets a new *d*-dimensional context vector and a new confidence radius.

c) Contextual-Bandit Based Personalized Recommendation with Time-Varying User Interests: Two models with disjoint and hybrid payoffs are considered in order to portrait how the users' preference targeting different object-items shift over time. To address the challenge of time-varying interests, the algorithm employs a change-detection procedure to identify potential changes on the preference vectors. Once a change is detected, an efficient restart is applied to re-estimate the preference vector using up-to-date observations. The disjoint and hybrid models achieve a 2.0% and 2.4% CRT increase over the LinUCB disjoint and hybrid models on Yahoo! news dataset [32].

3) Bootstrapped approaches: Bootstrapping is generally referred to as a way to randomize historical data [33]. It consists in taking resamples of the data of the same size as the original by random selection from the historical data and replacing them. Cortes [27] proposes 2 contextual UCB algorithms - online and offline that take as inputs a number of resamples, oracles and a percentile *p* in order to choose the arm with the maximum reward. The offline variant is refit to a bootstrapped resample, while the online paradigm takes a sample observation weight *w* and updates the oracle with the new observation of context and reward weighted on *w*.

4) Neural Approaches: Recently, some neural approaches with MABs have been developed. *Neural Contextual Bandits with UCB-based Exploration* proposes an algorithm names NeuralUCB that uses a NN based feature mapping to construct the UCB. The reward function is learned using a fully connected neural network with depth $L \geq 2$. The key idea of NeuralUCB is to use a neural network $f(x;\theta)$ to predict the reward of context *x* and upper confidence bounds. They have experimented on both synthetic and real-world datasets achieving promising empirical results. [34]

B. Probability matching

In contrast with the methods described that are deterministic, the algorithms from this section are constructed in a probabilistic manner to deal with selecting an arm based on its probability of being the best arm. As described in the

section on context-free bandits, Thompson Sampling (TS) is a fairly old algorithm that has once again spiked the attention of the researches, adapting it to the contextual setting. Most of the algorithm in this section are based on TS.

1) Logistic Thompson Sampling: Thompson sampling is an important decision making strategy, observed to reach state-of-the-art results, acting better than some other strategies such as UCB. Thompson Sampling with logistic regression is described by Chapelle and Li [20]. The authors argue that because of its simple heuristics and ease of implementation this is one of standard baselines to compare against. Let D be set of past observations, made of triplets (x_i, a_i, r_i), with x_i as context, a_i as action and r_i as reward. They are modeled using a parametric likelihood function $P(r|a, x,\theta)$, where θ are some unknown parameters [20]. If we know the prior distribution on these parameters, we can compute the posterior distribution on them by Bayesian rule.

The algorithm works by receiving the context vector x_t and draws θ^t form the probability $P(\theta|D)$ then selects the arm that maximizes $\mathrm{E_r}(r|x_t, a, \theta^t)$, observes reward, updates history and start over.

Each weight is drawn independently according to its Gaussian posterior approximation. Logistic regression is used to learn the unknown weight vector θ. One of the experiments was to predict the CTR in an advertising display experiment. In a simulated environment, a feature vector for every (context, ad) pair was created and the clicks were simulated using a weight vector θ^*. The click for every ad (arm a) was generated with the probability $(1 + exp(-x^{\mathsf{T}}, \theta_a))^{-1}$. Thompson sampling proves to have the lower regret out of all tested algorithms [20]. Another experiment is conducted on article recommendation in personalized news article recommendation on Yahoo! front page. They use logistic regression to model the article CTRs. A feature vector x of dimension 21 is used and a weight vector θ_a with the same dimension needs to be learned. Thompson sampling appears competitive this time as well [20].

2) Thompson Sampling for Contextual Bandits with Linear Payoff: This algorithm attempts to generalize of Thompson Sampling for the stochastic contextual multi-armed bandit problem with linear payoff functions [35]. The contexts in this case are produced by an adversary after observing the played arms and their rewards. The authors assume some sort of a unknown underlying parameter $\theta \in \mathbf{R}$ where the expected reward for each arm i with the context x_i is $x_i^{\mathsf{T}}\theta$. They use Gaussian likelihood function and Gaussian prior to construct the algorithm. At every step t, the algorithm generates a sample θ_t from the distribution and picks the arm that maximizes $x_i(t)^{\mathsf{T}}\theta_t$. The algorithm and all the demonstrations are succinctly explained in the paper [35].

3) Bootstrapped approaches: As in the case of Bootstrapped UCB, the used re-samples are drawn from the same distribution as the original samples. In essence, the bootstrap uses the empirical distribution of a sampled dataset as an estimate of the population statistic [36]. Cortes in [27] constructs 2 Thompson Sampling (TS) based bootstrapped algorithms with the offline and online version that take as inputs a number of resamples and oracles and select the action that maximizes the reward. The offline example takes a bootstrapped resample of context and rewards and refits the oracle to this resample; while the online paradigm takes a sample observation weight w and updates the oracle with the new observation of context and reward weighted on w.

Osband and Roy propose a bootstraped TS for deep exploration that pulls the arm with the highest bootstrap mean, which is estimated from a history with pseudo rewards. Their algorithm is similar to Thompson Sampling, though the posterior sampling step has been replaced by a single bootstrap sample [36].

Personalized Recommendation via Parameter-Free Contextual Bandits is a bandit policy with no exploration parameters, that uses a bootstrapping technique based on principled resampling called online bootstrap [37]. The recommendations are created by randomly sampling the model's coefficient vectors from a derived distribution (bootstrap replications). They apply their algorithm to the Yahoo! Today news and KDD Cup 2012 Online Ads, evaluating it by the *replayer* method [38].

4) Neural approaches: Deep bayesian bandits showdown [39] shows an empirical comparative study on bayesian deep networks for Thompson Sampling. The authors propose a series of algorithms based on deep learning that use decision making via TS. They employ several well-studies as well as recently developed methods to approximate posterior distributions, combined them with Thompson Sampling and apply them to contextual bandit problems. There are 10 groups of algorithms: Linear Methods, Neural Linear, Neural Greedy, Variational Inference, Expectation-Propagation, Dropout, Monte Carlo, Bootstrap, Direct Noise Injection, Bayesian Non-parametric. The architecture of the neural networks in all algorithms is the same: a simple fully-connected feed-forward network with 2 hidden layers with 100 units each and ReLu activations. The input of the network has dimension d (same as the contexts), and there are k outputs, one per action [39]. They found that robust methods approximate the uncertainty and representations faster than more sophisticated approaches that require heavier training. We encourage the reader to refer to the cited paper for a detailed explanation of each of the approaches.

C. More contextual bandits

1) Randomized UCB: We have to mention here that some algorithms briefly that are often cited in the literature, but are (yet) too complicated to be optimally implemented. One of such algorithm is Randomized UCB [40]. The algorithm could achieve optimal regret while having a polylog(N) running time. The key insight boils down to a game between an adversary and the algorithm. The distribution over action is constructed as a distribution via policies. However, according to [41] this algorithm is not yet practical to implement.

2) ILOVETOCONBANDITS: Based on Randomized UCB, the researchers from Microsoft propose an alternative algorithm based on coordinate descent, called Importance-weighted LOw-Variance Epoch-Timed Oracleized CONtextual BANDITS algorithm (ILOVETOCONBANDITS) [42]: in each iteration, the algorithm calls the optimization oracle to obtain a policy; the output is a sparse distribution over these policies. The algorithm begins with a initial distribution of policies Q_1 over the policies π. For each t= 1,2...T, the context x_t is observed. The distribution p_t over actions 1,2,...,k is computed based on Q_t and x_t. An action a_t is drawn from p_t

and the reward $r_t|a_t$ is collected. Then create the importance weighted reward estimates $\hat{r}_t$. Compute sparse Q_{t+1} using the coordinate descent to get sparse policy distribution that balances explore/exploit.

