Cold-start Problems in Recommendation Systems via Contextual-bandit Algorithms

Hai Thanh Nguyen* Jérémie Mary† Philippe Preux†

Abstract

In this paper, we study a cold-start problem in recommendation systems where we have completely new users entered the systems. There is not any interaction or feedback of the new users with the systems previoustly, thus no ratings are available. Trivial approaches are to select ramdom items or the most popular ones to recommend to the new users. However, these methods perform poorly in many case. In this research, we provide a new look of this cold-start problem in recommendation systems. In fact, we cast this cold-start problem as a contextual-bandit problem. No additional information on new users and new items is needed. We consider all the past ratings of previous users as contextual information to be integrated into the recommendation framework. To solve this type of the cold-start problems, we propose a new efficient method which is based on the LinUCB algorithm for contextual-bandit problems. The experiments were conducted on three different publicly-available data sets, namely Movielens, Netflix and Yahoo!Music. The new proposed methods were also compared with other state-of-the-art techniques. Experiments showed that our new method significantly improves upon all these methods.

1 Introduction

The goal of a recommender system is to select some items (*e.g.* movies, music, books, news, images, web pages and so on) that are likely to be of interest for a user on a webpage. This is done by matching some user-based characteristics with item-based characteristics. These characteristics can be information over the items to recommend (the content-based approach) or to find users with similar tastes (the collaborative filtering approach). We consider the quality of a recommendation to be the interest of the user on the item being recommended.

*INRIA Lille Nord Europe, 40 avenue Halley, 59650 Villeneuve d'Ascq, France, firstname.lastname@inria.fr

†University of Lille / LIFL (CNRS) & INRIA Lille Nord Europe, 59650 Villeneuve d'Ascq, France, firstname.lastname@inria.fr

In the content-based approach, a description of the items is readily available. We have to build a model of the user's tastes using the feedback we receive — this feedback can be implicit (*e.g.* clicks, browsing, ...) or explicit (*e.g.* rates). When a new user — without any side information — is introduced in the system, we need to collect some data in order to build a good enough model before being able to produce any valuable recommendation. This is a first kind of cold-start problem that we qualify as the *new user problem*. In this setting we want to balance the exploration of the tastes of the new user and the usage of this modeling to perform good recommendations.

In the collaborative filtering approaches, recommendations are made trying to identify users with similar preferences. Of course, it suffers from the new user problem, but it also suffers from a *new item problem*. Indeed, if a new item is introduced, the system will not recommend it until some users provide some positive feedback on it. Certainly, the probability of receiving some feedback is related to the number of recommendations of that item, so it sounds reasonable to provide a *boost* to the new items. Of course, this boost has to be designed carefully. If the boost is too strong, it will display too much the new items or even it will not be adapted to the current user (but we will have a lot of information on theses new items). If the boost is too low, the new items will never be displayed or even if they would be better than the old ones. So we have to set a balance between the exploration of theses new items and the quality of our recommendations.

Cold-start problems naturally arise in statistical data modeling when not enough data is available. A common way to solve the new item problem is to assign a default rate to new items based on the ratings assigned by the community to other similar items [3]. Item similarity is computed according to the items content-based characteristics. Certainly, this only holds when content-based characteristics are available.

In this paper, we make no use of any content-based information on users and items. We propose a new approach based on the bandit problem to tackle both the new user and the new item problems. In fact,

we cast the cold-start problems into contextual-bandit problems. We consider all the past ratings of previous users as context information in the recommendation systems. We adapt the standard LinUCB algorithm to our cases and propose a new efficient method, namely A-LinUCB algorithm.

To verify the proposed algorithm, we conduct an experiment on three data sets from Movielen, Netflix and Yahoo!Music. We also compare the new algorithm with different approaches, which are the random policy, ε-greedy, UCB, EXP3 and Thompson sampling. Experimental results showed that our new method significantly improves upon all these methods in both new user and new item recommendation systems.

The paper is organized as follows. Section 2 provides definitions of the new user and new item problems. In the Section 3, we describes the bandit algorithms and their perspectives for cold-start problems. Section 4 introduces the contextual-bandit approach and an adapted version of the LinUCB algorithm for the cold-start problems. We present the experiments in Section 5. The last section summarizes our findings and discuss about future works.

2 Problem Definition

Assume that at a given moment, we have a set of n possible items to recommend. Let $X \in \mathbb{R}^{k\times n}$ be a matrix of description of theses items (one item per column).

For all $i \in \{1, \ldots, m\}$, let $\theta_i \in \mathbb{R}^k$ be a row vector describing the i^{th} user, and let $b(\theta_i) \in \mathbb{R}^n$ be a vector of tastes of this user for the n items, a taste being expressed as a number. The j^{th} value of $b(\theta_i)$ is the affinity of the user i for the j^{th} item. The affinity is built using implicit of explicit feedback over the tastes of the users. Some of the values are unknown.