3) The Epoch Greedy Algorithm for Contextual Bandits: Proposed in 2008 by Langford and Zhang, the Epoch Greedy Algorithm is based on balancing the exploration and exploitation to receive a small overall regret in a time horizon T. One important problem the algorithm has to solve is to decide when to explore and when to exploit [26]. The authors find useful to first perform a first stage of exploration steps, followed by a second stage of exploitation steps. In the first step, the algorithm samples an arm uniformly at random. In an exploitation step, it selects the arm based on the best policy learned from the exploration samples, using an unbiased estimator [28]. The algorithm finds the optimal arm by solving

$$\max_{h \in \mathcal{H}} \sum_{(x,a,r_a) \in W_l} r_a I(h(x) = a) \tag{13}$$

and attribute $a_t = \hat{h}_l(x_t)$, where $\hat{h}_l \in \mathcal{H}$ is the best hypothesis, r_a is the reward for action a; $I(.)$ is the identity function and x_t is the context at moment t for the action/arm a_t. This algorithm is deemed to have a regret bound incurred by complexity bound for a hypothesis class, scaling as $O(T^{2/3}S^{1/3})$ or better. No knowledge of a time horizon T is necessary.

V. Different Flavors of MABs in RecSys

In order to adapt the MAB framework to the task of recommendations, many researchers understood that these frameworks have to employ various strategies to align bandits to real-world scenarios. A plethora techniques and ideas were developed to make bandits suitable for integration in different RecSys. In this chapter we will like to describe various techniques and research directions that extend the MAB framework and show how other directions of study can be leveraged to improve bandits' possibilities.

A. Multiple Play Bandits

Usually MAB focus on selecting only 1 arm from the available decision space. Sometimes, for some application this is not enough. The decision system should be capable to select multiple arms that correspond to the best candidates. There have been some algorithm that extend the traditional bandits to multiple actions (or multiple plays), allowing the algorithm to choose not one but many arms in a fashion that has sound theoretical foundation.

1) EXP3.M: This is an adversarial algorithm that studies the problem with multiple plays on K arms [43]. Just like in the 1-arm selection case, the algorithm selects an action i with the probability $p_i(t)$ and computes an estimation of reward $r_i(t)$ as $\widehat{r_i}(t) = r_i(t)/p_i(t)$ if the action i is indeed selected and as $\widehat{r_i}(t) = 0$ otherwise. Now the main problem is reduced to selecting the best m arms out of K.The algorithm is based on the online mirror descent principle to minimize regret in online convex optimization. Louedec et al in *A Multiple-play Bandit Algorithm Applied to Recommender Systems* [44] propose a computationally efficient implementation of the same algorithm and test it on movielenas and Jester datasets and prove empirically its performance.

2) CBMA: The Contextual Bandit with Multiple Actions, (CBMA), proposed by Chang and Lin, uses pairwise ridge regression with upper confidence bound to select multiple actions. The key idea is to choose an action as a (pairwise) ranking task and solve it by transforming it into a pairwise linear regression problem [45]. The approach assumes T iterations, and for each of it the environment provides a context matrix X_t; m actions are chosen from the K candidate pool an the reward vector r_t is revealed; the algorithm is updated with information about this iteration. The actions are scored, and the action with the best score is played. CBMA uses the PairUCB - the proposed algorithm that deals with computing the unknown weight vector θ (called pairwise reward estimator) using pairwise ridge regression between the the pairwise context matrix and the as the pairwise reward vector [45]. The algorithm was tested on real-world and synthetic data and has shown better results than LinUCB, being also less susceptible to the α parameter.

3) MP-TS: A Multiple Play Thompson Sampling (MP-TS) approach was also proposed for the problem of multiple selections and was presented in [46]. The algorithm reassembles the classical TS algorithm, with the exception that instead of choosing the best arm based on the posterior sample $\theta_i(t)$, it chooses the top L-best arms from the candidate pool. In order to ensure a low regret bound, the algorithm suppresses the simultaneous draw of 2 or more sub-optimal arms. They also propose an enhanced version of the algorithm, which instead of L arms, exploited L-1 arms, which proved good empirical results. The MP-TS algorithm performs better than the earlier described EXP3.M and the *Combinatorial multi-armed bandit* proposed by Chen et al in 2013. The experiments were conducted on the KDD Cup 3 2012 dataset for online advertisement based CTRs optimization and other 2 small synthetic models.

B. Multi-objective optimization in MABs

Sometimes the system might have demands to optimize for different final objectives. If the majority of examples provided in this paper relied on optimizing the CTR, this might not always be the case. If we have a multi-stakeholder platform (let's say we want to optimize for the reader and for the publisher on the same platform), we have to consider bandits that can optimize multiple objectives. In this subchapter we will show some the multi-objective optimization approaches.

1) Multi-objective Bandits: Optimizing the Generalized Gini Index: The authors formalize the problem of optimizing a bandit with multiple objectives, where agent receives a D-dimension vectorial feedback that encodes many possibly competing objectives to be optimized [47]. The main objective of the paper is to find a fine line between optimizing all the objectives and yet taking into consideration the balance between this optimizations. Instead of using a Pareto front, the authors employ the Generalized Gini Index [48]. In this setting, the cost of each arm is no longer a scalar value, but a vector. In order to find the best arms, they are compared by their GGI, where the optimal arm minimizes the Gini score. The optimization algorithm is based on the Online Gradient Descent.

2) Bandit based Optimization of Multiple Objectives on a Music Streaming Platform: Recommender systems powering online multi-stakeholder platforms have to take into

consideration optimizing the recommendations towards more than one direction. This paper discusses how Spotify tries to optimize their recommendation in order to satisfy the user, the artist and the platform. This translates into optimizing more than one single objective (for example they need to take care not only of the click rate, but also of the time listened, the artist displayed, streaming time etc) [3]. In order to deal with this task, the authors formalize the problem as a contextual multi-armed bandit with multiple objectives, where for each round t a set of features $\mathcal{F}_t$ of length M is associated with every arm. Now under a linear presumption, the reward r_t at time t after pulling the arm k is

$$r_t = \mathcal{F}_{t,k}^{\mathsf{T}} \vartheta^* + \zeta_t \tag{14}$$

where $\vartheta^* \in \mathbf{R^{D \times M}}$ is a fixed unknown param and $\zeta_t \in \mathbf{R^D}$ is the independent random noise for each of the objectives. For selecting the optimal arm, instead of choosing a solution from the Pareto front, the authors use the Generalized Gini Index based aggregation function. The proposed algorithm is called MO-LinCB that optimizes the regret for an aggregation function. The algorithm was tested on synthetic data and a real Spotify dataset, compared with a baselines including greedy approaches, MO approaches (MO-ODGE described above) and simple selection strategies. The algorithm performs better in terms of optimizing all the objectives and also in regards to time constrains. An important finding of this papaer is that multi-objective models perform better than the single objective case, across all metrics [3].

3) : Other studies [49] propose multi-objective multi-armed bandit that use Thompson Sampling to pick weights that identify new arms on the Pareto front with higher frequency; explore bi-objective problems with one dominant objective [50] or consider a problem where the agent uses a linear scalarization function in order to transforms a multi-objective problem into a standard problem by summing the weighted objectives [51].

C. Clustering of bandits

Bandits were also studies in terms of clustering. Researchers found out that it can be useful to cluster bandits, arms or user features in order to reduce the dimensionality of the problem, while maintaining a promising regret bound. Here are some interesting approaches in bandit clustering.

1) Multi-armed Bandit Problems with Dependent Arms: This approach focuses on clustering arms of the bandit based on some dependencies among them. The assumption is that the arms are *dependent* [11]. The problem is formulated in terms of N arms that grouped into K known clusters with the objective to maximize the finite-time reward or minimize the expected regret. Considering π_i the parameter set for cluster C_i, every time an arm i is pulled it alters $\pi_i(t)$ simultaneously for all arms in cluster C. At every time step the policy computes an (index, arm) pair for every cluster. It calculates the rewards and the variance estimate for each cluster, calls the second policy to select the cluster and inside the cluster calls the policy again to select the arm. This approach was called the Two Level Policy (TLP) [11]. The experiments were conducted on synthetic data and an ad-placement campaign dataset, using different instances of the TLP and UCB as baseline. TLP gave better empirical results and the authors conclude that the regret depends on the characteristic of the clusters.

2) CLUB: Online Clustering of Bandits The idea behind this concept is that the system has to serve recommendations to a N users that can be grouped on C clusters, based on their feedback on recommendations [52]. It relevant to group recommendations. Each cluster C_j is parametrized by some unknown vector $c_j \in \mathbf{R}^D$, shared by every user in C_j. The authors assume one linear bandit per node inside each cluster and one linear bandit per cluster. Each node i hold a proxy vector w_i, while the cluster j holds a proxy vector z_i, and z_i is an aggregation of w_i. Nodes are served in sequential order. They start with the big cluster of n elements and delete edges were the proxy vectors are too different. After the user i is served in cluster j, the proxy vectors of the user and cluster are updated. The algorithm was tested on synthetic datasets, as well as LastFM and Delicious and Yahoo! datasets, competing with 2 variations on LinUCB (that allocates an instance of the algorithm across all users and one that allocates a instance of the algorithm to each user) and proved better results than its competitors.

3) DynUCB: Dynamic Clustering of Contextual Multi-Armed Bandits The main idea behind this concept is to divide the users into different clusters and customize the bandits for each cluster. Also, the clustering is deemed to be dynamic, meaning that the users can shift from one cluster to another, which illustrates the real-world scenario of preference-shifting. DynUCB is based on LinUCB, taking as input the desired number of clusters K. For each user u, associated with the cluster C_k at time t, the algorithm learns the coefficient θ_k from the cluster-level parameters of the bandit $\mathbf{b_k}$ and the matrix $\mathbf{M_k}$. For the action a the generated reward is $\left(\hat{\theta}_k^{\mathsf{T}} x_{t,a} + \alpha\sqrt{x_{t,a}^{\mathsf{T}} \hat{\mathbf{M}}_a^{-1} x_{t,a} \log(t+1)}\right)$ where θ_k is a cluster level parameter for a user u in cluster C_k and x_t is the context vector of the action a [53]. The reward is used to update the user's own coefficient θ_{u_t} and re-assign user u_t to the cluster C_k which has the cluster coefficient $\widehat{\theta}_{k'}$ closest to its own.The experiments were conducted on Delicious and LastFM data against the baseline algorithm CLUB (described earlier) and the 2 variations of LinUCB just as previous and has seen significant better results on the Delicious dataset [53].

D. Collaborative Filtering Approaches

Collaborative filtering that is typically used in RecSys can be combined with bandit usage. The next couple of example illustrate these implementations.

1) COFIBA: Collaborative Filtering Bandits (COFIBA) is at a crossroads of clustering and collaborative filtering and and takes advantages of preference patterns in the data. COFIBA is an upper-confidence-based heuristic, combined with adaptive clustering procedures as for items and users [10]. It explores the collaborative effect induced by the shiftings in user/item interaction. This collaborative component means that the users are dynamically clustered in groups based on the surveyed items, while, at the same time the items are clustered based on the clustering similarity induced over users. This is the first step of the algorithm. Then, it explores the priori of items (denoted by their feature vectors) $I = \{x_1, x_2, .., x_{|I|}\}$ and performs multiple clustering over the set of U of users and a single clustering over the set I of items. The algorithm then computes the neighborhood sets $N_{i,t}(x_{i,t})$ with respect to the items in $C_{i,t}$

and are stored into clusters from the users side (pointed by these items). The clusters are updated on user and item side. The algorithm was tested on 3 datasets and yielded better results than over surveyed algorithms [10].

2) FactorUCB: The approach assumes similarity of users incorporated through a weighted graph by completing a low-rank matrix of user-item interaction and employs a UCB selection strategy [54]. A factorization-based bandit algorithm is placed on every user. One of the main questions in matrix factorization is how to choose the next user-item feedback pair for model update so that it will be less susceptible to bias. Exploration of optimal items become therefore a good option to ensure more optimality; in order to do that the authors employ the UCB strategy and assume that the observed reward from each user is determined by the user's neighbors. The best item is selected based on the contextual features(known and observable) and latent item factors, as well as a matrix Θ storing latent user factors. In other words, the best item is selected according to the predicted expected reward based on the current estimation of latent user and item factors. Experiments were conducted on synthetic data, as well as on Yahoo! and LastFM datasets and compared to several state-of-the-art factorization-based and bandit-based collaborative filtering methods and proved that the leveraged contextual features and user interactions help conquer the cold-start problem [54].

3) PTS: Particle Thompson Sampling for Matrix Factorization In this paper the RecSys delivers an item recommendation to the user using Thompson Sampling. The user then rates the item and the system updates the statistics. The original thing here is that the systems updates not only the latent matrix of the user, but also the latent representation of the item in a matrix factorization model based on the rating of the item [55]. In order to update the matrix in real time, the system uses the a Rao-Blackwellized particle filter for online matrix factorization. Posterior density is estimated by particle filtering(with a set of weighted samples) [55]. The system was evaluated on 1 synthetic and 5 real world datasets. It yielded good performance applied to real world problems. The approach was patented by Adobe.[1]

E. Time bound bandits

Some of the approaches consider the time aspect of the context(think about placing adds or videos that have an expiration date), items, users and user/item interactions that change over time even in terms of the rewards received (change in consumption preference). There are some approaches that deal with the induced time-dynamic aspects of multi-armed bandits.

1) Mortal bandits: The idea behind this approach is that arms have a lifetime after which they expire and the algorithm needs to explore constantly [56]. There are 2 scenarios proposed: a less realistic scenario called *state-aware*, where the reward is also the best reward possible for that arm, and the *state-oblivious*, with stochastic rewards, thus a more realistic scenario. The authors model the mortality of the bandits in 2 ways, whether the arm lives after being picked or not. Because of the way the rewards are assumed, the regret in the mortal bandit can never go to zero (in typical MABs it is possible). Considering this setting, there are separate algorithms developed for the *state-aware* and the *state-oblivious* cases, which are based on the same assumptions. The algorithm proves optimal in the setting of deterministic rewards: given a distribution and an expected live span, compute the expected reward for an arm r_i; select a random arm; if the reward of the arm is bigger than the computed reward, stay with the new arm every turn until it expires. The only difference between this algorithm and the stochastic one is that instead of pulling an arm one time and determine its reward r_i, the algorithm pulls each arm *n* times and abandons if it looks unpromising. The paper uses an epoch based (with a subset of *k*/*c* arms per epoch) UCB and epsilon-greedy with mortal bandits, but it is important to note that any search heuristic can be plugged into the mortal bandit. The empirical evaluation is performed on 3 different distributions of arm payoffs.

2) Time Varying Bandit: Online Context-Aware Recommendation with Time Varying Multi-Armed Bandit This paper focuses on capturing the time dependent changes of the reward in contextual MAB problems. The authors propose a dynamic setting for the context and model the dynamic behavior of the reward as a set of random walk particles. An arm's particle is a container which stores the current status information that arm [57]. The learned parameters are integrated into a bandit selection strategy in order to serve recommendations. The reward of the arm a is modeled as a linear combination of the contextual features x_t and the coefficient vector θ_t. In order to count for the drift in the context, the authors assume that x_t is comprised of a constant and a drift component, where the stationary c_t is generated with a conjugate prior distribution and the drift component is element wise product of a standard Gaussian random walk $rand_{t,k}$ and a parameter μ_k. After getting a score for arm a given context x_t, the algorithm learns the parameters based on all particles' inferences of μ_k and computes a score based on that parameters; then, the inference is updated and weights are computed for each particle; particles are re-sampled according to the weights and all the statistics are updated. The algorithm is evaluating using simulation and the *replayer* method described in VI-A1 on KDD Cup 2012 online ads data and Yahoo! Today News.

F. Ensemble methods

In the last section of this chapter we will discuss some ensemble methods with bandits for recommendations. Ensemble in this case refers to aggregating multiple policies of bandit algorithms, rather than learning a unified predictive model. An ensemble method combines the prediction of different algorithms to obtain a final prediction [58].