Let U be the matrix of description of all users. The i^{th} row of U is θ_i. Let $B \in \mathbb{R}^{m\times n}$ be the affinity matrix. The i^{th} row of B is the vector $b(\theta_i)$.

A common goal is to predict the missing values of B using the available descriptions of users and/or items. Classically the error of an algorithm is seen as the reconstructing error of B — the users tastes — from the available data. For that reconstruction, a classical measure of performance is the root mean square error (RMSE) but some authors have proposed different criterions such as rank preservation [19].

To reconstruct the missing values, a very common assumption is to consider that there exists a latent description space common to both items and users, so that a linear relation holds between the tastes and the vectors of descriptions. Formally for each user we want to write $b(\theta_i) = \theta_i \cdot X$. When performed for all users at once, this method is known as matrix factorization since we are trying to write $B = U \cdot X$.

But this approach fails short to deal with the occurrence of new users or new items. Indeed, for rows or columns of A with no or almost no information in the data set, there is no way to know if our reconstruction is correct or not. In a real application, data are received in an online way: after each recommendation to an user, we have the possibility to collect a feedback. This means that it is possible to recommend items in order to collect information on the user (so as to categorize it) in order to better recommend latter. In particular, we are going to show that it is not optimal to always recommend the "best" item according to the current estimates of the tastes of the user, a strategy said to be *greedy*.

As in this online setting the RMSE evaluation is not enough, we have to rely on a different evaluation process. Assuming a brand new user is visiting us, a reasonable recommender system tries to present the item with the highest affinity. One of the most common online evaluation is the regret, which is the deference of performance between making the best decision and the decision that has been made. Cumulative regret is simply the sum over time of the regret. The behavior of this quantity is studied in the bandits framework [6, 1]. In fact, the cumulative regret is similar to the average score of our recommendations when done sequentially. The regret is more used than the average score because it highlights better the difference between algorithms — this is because the average score of the items does not appear in the regret formulation so cumulative score grows in $O(T)$ while differences between algorithm will be in $O(\sqrt{(T)})$ where T is the number of recommendations made —Translated in our setting this means at each time step t:

1. we receive the visit of a user i_t who is either already known to the recommendation system, or not. For this user, at timestep t some of her tastes $b(\theta_{i_t})$ are known by the recommender and some others are unknown. We note b_t^* the highest $b(\theta_{i_t})$ currently not known by the recommender . Of course b_t^* is not revealed, it is just used for the sake of the definition of the regret.

2. The recommender chooses an item to recommend $j_t = \pi(i_t, t)$ among the unknown one for i_t. The corresponding value of $b(\theta_{i_t})$ is revealed. The regret is increased by $(b_t^* - b_{\pi(i_t,t)}(\theta_{i_t}))$.

The objective is to find a best strategy that provides the minimal cumulative regret:

$$CumulativeRegret = \sum_{t=1}^{T} (b_t^* - b_{\pi(i_t,t)}(\theta_{i_t}))$$

Of course, the computation of the cumulative regret requires the knowledge of all the values of B which can be problematic on some real datasets.

2.1 New User/Item Problem

Definition 1. (The new user problem) *When a set of new users is introduced in our recommender system, we want to recommend the items to them and get the feedback on the recommended items minimizing the cumulative regret — see section 2 — on theses users.*

Our global strategy for solving this problem is following:

1. Select k users. The tastes of theses users are seen as the descriptions of the items. This means that we use as X matrix some of the rows of B. This special set of users is designed as the *base users* and the X is the *base matrix*.

2. As the description of the items contains some missing values, we are going to fill them using different strategies.

3. Then select items using a contextual bandit algorithm as described in section 4.

We are going to find how to express the tastes of our new users as a linear combination of the base users. This is equivalent to find the θ_i parameters for this new users. When searching this combination we pay attention to the confidence intervals of our estimates and sometimes sample in order to reduce the variance of the estimates instead of selecting the optimal choice with respect to the current maximum of likelihood.

Definition 2. (The new items problem) *When a set of new items is introduced in our recommender system, we want to use the visits of our users in order to improve our statistical confidence on theses items. We perform this allocation trying to minimize the cumulative regret on the users.*

Our global strategy for this case is almost the same with the new user problem's strategy. The only difference is that we will use the *base users* to compute a confidence bound over the description of the items.

In the next section, we will introduce the bandit algorithms and their perspectives to solve these cold-start problems.

3 Bandit algorithms and perspectives for cold-start problems

At each moment, we have to select an item to recommend to the present user. This selection is based on uncertain information. Therefore, we may either choose an item that is most likely to appeal to the user (based on uncertain grounds, though), or choose an item that is likely to bring new information. This means that we may either suggest an item that will bring immediate benefit, or select an item that will let the system improve in the longer run. This is the usual dilemma between exploitation (of already available knowledge) *versus* exploration (of uncertainty), encountered in sequential decision making under uncertainty problems. This problem has been addressed for decades in the bandit framework that we briefly introduced now.