1) HyperTS and HyperTSFB: Ensemble Contextual Bandits for Personalized Recommendation This approach employs a meta-bandit paradigm that constructs a hyper-bandit that coordinates the work of some other base bandits and observe their importance in terms of explore/exploit depending on the user feedback [37]. Because of the nature of the ensemble strategy that estimates the performance of each policy, this strategy proves a very good for solving the cold-start problem. The idea of the ensemble is to have a set of action $\mathcal{A}$ and a set of policies $\Pi = \{\pi_1, ..., \pi_n\}$ and to find a policy that is closest to π^*, where $\pi^* = \arg\max_{\pi_i \in \Pi} \mathrm{E}[r_{\pi_i}]$. The paper presents 2 algorithms HyperTS and HyperTSFB.

[1] https://patents.google.com/patent/US10332015

HyperTS estimates the expected reward of each policy $\pi \in \Pi$ using the Monte Carlo method. The context $x_1, ..., x_n$ is drawn from a distribution in which the policy π_i is selected. For an input context x_j, π_i plays the arm a_j and obtains the reward r_j. Considering all the distributions, the Monte Carlo estimate becomes $\hat{E}[r_{\pi_i}] \sim \text{Beta}(1 + \alpha_{\pi_i}, 1 + \beta_{\pi_i})$, where $\alpha_{\pi_i} = \sum r_j$ and $\beta\pi_i = n - \alpha_{\pi_i}$. In each trial r_i is a sample from $\hat{E}[r_{\pi_i}]$ drawn from the Beta distribution. The final policy that get selected is the one with the highest r_i. Continuing, HyperTSFB improves the estimation efficiency of HyperTS by fully utilizing every received feedback for expected reward estimation [37]. Evaluation is performed on Yahoo! Today News and KDD Cup 2012 Track 2 online advertising data using the replay method. The policies are compared against with Random, epsilon-greedy, LinUCB, Softmax, Epoch-greedy; Thomspon Sampling is used as base policy. HyperTS and HyperTSFB are able to achieve comparable performance with the top ranked base policies.

2) BEER: Ensemble Recommendations via Thompson Sampling: an Experimental Study within e-Commerce This approach extends Thompson Sampling in order to orchestrate a set of base RecSys for e-commerce. It takes into consideration realistic settings and implements TS when neither action availability nor reward stationary is guaranteed [59]. This approach is similar to the one proposed by [37] in the manner that each bandit arm represents a recommendation strategy to match the user's query to items in the database, but is different in terms of how the authors model the arms and attribute rewards. The arms, that are simple contextual recommendations, are partitioned into disjoint groups using a IDF (Inverse Document Frequency)-based partitioning. The proposed bandit ensemble framework BEER(Bandit Ensemble for E-commerce Recommendations) is used with Thompson Sampling on top of this partitioning method. As the authors suggest, the recommendation components of the ensemble can be created automatically from the existing item attributes or user-item interaction history. This type of architecture achieves scalability because the number of arms in not dependent on the number of items [59].

a) : Another approach we will briefly like to mention here is called *Prediction Model Selection* (PdMS) that tackles the problem of cold-start users and relies on feedback to pick the most appropriate recommendation. The arms are considered as clusters of users computed using matrix factorization and employ UCB and epsilon-greedy as selection policies [14]. There are a myriad of additional implementations of MABs used in RecSys. We described only some of them, but decided to mention a couple more for the reader to further investigate. Thompson Sampling is highly used as a policy for creating recommendations and can be found in document labeling as Active Thompson Sampling [60]; search query recommendations using an M-Independent arm Thompson sampling [61] and tutoring systems [62].

VI. Evaluating bandits

Evaluating multi-armed bandits is difficult [20]. Bandits prove to be not easy to work with and can be difficult to evaluate because of the online learning paradigm that they were created for. With that said, it would be highly unwise to deploy a method to production without evaluating it in any way. There are some established way to evaluate the bandits in an *offline* fashion and in pseudo-online fashion in simulated environments. In the next sub-chapters we will discuss these approaches.

A. Offline Evaluation

1) Replay Method: Many papers cites in this study have been using the replay method to evaluate their bandits [3], [52], [10], [37], [32], [54], [57] and others. *Unbiased Offline Evaluation of Contextual-bandit-based News Article Recommendation Algorithms* [38] or from now on the *replayer* method was proposed as a data-driven and proven unbiased offline evaluation of bandit algorithm and became one of the industry's standards. The technique assumes that the arms are i.i.d. and that each arm is chosen uniformly at random by the policy at each step. The crux idea of this approach is that having access to historical data, in the evaluation process, the algorithm will only consider those actions that selected the same item as in the historical dataset and discarding the rest. As noted in [57]: "The main idea of replayer is to replay each user visit to the algorithm under evaluation". Formally, if the current policy A chooses the same arm as the logging policy L, than the event is retained and the statistics are updates, otherwise, the event is discarded.

One downside of this method is that it needed a very large initial historical set, especially if there are many arms to choose from, because the number of discarded event will rise significantly. If you have k arms and T samples, than the size of the new dataset will be T/k.

2) Bootstrapped evaluation of contextual bandit algorithms: This approach assumes a setting called dynamic recommendation, where only a couple tens of items are available at any given moment. Based on the replayer method, but realising its short-comings, this method uses bootstrap in order to generate the datasets used for evaluating, more precisely: from a dataset of size T with K possible choices of action at each step, they generate B datasets of size K/T using sampling by replacement with a non-parametric bootstrap procedure [63]. For the newly created data subsets the classic replayer method is used to estimate the per-trial payoff. The authors deduce that because of this, the algorithm A is evaluated on T records on average. The authors compared their method with replay and concluded that it converges faster and gets higher accuracy on real world datasets.

3) Direct Method: This method, proposed by Beygelzimer and Langford, employs using a regression model in order to learn the reward function. The reward is conditioned by the actions and the context. The method allows one to reuse of any existing, fully supervised binary classification algorithm in a partial information setting [64]. The value of the policy is given by:

$$\hat{V}_{DM}(\pi) = \frac{1}{n} \sum_{k=1}^{n} \sum_{a \in \mathcal{A}} \pi(a|x_k)(\hat{r}(x_k, a)) \qquad (15)$$

with $\hat{r}(x, a)$ being the approximation of the reward and π the policy in question.

4) Doubly Robust Policy Evaluation and Optimization: According to the authors, the *doubly robust* method is a type of statistical approximation from incomplete data, where if either one of the two estimators (for example Direct Method-estimated rewards from given data or Inverse Propensity score - uses importance weighting to correct the shifts and biases in historic data) is correct, then the estimation

is unbiased. The approach suggests a replay-based non-stationary policy evaluator, with an improved acceptance rate (as discussed earlier, discarding a large part of the dataset is one limitation of the replay policy).

$$\hat{V}_{DR}(\pi) = \frac{1}{n}\sum_{k=1}^{n}\left[\hat{r}(x_k, \mathbf{v}) + \frac{\mathbf{v}(x_k|a_k)}{\hat{\mu}(x_k|a_k)} \times (r_k - \hat{r}(x_k, a_k))\right] \tag{16}$$

where

$$\hat{r}(x_k, \mathbf{v}) = \sum_{a \in \mathcal{A}} \mathbf{v}(a|x)(\hat{r}(x, a))$$

is the estimated value of the policy derived from $\hat{r}$ [65]. Doubly robust policy uses the estimated mean reward function to decrease variance and a weighted version of rewards [66]. In some way, DR combines the ideas from other two estimators - Direct Method and Inverse Probability Weighting.

B. Simulated environments

Another way to test MAB is by creating simulations of the online settings. For example, in order to test their algorithm [20] used simulated clicks,included in some vector w^*, and generated with a certain probability. The simulation method is considered to give better results *replayer* when the available collection of items to recommend is large [57]. Simulations and synthetic data are used in many of the cited papers. With that said, creating simulator can be both tedious, time-consuming, non-trivial and at the same time may not reflect actual performance [38].

Open AI offers different synthetic environments to build and test Reinforcement Learning algorithm, meaning that they can also be compatible for bandits (open AI Gym for example, offers a simple yet versatile setting for creating such synthetic fields)[2]. More recently, Tensorflow developed TF-agent bandit library, that offers environments for different setups (for linear and non linear reward functions; stationary and non-stationary dynamics) and many more advanced functionalities[3].

VII. Multi-armed bandit recommendation systems used in industry

In this sub chapter we would like to show how different companies used the multi-armed bandit framework in order to solve different challenges and present personalized content to the users. It is interesting to see how various problems were modeled as MAB problems, in different ways and under different assumptions.

A. Amazon

The paper *An Efficient Bandit Algorithm for Realtime Multivariate Optimization* tackles the problem of page layout optimization at Amazon by using a parametric Bayesian model that explicitly incorporates interactions between components of a page, applying a hill climbing optimization algorithm to approximate a contextual Thompson Sampling. Deployed to production, the solution increased conversion rates with 21% on certain Amazon landing pages [67]. Each possible layout is comprised of D different widgets. If the each widget has N variations, than the stochastic MAB problem is formulated as having N^D arms on $t = 1, 2, .., T$ timesteps with a feature vector b_t (user or session information) and a arm feature vector a_k (representing to the layout). The final feature vector $x_{a,t}$ is generated from user feature vector and arm feature vector. The reward is given by a linear scaling of $x_{a,t}$ over a unknown weight vector θ. The algorithm selects the optimal content, but also is responsible for contextualization and personalization of the page's layout. They have simulated the experiment on 512 different layouts comparing the different flavours of the algorithm versus non-contextual multi-armed bandit with N^D arms and non-contextual N-armed bandit for each of D widgets. The method proved to be suitable for capturing the interactions between the content pieces that are displayed together, while taking into consideration the context.

B. Netflix

An interesting way in usage of bandits can be seen at Netflix. According to Netflix's technology blog, the streaming giant uses bandits to select the personalized artwork for each movie it presents to the user; the authors claim on using contextual bandits in order to pick the most suitable image from a dozen of candidates, specifically for that particular user [68]. In order to create the context for the bandit, the authors use information as: the titles played by the user, the genres those titles are attributed to, past interactions of the user with different titles (if available), demographics etc. Then the model ranks images for each context, by predicting the probability of play for each image accompanying a given title and the one with the highest probability gets chosen. The evaluation of the bandit is done offline, by using the *replay* method VI-A1. At the DataEngCong in San Francisco in 2018, Netflix presented a framework of MABs that they use in order to get personalised homepage recommendations for each member [1]. The system can be broken down into the offline(attribution assignment, model training) and online components (use explore/exploit policy, log contextual information, generate and serve recommendations). The contextual information about each decision is being stored, in order to understand further on why that exact recommendation has been made. The main idea for the billboard is to try to maximize the incremental probability of play, where this probability is computed as the difference of probability of play of when the title was shown on the billboard vs when it was absent. The researchers noticed that implementing incrementality is able to shift engagement within the candidate pool (from popular titles to less popular).

C. Spotify

Spotify has a mechanism in their recommendations that they call "recsplanations" - making the user to understand their recommendations and provides a method that combines the explanations of the recommendations and the bandit algorithms behind them in an approach called BART - BAndits for Recsplanations as Treatments in a paper called *Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits*. The paper provides the first method that combines bandits and explanations in a

[2]https://gym.openai.com/

[3]http://github.com/tensorflow/agents/tree/master/tf_agents/bandit

principled manner [2]. BART chooses as context and reward model that imposes a linearity on the reward r:

$$r(j, e, x) = \sigma(\theta_{\text{global}} + \theta^{\mathsf{T}} x')$$

where $x' = [1_j^{\mathsf{T}}, 1_e^{\mathsf{T}}, x^{\mathsf{T}}]^{\mathsf{T}}$ is the aggregated context containing information about item(j), explanation(e) and context(x) with 1_j being a 1 hot encoded vector with a single 1 at index j; $\theta = [\theta_j^{\mathsf{T}}, \theta_e^{\mathsf{T}}, \theta_x^{\mathsf{T}},]^{\mathsf{T}}$, with θ being a coefficient in logistic regression. To overcome the disadvantage of logistic regression in recommendations (the choice of the item and explanation is disregarding the user's attributes) and to get more personalized recommendations, the authors introduced a weighted sums of higher order interactions between elements in the aggregated context vector [2]. Then they used a use counterfactual risk minimization (CRM) for bandit training. Finally, they use an epsilon-greedy exploration-exploitation approach. BART uses conditional exploration, meaning it decides if it's worth exploring or exploiting the items apart from the explanations, while keeping the underlying model for the reward the same. Also, experiments with multi-objective optimizations bandits were conducted at Spotify; for more details see V-B2.

D. Expedia Group

Expedia uses multi-armed bandits in a production setting, where the arms of the MAB are the recommender system's variants. There are 4 different variants of the RecSys: the first uses an adaption of content based recommender system(works based on top key item features); the second includes a variant of probabilistic recommender system; the third uses session based embeddings, while the last algorithm (arm of the MAB) makes use of a Matrix Factorization model. They use click-through-rate and conversion rate as performance metrics. The system receives the statistics on views and clicks for each variant and stores the traffic quotas each armed received to some cache [69]. They use Thompson Sampling for this traffic allocation and run it in mini-batch daily runs. They compare the performance of the new variants (expressed on CTR) with the baseline(the model that is currently in production.)

E. Zozotown

Zozo is the largest Japanese fashion e-commerce company. In a recently released work, *A Large-scale Open Dataset for Bandit Algorithms*, the authors reveal that they have been using Thompson Sampling along with a Random selection policy in order to present recommendations to their customers. The most important input of this paper is that the authors propose an open bandit pipeline (OBP) to facilitate scalable and reproducible research on bandit algorithms [66]. They implement some of the most famous bandit policies with a bunch of policy estimators, along with a open dataset of subsets of different sizes.[4]. Besides the classical *replay* estimator, the authors implement and compare some other estimators from the literature (Direct Method by [64], Inverse Probability Weighting [70], Doubly Robust [65]) and conclude that finding a stable and optimal evaluator is as important as selecting the best performing policy. The work is important because if offers a pipeline implementation of bandits and estimators that can be extended with own implementations and applied to different datasets.

F. Yahoo!

There are many experiments at Yahoo! using the MAB framework [38], [4]. Another one is presented in *A Batched Multi-Armed Bandit Approach to News Headline Testing*, that uses Thompson Sampling to choose from different versions of titles, in order to pick the one that will maximize the click prospect. There are K different headline variants, each associated with one arm. The idea is to find the optimal arm that maximizes the CTR, but also to considerate the perish of the articles, which may lead to changes in click gains. The authors introduce batched Thompson Sampling(bTS) [5] that is tuned for optimal feedback when the user's feedback is processed in batches. The batched idea was considered because of the massive traffic on the Yahoo! front page - it would have been an immense burden to update the algorithm after each user interaction. The idea behind the bTS is that within each batch, the Beta distribution of each arm remains the same. The traffic across arms is allocated based on their random Beta distribution samples drawn for each incoming view event. The statistics are updated at the end of each run. There are 2 ways of updating: the *summation update*(raw counts of clicks) and *normalization update*(number of normalized clicks and non-clicks). Three moments need to be tuned in bTS: the algorithm stopping point, the beforementioned update methods and the update frequency. The benefits of the bTS are at follows: it exposes much fewer suboptimal headlines, thus increasing user experience; it helps distinguish the optimal arm faster(having more samples to compare among the good arms) and it gained a 3.69% overall click gain.