Let us consider a set of possible items. Each item is associated to an unknown probability of ratings. The game is repeated and the goal is to accumulate the minimal cumulative regret. In the present context, ratings may be binary (click/no-click), or belong to a set of values (a set of possible rate, ranging from 1 to 5). This problem has been studied for decades, many approaches have been proposed. Let us introduce a few of them that we use later in this paper:

- **Random** consists in picking up one of the possible items, uniformly at random.

- **ε-greedy(EGreedy)** [6] consists in picking up the item that is currently considered the best with probability ε (exploit current knowledge), and pick it up uniformly at random with probability $1 - \varepsilon$ (explore to improve knowledge). Typically, ε is varying along time so that the items get greedier and greedier as knowledge is gathered.

- **UCB** [6] consists in selecting the item that maximizes the following function: $\hat{\mu}_j + \sqrt{\frac{2 \ln t}{t_j}}$ where t is the current timestep, μ_j is the average rating obtained when selecting item j, t_j is the number of times item j as been selected so far. In this equation, $\hat{\mu}_j$ favors a greedy selection (exploitation) while the second term $\sqrt{\frac{2 \ln t}{t_j}}$ favors exploration driven by uncertainty: it is a confidence interval on the true value of the expectation of rating for item j.

- **EXP3** [7] consists in selecting an item according to a distribution, which is a mixture of the uniform distribution and a distribution that assigns to each item a probability mass exponential in the estimated cumulative ratings for that item.

- **Thompson sampling(TS)** [5] is a Bayesian approach to this problem. It consists in computing the probability of success of each recommendation, and then greedily selecting the item associated with maximum *a posteriori* probability.

In the setting we consider in Section 2, we have information about the base items/users and we may also have information about the ratings of the base matrix X. This is known as side information, or context, hence bandit with side information or contextual bandit. The bandit approaches considered so far do not take this side information into account. There are various methods for the contextual bandits (*e.g.* OTS [16] and LinUCB [13]); here, we concentrate on LinUCB [13] because of its efficiency. More details are given in the section 4.

4 Contextual-bandit for cold-start problems

In the setting considered in this paper, we assume that there is no contextual information available about neither the items, nor the users. However, the ratings already recorded from users on items may be used as a context for the new users. We elaborate on this idea in the subsequent section.

We assume that there is a base rating matrix $X = (X_j)_{j=1}^n$ of dimension $k \times n$, where X_j is the j^{th} column of X. It is not necessary that the X to be a full rating matrix. Because in practice of recommendation systems, it is difficult to get all the ratings of the users for all available items. However, each user should have at least a rating. The missing values in X can be filled by zero, or average values or the values approximated by using matrix decomposition techniques, such as Singular Value Decomposition (SVD). We will discuss about this more in the experiment part.

At a time step t, if a new user comes, her rating for a particular item j is linear in its context vector X_j with some unknown coefficient vector θ_{i_t}:

$$b_j(\theta_{i_t}) = \theta_{i_t} X_j$$

Let $D_{t,j}$ be a context description matrix of dimension $t \times k$, where rows are the context vector X_j. When applying the ridge regression to the training data $(D_{t,j}, b_j)$, we get the estimation of the unknown variable θ_{i_t} as follows:

$$\theta_{i_t} = (D_{t,j}^T D_{t,j} + I_k)^{-1} D_{t,j}^T b_j$$

where I_k is the $k \times k$ identity matrix. As shown in [13], with probability at least $1 - \delta$ we have:

$$|\theta_{i_t} X_j - E[b_j(\theta_{i_t})|X_j]| \leq \alpha \sqrt{X_j^T (A_{t,j})^{-1} X_j}$$

for any $\delta > 0$, $\alpha = 1 + \sqrt{ln(1/\delta)/2}$; where $A_{t,j} = D_{t,j}^T D_{t,j} + I_k$ and $E[b_j(\theta_{i_t})|X_j]$ is the expected rating value of the new user on the item j, given the X_j. It can be easily seen that: $D_{t,j}^T D_{t,j} + I_k = tX_j X_j^T + I_k$. Therefore,

$$|\theta_{i_t} X_j - E[b_j(\theta_{i_t})|X_j]| \leq \alpha \sqrt{X_j^T (tX_j X_j^T + I_k)^{-1} X_j}$$

Obviously, we can immediately apply the standard LinUCB algorithm for this case. However, the efficiency of the algorithm will decreased over the time. Because when the size of $D_{t,j}$ increases, the inversion of the matrix $A_{t,j}$ is harder and slower. It is desirable to get more efficient algorithm than the standard LinUCB. Below, we propose a new adapted LinUCB (A-LinUCB) for the new user/item recommendation system problem and we will demonstrate its performance in the experiment part. Also note that unlike the standard LinUCB we work on context defined by rates and do not try to discover the *best* item as we do not want to recommend an item with a known taste.

LEMMA 4.1. *For all $t \geq 1$ and $X_j \in [0,1]^k$, the following inequation holds:*

$$(tX_j X_j^T + I_k)^{-1} \leq (X_j X_j^T + I_k)^{-1}$$

Following the lemma, we have a new inequality for A-LinUCB as follows:

$$|\theta_{i_t} X_j - E[b_j(\theta_{i_t})|X_j]| \leq \alpha \sqrt{X_j^T (X_j X_j^T + I_k)^{-1} X_j}$$

The inequality leads to a new adapted LinUCB (A-LinUCB) as described in Algorithm 1.

As the confidence interval $(X_j^T (X_j X_j^T + I_k)^{-1} X_j)$ for each item does not change over time, it may be a question about the impact of the exploration on the actual recommendation results. In other words, what if we do not explore different items by setting α to zero. It turns out that in practice we still need a little exploration because when $\alpha = 0$, the performance of the algorithms will be dramatically decreased. The details of this discussion will be provided in the experiment part.

5 Experiment

In this section, we conduct comprehensive experiments to evaluate the performance of the proposed A-LinUCB on solving the cold-start problems in recommendation systems. In order to do that, we first give the detail on the data sets used in the experiments. We then describe the experimental settings, in which the implementation of the A-LinUCB and other methods to compare with are provided . Finally, we analyze and discuss about the experimental results.

Algorithm 1 A-LinUCB for new user problem. (A-LinUCB for the new item problem is almost the same, except that the base matrix X needs to be transformed before running the algorithm)

procedure A-LinUCB(T, α, X)
  Store the matrices $A_j = I_k + X_j X_j^T, j = 1 \dots n$.
  for t in $1 : T$ **do**
    for j in $1 : n$ **do**
      if j is new **then**
        $b_j \leftarrow 0_{n \times 1}$
      end if
      $\theta_t \leftarrow A_j^{-1} b_j$
      $p_{t,j} \leftarrow \theta_t X_j + \alpha \sqrt{X_j^T (A_j)^{-1} X_j}$
    end for
    Draw a user at random i_t
    Choose item
  $j_t = \text{argmax}_j \{p_{t,j} |$ taste of i_t for j is unknown $\}$
    Observe the real rating b_{j_t}
    $b_j \leftarrow b_j + b_{j_t} X_{j_t}$
  end for
end procedure

5.1 Data sets

We used three different publicly-available data sets, namely Movielens[1], Netflix[2] and the Yahoo!Music[3]. The data sets Netflix and Yahoo!Music contain users that have not any ratings for any items. We eliminated these users from the original data sets because with these users, it would not be possible to estimate the regret. In more detail, originally the Netflix data set contains 100,000 ratings (from 0 to 5, where 0 = not rated) of 68,357 users on 17,770 movies. After the elimination, we obtained a data set that has 13,545 ratings of 6423 users on 1250 items. With the original Yahoo!Music data set, we have 11,557,943 ratings of 98,111 artists by 1,948,882 anonymous users. We reduced the data to a smaller one, which contains 20,361,089 ratings of 50,080 artists by 483,273 users. Because of the limited memory, from the original eliminated Yahoo!Music dataset we randomly selected several subsets for our experiment and then the results will be averaged. In Table 1, we summarize the size of data sets in our experiments.

For the convenience in analysis of the results and in presenting the cumulative regrets on the graphs, we normalized the data sets before running the experiments.

With each data set, we randomly splited it into two parts: The first part is the base matrix X and the second part Y is the remained data. Note that we will fill the missing values in the X only and use the matrix Y to measure the performance of the algorithms.

[1]http://www.grouplens.org/
[2]http://www.netflixprize.com/
[3]http://kddcup.yahoo.com/

Table 1: Number of users, items and ratings in datasets after eliminating the users without any votes from original datasets.

Datasets	No. Users	No. Items	No. Ratings
Movielens	6,040	3,952	1,000,209
Netflix	6,423	1,250	13,545
Yahoo!Music	483,273	50,080	20,361,089

5.2 Experiment setting

This subsection gives a detailed description of our experimental setup, including missing values handling, A-LinUCB algorithms for the new user and new item recommendation systems, competing algorithms and performance evaluation.

5.2.1 Dealing with missing values

It happens always in recommendation systems that some users do not give the rates on many items. Therefore, the data sets usually contain many missing values. In our experiments, the base matrix X may have these values as well. It is necessary to fill the matrix X before running the A-LinUCB algorithm.

- The simplest approach is to use the zero value. This method is reasonable since in some cases, if an user does not rate for a particular item, then it is probably that she does not like the item. Of course, some items are unknown to the user. However, considering the efficiency of this approach, especially, when we will be able to use the computing advantage with large sparse matrix, we would take into account it and compare with other possible techniques.