G. Deezer

For the RecSys 2020 conference, Deezer presented their usage of bandits in recommending playlists in a carousel (the user is recommended a list of L swipe-able items, from which only 3 are visible without scrolling) fashion [71]. The approach considers a semi-personalized recommendation setting, with number of Q clusters on N users, where $Q << N$. At each round, a random subset of users is presented a policy that updates its model based on the users' feedback. The policies have to recommend a list $L = 12$ playlists. The policies used in the experiment are ϵ-greedy with segmentation; an explore then commit strategy; KL-UCB (based on Kullback-Leibler divergence); Thompson Sampling and Linear Thomspon Sampling against a random strategy baseline. The experiment was conducted on a sample of over 900K users with 862 playlist[5] and then implemented in the live Deezer app by an online A/B test. One of the impressive results was that semi-personalized recommendations perform as good as as fully-personalized contextual models in terms of playlist ranking, assuming a good initial clustering [71].

VIII. Putting bandits to work

In this section we will showcase the usage usage of some MABs in RecSys by conducting a couple of short experiment

[4]https://github.com/st-tech/zr-obp

[5]The dataset is available at https://zenodo.org/record/4048678#.X22w4pMza3J

on some bandits and datasets. We run some experiments in order to see how some of the described policies perform of different datasets. We are also curious about understanding the importance of using context and observe the trade-off between performance and run time. It si also important to see how much context actually impacts the quality of the algorithm.

A. Comparing different policies

1) Goal of the experiment and included algorithms: We have based our experiment on a number of policies described in the previous chapters of the paper and evaluated them of 2 datasets (a subset of the Open Bandit Pipeline described in section VII-E and Movielens 1 million) using the *replay* method. Classically, we have modeled recommending the best item to the user as a bandit problem and used MABs that tries to maximize the Click-through rate(CRT) of each item. We are interested in finding policy will maximize the total reward (modeled here as CRT) over the items of different datasets. We have evaluated the following policies:

1) Upper Confidence Bound with Ridge Regression (LinUCB)
2) Epsilon Greedy with Ridge and Logistic Regression
3) Thompson Sampling with Online Logistic Regression
4) Epsilon Greedy, UCB, Thompson Sampling with no context for comparing
5) Random policy as baseline

2) Algorithm evaluation metrics: We will use 2 metrics in order to identify the best performing algorithms: cumulative reward of each policy and worst regret (the difference between the sum of rewards yielded by the random policy and the sum of rewards collected by the observed policy). Of course, the best performing policies will have the maximum cumulative reward; at the same time they will also exhibit a bigger difference between the cumulative reward of the random policy and their own. We will evaluate the performance using the *replay* method described in VI-A1. Note that this method should only be used for a small number of arms (not exceeding 50 as noted in [72], [73]). Thus, we will use up to 50 arms, depending on the dataset, as described in the datasets subsection.

3) Baselines: We will compare the algorithms with the random policy and also will be interested in comparing the performance of contextual vs non-contextual models. Now, a couple of remarks should be made here. First, in order to compare apples with apples we ran some warm-up experiments to figure out the best parameters for each policy. Algorithms depend on parameter tuning. We have to observe how these parameters impact the performance of the model; For example, we found out that for epsilon greedy we get the best performance when epsilon is equal to 0.1, ridge UCB also with alpha of 0.25 and in on-line logistic regression for TS the best regression parameter alpha (responsible for the prior distribution of the weights) turned out to be also 1.0 These values can vary in dependence of the dataset, the number of arms and the distribution of weights. Second, we usually compare contextual disjointed models, as described in [4], meaning that we assume that the arms do not have any common features.

4) Datasets: MovieLens 1m Dataset consists of 3 subsections: user data (user ID with corresponding user information

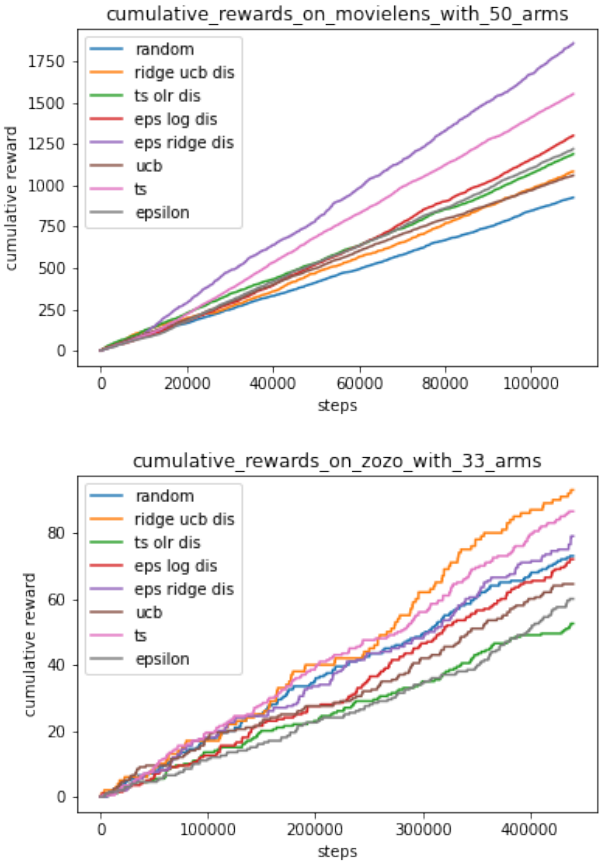

Figure 1. Comparative cumulative rewards of different bandit policies using replay. 1 - Movielens1m; 2- Zozo/men

such as age, gender, occupation, zipcode), movie data (item ID with the corresponding genre), and user-movie ratings (reaction of the user on a movie from 1 to 5). The reward is defined to be 1 if the user's rating for a movie is 5, and 0 otherwise (4 or below).For users, the age group buckets were created. One-hot encodings were created for "age-groups", "gender" and "occupation".For Movies features, it was simply left as one-hot encodings of "genres". Data were filtered to include only top N movies. N will will be set to 50 (so 50 arms).

a) Zozo/men: This is a subset of the datasets offered by Zozo[6], that contains over 1 mln impressions. The dataset is described in [66] and also on the project's github page[7]. We included in the processing only the user and item features and did not take into consideration the position of the item on the page when evaluating. It has 33 different items, thus 33 arms to choose from.

5) Results: We run the experiments on these 2 datasets and evaluated each policy using the replay method, which was suitable for us, as we have chosen a small number of arms. The scope of the experiment was to illustrate the performance of the models on various datasets. It is noticeable that different policies perform differently for various datasets. One important thing to to notice is that on Movielens1m all the policies perform better than the random one, but, at the same time, we can notice that this is not

[6]https://research.zozo.com/data.html
[7]https://github.com/st-tech/zr-obp

always the case for the Zozo data; as we can see the Epsilon Ridge performs better on the Movielens dataset, but the UCB ridge proves to be the best for a the Zozo dataset. One conclusion here is that the structure of the data is important when evaluating the algorithm performance; at the same time it is important to also correctly tune the algorithm to find their best performance. Also find the tables with the result below.

Table I
Comparative results of policies

policy	reward	worst reg	time(s)	dataset	arms	samples
random policy	73	0	13.8	zozo/men	33	440435
ucb no context	64.5	8.5	117.68	zozo/men	33	440435
ucb ridge dis	93	-20	408.89	zozo/men	33	440435
ts no context	86.5	-13.5	80.03	zozo/men	33	440435
ts olr dis	52.5	20.5	584.73	zozo/men	33	440435
epsilon no context	60	13	100.83	zozo/men	33	440435
epsilon ridge dis	79	-6	332.23	zozo/men	33	440435
epsilon logistic dis	72	1	1265.1	zozo/men	33	440435
random policy	927.5	0	3.43	movielens/	50	109804
ucb no context	1061.5	-134	42.19	movielens/	50	109804
ucb ridge dis	1085	-157.5	142.27	movielens/	50	109804
ts no context	1554.5	-627	28.91	movielens/	50	109804
ts olr dis	1191.5	-264	111.96	movielens/	50	109804
epsilon no context	1221.5	-294	32.44	movielens/	50	109804
epsilon ridge dis	1862	-934.5	116.19	movielens/	50	109804
epsilon logistic dis	1303	-375.5	588.11	movielens/	50	109804

IX. Conclusions

In this paper gave an overview on the problem of multi-armed bandits and how bandits are used in recommendation systems. We explained and formalized the MAB problem, giving definitions on the regret and reward concepts that are important in bandit literature. We described the most famous MAB contextual and non-contextual algorithms. We have shown how applications of MABs can be molded into RecSys in different industries and also discussed about different types of advancements in the field. Some of the methods and approaches discussed in this paper were successfully applied and implemented across the industry, while others make a solid theoretical ground for future implementations. However, there are some challenges in the productization of MABs. As [74] notes, determining the correct context vector is critical for contextual bandits; at the same time adding arms on the fly, time-dependencies can also prove problematic. Evaluating bandits is again one of the concerns in the literature. This paper wants to be a survey and a to-go source, without being exhaustive, for those who are interested in getting started with MABs and their applications in recommendations.