- The second approach, which is also very efficient in practice, is to utilize the average values of the ratings for items to fill the missing values.

- Recently, several matrix decomposition/completion techniques are applied and demonstrated to be promising approaches. In our experiment, we implement two popular algorithms, which are 1) the imputed SVD and 2) ALS-WR decomposition.

  Ad.1 Firstly, the missing values in a column vector of X are filled with the average value. The obtained matrix is then approximated by the standard SVD method [8] keeping only the k biggest singular values.

Ad.2 ALS-WR [22] which is a Tikhonov regularization of the classical matrix decomposition with low rank assumption. The minimization is solved by a well initialized alternate least square and is very efficient on Netflix dataset.

Three above described approaches are compared with each other in terms of the performance of the A-LinUCB. We will use the best result for later experiments.

5.2.2 Settings of the A-LinUCB algorithms

The implementation of the A-LinUCB algorithm is simple as shown in Algorithm 1. The only parameter of this policy is the α. We executed the A-LinUCB with different values of α and recorded the best choice.

Basically, the A-LinUCB algorithms for the new user problem is different from the A-LinUCB for the new item problem. However, their implementations as shown in Algorithm 1 are almost the same, except that the selection of the base matrix X of dimension $k \times n$ is distinct. The meanings of the k and n will be exchanged when we change from a problem to another problem. In other words, for the new item problem, the k will become the number of base items and the n will become the number of base users.

In general, the values of k and n of the matrix X can be arbitrary numbers. In our experiment, we observed that the case when k equals to n ($k = n$) provided the most consistent results and because of the limited size we only reported these results.

In our experiment, we selected the base matrix X for each problem on each data set as follows:

- For the new user problem, with the Movielens and Netflix data sets we used the initial numbers of items for the dimension of the base matrices X. In more detail, the X for the Movielens and for the Netflix will have the dimensions of 3952 and 1250, respectively. With the Yahoo!Music data set, because of the limited memory we randomly chose the base matrix X of the size 1000×1000 five times. The results were averaged.

- For the new item problem, we picked up the matrices X with following dimensions: 1000×1000 (Movielens), 500×500 (Netflix) and 1000×1000 (Yahoo!Music).

5.2.3 Competing algorithms

To the best our knowledge, no existing work applies contextual multi-armed bandit algorithms to solve the cold-start problems in recommendation systems when the side information on users and items is not available. Nevertheless, we compare the new proposed A-LinUCB with the following algorithms:

- The first and the most naive approach for the cold-start recommendation systems is to choose randomly an item to recommend to a new user. This algorithm is very efficient, especially, when we do not have any description on users or items, the algorithm seems to be reasonable. However, its performance in many cases fails as we will show in the experimental results. It is desirable to design better algorithms.

- The next class of algorithms that we want to compete with is the zero-contextual multi-armed bandit algorithms. The purpose of this comparison is to show the value of taking into account context information in the cold-start recommendation systems. The algorithms involved in the comparison are:

  – ε-greedy(EG): As described in Section 3, it estimates the rating of each user/item; then selects a random user/item with probability ε_t, and choose an user/item of the highest average value of revealed ratings with probability $1 - \varepsilon_t$. The parameter ε_t is decreased over the time t. In fact, the ε_t is calculated as follows: $\varepsilon_t = min(1, (cn)/(d^2(t - n - 1)))$, where c and d are chosen constants.

  – UCB: As described in Section 3, this policy estimates the average value of previously revealed ratings for each item as well as a confidence interval of the estimation. Afterward, the UCB always choose the item with the highest UCB. Specifically, following UCB [6], we calculated an item j's confidence interval by $\sqrt{\frac{2 \ln t}{t_j}}$, where t_j is the number of times the item j was selected prior to trial t.

  – EXP3: As described in Section 3 and accodring to the algorithm in [7], we selected $\gamma = 0.01$ before drawing the probability to select the best item to recommend to the new user.

- Thompson Sampling(TS) for contextual bandits: This policy attempts to utilize the previous ratings as the context information for the cold-start recommendation systems. By means of this comparison, we want to show that the utilization of the context information should always be in an appropriate way. The details on the implementation of the TS algorithm is as follows: At every step t, we generated a n-dimensional sample $\mu(t)$ from multi-variate Gaussian distribution, which depends

on the true rating of the last time recommended item and its context information. We then solved the problem $argmax_i(\mu(t)X_j)$ to get the next item for the recommendation.

- Finally, since the confidence interval $X_j^T(X_jX_j^T + I_k)^{-1}X_j$ described in Section 4 is fixed for each item, we doubt about the impact of the exploration in the cold-start recommendation systems. To answer the question, we compare the A-LinUCB with the A-LinUCB where the α is set to zero. For this case, at every time step t an item with maximal value $\theta_t X_j$ is selected.

We ran each of these algorithms 10 times with different choices of parameters. The best results were recorded as shown in Table 3, Table 5 and Table 7.

5.2.4 Performance metric

As discussed in Section 2, it will not be able to use the offline evaluation methods, such as RMSE or MSE for the cold-start recommendation systems. In our experiments, we utilize an online measurement, which is the cumulative regret as defined as follows:

$$CumulativeRegret = \sum_{t=1}^{T}(b_t^* - b_{\pi(i_t,t)}(\theta_{i_t}))$$

For this evaluation, the b_t^* of the new user may be not available. Therefore, we will use the maximal value among known ratings of the user instead. We assume that the real rating value $b_{\pi(i_t,t)}$ is revealed after the recommending the item $\pi(i_t, t)$ to the new user. However, if this rating value is unknown in the testing data Y, we will replace it by zero.

In practice, it is not always that the value b_t^* is accessible, yet we can use the maximal value of the rating scale in recommendation systems, such as the number 5 in the Netflix data. Moreover, in some recommendation systems, we do not get the explicit values of $b_{\pi(i_t,t)}$. For this case, we must define another online evaluation measurement. The topic is out of the scope of this paper and we reserver it for the future study.

5.3 Results and Analysis

5.3.1 Results on dealing with missing values

We filled the missing values in the base matrix X by using three approaches described above. We then ran the A-LinUCB algorithm with $\alpha = 0.001$, which is the best choice in our experiments, on the matries Y of the Movielen, Netflix and Yahoo!Music data sets. Finally, we measured the cumulative regrets of the new user recommendation systems. The Table 2 shows the obtained performances.

It can be seen that the Average approach works best with smallest cumulative regret. The Zero method is slightly worse than the Average on the Movielen data set only, yet it gained more efficiency in terms of running time. Supprisingly, the impute SVD and the alternative SVD provided worst resutls in our experiments.

Over all, we decided to use the Zero approach for the further comparisons because of its performances and efficency.

Table 3: Cumulative regrets for the **new user** recommendation system.

Methods	Movielens	Netflix	Yahoo!Music
Random	2024.26	3852	1514.30
Aver	1255	3825	1125.15
EGreedy	1218.4	3758.66	1108.97
UCB	1973.3	3850.86	1514.77
EXP3	1868.4	3836.66	1276.42
LinUCB	1595.86	3732.86	1325.86
A-LinUCB	1069.8	3659.4	989.92

Table 4: Running time of the **new user** recommendation system (in seconds).

Methods	Movielens	Netflix	Yahoo!Music
Random	93	15.16	12.43
Aver	117	21.9	18.44
EGreedy	113	22.12	18.57
UCB	48.96	22.39	18.77
EXP3	38.28	23.43	19.12
LinUCB	20214	2361.6	2448
A-LinUCB	5184	654	612

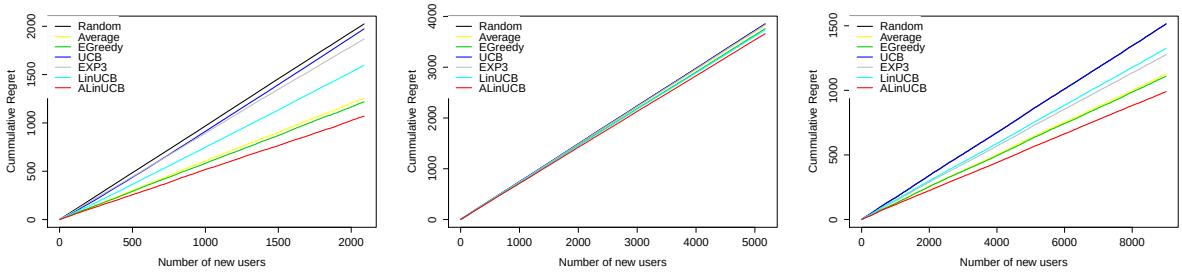
5.3.2 Results on competing different methods

As shown in Table 3 and Table 5, obviously the random policy is much worse than the A-LinUCB in terms of the cumulative regrets. For example, in the cased of the new user recommendation system on the Yahoo!Music data set, the A-LinUCB algorithms provided a better performance than the random's result by 34%.