References

[1] J. Kawale and E. Chow, "A Multi-Armed Bandit Framework For Recommendations at Netflix," 2018, library Catalog: SlideShare.

[2] J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, and R. Mehrotra, "Explore, Exploit, and Explain: Personalizing Explainable recommendations with Bandits," in *RecSys '18: Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 31–39.

[3] R. Mehrotra, N. Xue, and M. Lalmas, "Bandit based Optimization of Multiple Objectives on a Music Streaming Platform," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Virtual Event CA USA: ACM, Aug. 2020, pp. 3224–3233. [Online]. Available: https://dl.acm.org/doi/10.1145/3394486.3403374

[4] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A Contextual-Bandit Approach to Personalized News Article Recommendation," *Proceedings of the 19th international conference on World wide web - WWW '10*, p. 661, 2010, arXiv: 1003.0146. [Online]. Available: http://arxiv.org/abs/1003.0146

[5] Y. Mao, M. Chen, A. Wagle, M. Natkovich, J. Pan, and D. Matheson, "A Batched Multi-Armed Bandit Approach to News Headline Testing," in *IEEE International Conference on Big Data (Big Data)*. Seattle, WA, USA, USA: IEEE, 2018.

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," vol. 42, no. 8, p. 30–37, Aug. 2009. [Online]. Available: https://doi.org/10.1109/MC.2009.263

[7] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, p. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[8] W. Sun, X. Zhang, W. Liang, and Z. He, "High dimensional explicit feature biased matrix factorization recommendation," in *Trends and Applications in Knowledge Discovery and Data Mining*, X.-L. Li, T. Cao, E.-P. Lim, Z.-H. Zhou, T.-B. Ho, and D. Cheung, Eds. Springer International Publishing, 2015, vol. 9441, pp. 66–77, series Title: Lecture Notes in Computer Science.

[9] O. Alieva, E. Gangan, E. Ilyushin, and A. Kachalin, "Automatic evaluation of recommendation models," *International scientific journal «Modern Information Technologies and IT-Education»*, vol. 16, no. 2, 2020. [Online]. Available: http://sitito.cs.msu.ru/index.php/SITITO/article/view/656

[10] S. Li, A. Karatzoglou, and C. Gentile, "Collaborative Filtering Bandits," in *SIGIR '16: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, Pisa, Italy, 2016, pp. 539–548.

[11] S. Pandey, D. Chakrabarti, and D. Agarwal, "Multi-armed bandit problems with dependent arms," in *Proceedings of the 24th international conference on Machine learning - ICML '07*. Corvalis, Oregon: ACM Press, 2007, pp. 721–728. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1273496.1273587

[12] S. Caron and S. Bhagat, "Mixing bandits: a recipe for improved cold-start recommendations in a social network," 08 2013.

[13] H. T. Nguyen and A. Kofod-Petersen, "Using Multi-armed Bandit to Solve Cold-Start Problems in Recommender Systems at Telco," in *Mining Intelligence and Knowledge Exploration*, R. Prasath, P. O'Reilly, and T. Kathirvalavakumar, Eds. Cham: Springer International Publishing, 2014, vol. 8891, pp. 21–30, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-13817-63

[14] C. Z. Felício, K. V. Paixão, C. A. Barcelos, and P. Preux, "A Multi-Armed Bandit Model Selection for Cold-Start User Recommendation," in *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. Bratislava Slovakia: ACM, Jul. 2017, pp. 32–40. [Online]. Available: https://dl.acm.org/doi/10.1145/3079628.3079681

[15] A. Slivkins, "Introduction to multi-armed bandits," *Foundations and Trends® in Machine Learning*, vol. 12, no. 1-2, pp. 1–286, 2019. [Online]. Available: http://dx.doi.org/10.1561/2200000068

[16] S. Bubeck and A. Slivkins, "The best of both worlds: Stochastic and adversarial bandits," *Journal of Machine Learning Research*, vol. 23, 02 2012.

[17] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 2020.

[18] J. M. White, *Bandit Algorithms for Website Optimization*. O'Reilly Media, Inc. [Online]. Available: https://www.oreilly.com/library/view/bandit-algorithms-for/9781449341565/

[19] P. Auer and N. Cesa-Bianchi, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning 47*, p. 22, 2002.

[20] O. Chapelle and L. Li, "An Empirical Evaluation of Thompson Sampling," in *Processing Advances in Neural Information Systems*, vol. 24, 2011, pp. 2249–2257. [Online]. Available: http://papers.nips.cc/paper/4321-an-empirical-evaluation-of-thompson-sampling.pdf

[21] A. Gopalan, S. Mannor, and Y. Mansour, "Thompson Sampling for Complex Online Problems," in *Proceedings of the 31 st International Conference on Machine Learning*, Beijing, China, 2014.

[22] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," ser. Proceedings of Machine Learning Research, S. Mannor, N. Srebro, and R. C. Williamson, Eds., vol. 23. Edinburgh, Scotland: JMLR Workshop and Conference Proceedings, 25–27 Jun 2012, pp. 39.1–39.26. [Online]. Available: http://proceedings.mlr.press/v23/agrawal12.html

[23] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on thompson sampling," *Found. Trends Mach. Learn.*, vol. 11, no. 1, p. 1–96, Jul. 2018. [Online]. Available: https://doi.org/10.1561/2200000070

[24] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *Journal of Machine Learning Research*, vol. 1, 02 2014.

[25] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The Nonstochastic Multiarmed Bandit Problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, Jan. 2002. [Online]. Available: http://epubs.siam.org/doi/10.1137/S0097539701398375

[26] J. Langford and T. Zhang, "The epoch-greedy algorithm for multi-armed bandits with side information," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 817–824. [Online]. Available: http://papers.nips.cc/paper/3178-the-epoch-greedy-algorithm-for-multi-armed-bandits-with-side-information.pdf

[27] D. Cortes, "Adapting multi-armed bandits policies to contextual bandits scenarios," *arXiv:1811.04383v2*, 2019.

[28] L. Zhou, "A survey on contextual multi-armed bandits," *ArXiv*, vol. abs/1508.03326, 2015.

[29] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: JMLR Workshop and Conference Proceedings, 11–13 Apr 2011, pp. 208–214. [Online]. Available: http://proceedings.mlr.press/v15/chu11a.html

[30] L. Zhou and E. Brunskill, "Latent Contextual Bandits and their Application to Personalized Recommendations for New User," *arXiv:1604.06743 [cs.LG]*, 2016.

[31] S. LI, B. Wang, S. Zhang, and W. Chen, "Contextual Combinatorial cascading Bandits," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 2016.

[32] X. Xu, F. Dong, Y. Li, S. He, and X. Li, "Contextual-Bandit Based Personalized Recommendation with Time-Varying User Interests," *arXiv:2003.00359v1 [cs.LG]*, 2020.

[33] B. Hao, Y. Abbasi Yadkori, Z. Wen, and G. Cheng, "Bootstrapping upper confidence bound," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019. [Online]. Available: http://papers.nips.cc/paper/9382-bootstrapping-upper-confidence-bound.pdf

[34] D. Zhou, L. Li, and Q. Gu, "Neural Contextual Bandits with UCB-based Exploration," in *Proceedings of the 37 th International Conference on Machine Learning*, Vienna, Austria, 2020.

[35] S. Agrawal and N. Goyal, "Thompson Sampling for Contextual Bandits with Linear Payoffs," in *Proceedings of the 30 th International Conference on Machine Learning*, vol. 28, Atlanta, Georgia,, 2013.

[36] I. Osband and B. Roy, "Bootstrapped thompson sampling and deep exploration," 07 2015.

[37] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li, "Personalized Recommendation via Parameter-Free Contextual Bandit," in *SIGIR '15: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 323–332.

[38] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased Offline Evaluation of Contextual-bandit-based news Article Recommendation Algorithms," *WSDM'11*, 2011. [Online]. Available: https://arxiv.org/pdf/1003.5956.pdf

[39] C. Riquelme, G. Tucker, and J. Snoek, "Deep bayesian bandits showdown," in *International Conference on Learning Representations*, Vancouver Canada, 2018.

[40] M. Dudík, D. J. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient optimal learning for contextual bandits," *ArXiv*, vol. abs/1106.2369, 2011.

[41] L. Reyzin, "New algorithms for contextual bandits," 2012. [Online]. Available: http://www.levreyzin.com/presentations/BGU2010.pdf

[42] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. E. Schapire, "Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits," in *Proceedings of the 31 st International Conference on Machine Learning*, Beijing, China, 2014.

[43] T. Uchiya, A. Nakamura, and M. Kudo, "Algorithms for adversarial bandit problems with multiple plays," in *Algorithmic Learning Theory*, M. Hutter, F. Stephan, V. Vovk, and T. Zeugmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 375–389.

[44] J. Louedec, M. Chevalier, J. Mothe, A. Garivier, and S. Gerchinovitz, "A Multiple-play Bandit Algorithm Applied to Recommender Systems," *Association for the Advancement of Artificial Intelligence*, 2015.

[45] Y.-H. Chang and H.-T. Lin, "Pairwise Regression with Upper Confidence Bound for Contextual Bandit with Multiple Actions," in *Proceedings of the Conference on Technologies and Applications for Artificial Intelligence*, 2013, pp. 19–24.

[46] J. Komiyama, J. Honda, and H. Nakagawa, "Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 1152–1161.

[47] R. Busa-Fekete, B. Szoenyi, P. Weng, and S. Mannor, "Multi-objective Bandits: Optimizing the Generalized Gini Index," *arXiv:1706.04933v1*, 2017.

[48] J. A. Weymark, "Generalized Gini inequality indices," Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), CORE Discussion Papers RP 453, Jan. 1981. [Online]. Available: https://ideas.repec.org/p/cor/louvrp/453.html

[49] S. Yahyaa, M. Drugan, and B. Manderick, "Thompson sampling in the adaptive linear scalarized multi objective multi armed bandit," vol. 2, 01 2015.

[50] C. Tekin and E. Turgay, "Multi-objective contextual bandits with a dominant objective," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017, pp. 1–6.

[51] E. Turgay, D. Öner, and C. Tekin, "Multi-objective contextual bandit problem with similarity information," 03 2018.

[52] C. Gentile, S. Li, and G. Zappella, "Online Clustering of Bandits," in *Proceedings of the 31 st International Conference on Machine Learning*, vol. 34, Beijing, China, 2017.

[53] T. Nguyen, "Dynamic clustering of contextual multi-armed bandits. in proceedings of the 23rd acm international conference on conference on information and knowledge management (cikm '14)," 11 2014.

[54] H. Wang, Q. Wu, and H. Wang, "Factorization bandits for interactive recommendation," in *AAAI'17: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 2695–2702.

[55] J. Kawale, H. H. Bui, B. Kveton, L. Tran-Thanh, and S. Chawla, "Efficient Thompson Sampling for Online Matrix-Factorization Recommendation," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1297–1305. [Online]. Available: http://papers.nips.cc/paper/5985-efficient-thompson-sampling-for-online-matrix-factorization-recommendation.pdf

[56] D. Chakrabarti, R. Kumar, E. Upfal, and F. Radlinski, "Mortal Multi-Armed Bandits," in *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, Vancouver, British Columbia, Canada, 2008.

[57] C. Zeng, Q. Wang, S. Mokhtari, and T. Li, "Online Context-Aware Recommendation with Time Varying Multi-Armed Bandit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, Aug. 2016, pp. 2025–2034. [Online]. Available: https://dl.acm.org/doi/10.1145/2939672.2939878

[58] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[59] B. Brodén, M. Hammar, D. Paraschakis, and B. J. Nilsson, "Ensemble Recommendations via Thompson Sampling: an Experimental Study within e-Commerce," in *IUI '18: 23rd International Conference on Intelligent User Interfaces*, 2018, pp. 18–29.

[60] D. Bouneffouf, R. Laroche, T. Urvoy, R. Feraud, and A. Robin, "Contextual bandit for active learning: Active thompson sampling," 11 2014.

[61] C.-C. Hsieh, J. Neufeld, T. King, and J. Cho, "Efficient approximate thompson sampling for search query recommendation," ser. SAC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 740–746. [Online]. Available: https://doi.org/10.1145/2695664.2695748

[62] M. Lopes, B. Clement, D. Roy, and P.-Y. Oudeyer, "Multi-armed bandits for intelligent tutoring systems," 10 2013.

[63] J. Mary, P. Preux, and O. Nicol, "Improving offline evaluation of contextual bandit algorithms via bootstrapping techniques," *ArXiv*, vol. abs/1405.3536, 2014.

[64] A. Beygelzimer and J. Langford, "The offset tree for learning with partial labels," ser. KDD '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 129–138. [Online]. Available: https://doi.org/10.1145/1557019.1557040

[65] M. Dudík, D. Erhan, J. Langford, and L. Li, "Doubly robust policy evaluation and optimization," *Statist. Sci.*, vol. 29, no. 4, pp. 485–511, 11 2014. [Online]. Available: https://doi.org/10.1214/14-STS500

[66] S. A. M. M. Y. N. Saito, Yuta, "A large-scale open dataset for bandit algorithms," *arXiv preprint arXiv:2008.07146*, 2020.

[67] D. N. Hill, H. Nassif, Y. Liu, A. Iyer, and S. Vishwanathan, "An Efficient Bandit Algorithm for Realtime Multivariate Optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax NS Canada: ACM, Aug. 2017, pp. 1813–1821. [Online]. Available: https://dl.acm.org/doi/10.1145/3097983.3098184

[68] A. Chandrashekar, F. Amat, J. Basilico, and T. Jebara. Artwork personalization at netflix. [Online]. Available: https://netflixtechblog.com/artwork-personalization-c589f074ad76

[69] M. Hejazinia, K. Eastman, S. Ye, A. Amirabadi, and R. Divvela, "Accelerated learning from recommender systems using multi-armed

bandit," *arXiv:1908.06158 [cs]*, Aug. 2019, arXiv: 1908.06158. [Online]. Available: http://arxiv.org/abs/1908.06158

[70] D. Precup, R. S. Sutton, and S. P. Singh, "Eligibility traces for off-policy policy evaluation," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, p. 759–766.

[71] W. Bendada, G. Salha, and T. Bontempelli, "Carousel personalization in music streaming apps with contextual bandits," in *Fourteenth ACM Conference on Recommender Systems*, ser. RecSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 420–425. [Online]. Available: https://doi.org/10.1145/3383313.3412217

[72] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li, "Personalized recommendation via parameter-free contextual bandits," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 323–332. [Online]. Available: https://doi.org/10.1145/2766462.2767707

[73] C. Zeng, Q. Wang, S. Mokhtari, and T. Li, "Online context-aware recommendation with time varying multi-armed bandit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 2025–2034. [Online]. Available: https://doi.org/10.1145/2939672.2939878

[74] D. Abensur, I. Balashov, S. Bar, I. Orlov, R. Lempel, and N. Moscovici, "Productization Challenges of Contextual Multi-Armed Bandits," *arXiv:1907.04884v1 [cs.IR]*, 2019.