Clearly, with no use of the information about previous ratings the UCB and EXP3 algorithms performed almost the same like the random strategy. Even the EXP3 is the worst choice in terms of the cumulative regret and the running time. The EG algorithm is exceptional since its performance is much better than the UCB and EXP3, yet still lower than the A-LinUCB. This can be explained by the fact that an item, which is preferred by an enough number of users, will likely be a

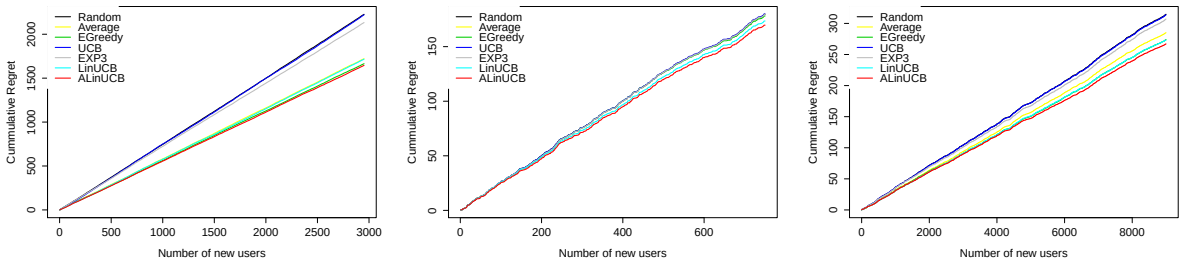
Table 2: Cumulative regrets for the new user recommendation system to compare three different approaches in dealing with missing values, namely by zero value, by average value, by impute SVD and by alternative SVD.

Dataset	By zero	By average	By impute SVD	By alt.SVD
Movielen	3008.19	3006.8	4872	4869.2
Netflix	3680	3680	3856	3856.8
Yahoo!Music	989.56	989.56	1143.81	1512.35



(a) Movielens with size 6040×3952. (b) Netflix with size 6423×1250 (c) Yahoo!Music with size $10,000 \times 1000$

Figure 1: Cumulative regrets for the **new user** recommendation systems. Several curses are hidden on the graphs as the performances of these methods is almost the same.



(a) Movielens with size 1000×3952 (b) Netflix with size 500×1250 (c) Yahoo!Music with size $1000 \times 10,000$

Figure 2: Cumulative regrets for the **new item** recommendation systems. Several curses are hidden on the graphs as the performances of these methods is almost the same.

Table 5: Cumulative regrets for the **new item** recommendation system.

Methods	Movielens	Netflix	Yahoo!Music
Random	2226.5	179.6	314.59
Aver	1721.86	179.6	285
EGreedy	1664.46	178.06	273.81
UCB	2221.46	179.93	313.75
EXP3	2133.19	179.2	306.92
LinUCB	1715.73	173.46	273.24
A-LinUCB	1645.46	169.6	266.95

Table 6: Running time of the **new item** recommendation system (in seconds).

Methods	Movielens	Netflix	Yahoo!Music
Random	35.36	0.94	12.26
Aver	47.75	1.49	18.28
EGreedy	45.29	1.46	18.41
UCB	35.62	1.43	18.67
EXP3	30.32	1.46	19.59
LinUCB	2325.27	57.5	2866.68
A-LinUCB	547.75	13.8	360

Table 7: Cumulative regrets for new user recommendation system to compare between A-LinUCB ($\alpha = 0$) with A-LinUCB ($\alpha = 0.001$).

A-LinUCB	Movielens	Netflix	Yahoo!Music
$\alpha = 0$	3491.39	3857	1514.15
$\alpha = 0.001$	3008.19	3680	989.56

good choice for the others. The greedy EG strategy has fully utilized this character of a recommendation systems to provide good results. For instance, in the case of the new movie recommendation system built on the Movielens data set, the EG policy performed very well with almost 10% better than the random strategy.

With the Thompson sampling (TS) policy, we got better results than the Random, UCB and EXP3. This illustrated that the information of previous ratings contributed to the better recommendation results. However, its performance is still lower than the EG and A-LinUCB. Therefore, we can conclude that the context information must be used in an appropriate way, which is the A-LinUCB as a sample in our experiments.

It can be seen from the Table 7 that the A-LinUCB algorithm with $\alpha = 0$ performed much worse than the A-LinUCB when $\alpha = 0.001$ on all three data sets. That means a little exploration helped to provide better results in solving cold-start recommendation systems.

6 Related work

The cold-start problem was readily identified as of the emergence of recommendation systems [18]. Along time, several solutions for these problems were proposed. However, these approaches depend strongly on side information available on the users and items, which is not always available. Therefore, it is very hard to build accurate recommendation systems in practice, despite the strong activity on the subject. For example, the work [11] suggested an interview process with users to gather more information about their preferences before the actual recommendations. Recently, a lot of works have been conducted to improve the estimation speed of the parameters for new items or new users by using hierarchy of items or various side informations [4, 21, 3]. The augmentation with information mined from social networks is also a common approach today.

Another common strategy to mitigate the cold-start user problem is to gather demographic data. It is assume that users who share a common background also share a common taste in products. Examples include Lekakos and Giaglis [12], where lifestyle information is employed. This includes age, marital status and education, as well as preferences on eight television genres. Correlation between users are found by applying the Pearson correlation coefficient. The authors report that this approach is the most effective way of dealing with the cold-start user problem in sparse environments.

A similar thought underlies the work by Lam et al., [10] where an *aspect model* (see e.g. [15]) including age, gender and job is used. This information is used to calculate a probability model that classifies users into user groups and the probability how well liked an item is by this user group.

Other examples of applying demographic information to mitigate the cold-start user problem exists, e.g. [9, 2, 17]. All the the solution above use similar demographic information; most commonly age, occupation and gender. Most of the solutions asked for less that five pieces of information. Even though five is a comparatively small number, the user must still answer these questions. Users do generally not like to answer a lot of questions, yet expect reasonable performance from the first interaction with the user [23].

Zigoris and Zhang [23], suggests to use a two part Bayesian model, where the prior probability is based on the existing user population and *data likelihood*, which is based on the data supplied by the user. Thus, when a new user enters the system, little is know about that user and the prior distribution is the main contributor. As the user interacts with the system the data data likelihood becomes more and more important. This approach performs well for cold-start users. Other similar approaches can by found in [14], suggesting a Markov mixture model and [20], who suggests a statical user language model that integrates an individual model, a group model and a global model.

In this paper, we consider this problem from a different perspective. Though perfectly aware of the potential utility of side information, we consider the problem without any side information, only focussing on acquiring appetence of new users and appeal new items as fast as possible with as few as possible "bad" recommendations.

7 Conclusions and future study

The main focus of this paper is on cold-start problems in recommendation systems. We have casted these problems as contextual-bandit problems and adapted LinUCB to solve them. We have conducted performance analysis of the proposed A-LinUCB algorithms and compared it with six different approaches: random policy, ε-greedy, UCB, EXP3, Thompson sampling and A-LinUCB($\alpha = 0$). We have used three data sets from Movielens, Netflix and Yahoo!Music and the performance of algorithms were measured by the cumulative regret. Our proposed A-LinUCB algorithms

have clearly demonstrated better results than all the others in both new user and new item recommendation systems. As proposed in this paper, A-LinUCB requires no side information on users and items; A-LinUCB may be extended to take advantage of such sources of information; this is left as future work.

There are two main directions for future works: first, it would be interesting to extend the proposed framework to solve the cold-start system problem, where we have completely new users and new items. Second, we plan to study another forms of the cumulative regret for the case when only implicit feedbacks of users are available, such as clicks or browsing.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 2312–2320, 2011.

[2] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM SIGKDD, ACM, June 28 - July 1 2009.

[3] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. Spatio-temporal models for estimating click-through rate. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 21–30, New York, NY, USA, 2009. ACM.

[4] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, Nitin Motgi, Seung-Taek Park, Raghu Ramakrishnan, Scott Roy, and Joe Zachariah. Online models for content optimization. In *Advances in Neural Information Processing Systems 21th (NIPS)*, pages 17–24, 2008.

[5] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. *CoRR*, abs/1209.3352, 2012.

[6] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002.

[7] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

[8] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[9] Fengrong Gao, Chunxiao Xing, Xiaoyong Du, and Shan Wang. Personalized service system based on hybrid filtering for digital library. *Tsinghua Science & Technology*, 12(1):1–8, 2007.

[10] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, pages 208–211. ACM, 2008.

[11] Yezdi Lashkari, Max Metral, and Pattie Maes. Collaborative interface agents. In *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 444–449. AAAI Press, 1994.

[12] George Lekakos and George M. Giaglis. A hybrid approach for improving predictive accuracy of collaborative filtering algorithms. *User Modeling and User-Adapted Interaction*, 17(1-2):5–40, 2007.

[13] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, abs/1003.0146, 2010.

[14] Eren Manavoglu, Dmitry Pavlov, and C. Lee Giles. Probabilistic user behavior models. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 203. IEEE Computer Society, 2003.

[15] Benjamin Marlin. Collaborative filtering: A machine learning perspective. Technical report, University of Toronto, 2004.

[16] Benedict C. May, Nathan Korda, Anthony Lee, and David S. Leslie. Optimistic bayesian sampling in contextual-bandit problems. *J. Mach. Learn. Res.*, 13(1):2069–2106, June 2012.

[17] Seung-Taek Park and Wei Chu. Pairwise preference regression for cold-start recommendation. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 21–28. ACM, 2009.

[18] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 253–260, NY, USA, 2002. ACM.

[19] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 139–146, NY, USA, 2012. ACM.

[20] Gui-Rong Xue, Jie Han, Yong Yu, and Qiang Yang. User language model for collaborative personalized search. *ACM Transactions on Information Systems*, 27(2):1–28, 2009.

[21] Yisong Yue, Sue Ann Hong, and Carlos Guestrin. Hierarchical exploration for accelerating contextual bandits. In *ICML*, 2012.

[22] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management (AAIM)*, pages 337–348, Berlin, Heidelberg, 2008. Springer-Verlag.

[23] Philip Zigoris and Yi Zhang. Bayesian adaptive user

profiling with explicit & implicit feedback. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 397–404. ACM, 2006.