Bandits and Recommender Systems

Jérémie Mary, Romaric Gaudel, and Philippe Preux

Université de Lille, CRIStAL (UMR CNRS), Villeneuve d'Ascq, France
{jeremie.mary,romaric.gaudel,philippe.preux}@univ-lille3.fr

Abstract This paper addresses the on-line recommendation problem facing new users and new items; we assume that no information is available neither about users, nor about the items. The only source of information is a set of ratings given by users to some items. By on-line, we mean that the set of users, and the set of items, and the set of ratings is evolving along time and that at any moment, the recommendation system has to select items to recommend based on the currently available information, that is basically the sequence of past events. We also mean that each user comes with her preferences which may evolve along short and longer scales of time; so we have to continuously update their preferences. When the set of ratings is the only available source of information, the traditional approach is matrix factorization. In a decision making under uncertainty setting, actions should be selected to balance exploration with exploitation; this is best modeled as a bandit problem. Matrix factors provide a latent representation of users and items. These representations may then be used as contextual information by the bandit algorithm to select items. This last point is exactly the originality of this paper: the combination of matrix factorization and bandit algorithms to solve the on-line recommendation problem. Our work is driven by considering the recommendation problem as a feedback controlled loop. This leads to interactions between the representation learning, and the recommendation policy.

1 Introduction

We consider the online version of the problem of the recommendation of items to users as faced by websites. Items may be ads, news, music, videos, movies, books, diapers, ... Being live, these systems have to cope with users about whom we have no information, and new items introduced in the catalog which attractiveness is unknown. Appetence of new users towards available items, and appeal of new items towards existing users have to be estimated as fast as possible. Currently, this situation is handled thanks to side information available on the users, and on the items (see [2,21]). 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 estimating the appetences of new users and the appeal of new items as fast as possible; the use of side information can be mixed with the ideas presented in this paper. Side information being unavailable, we learn a latent representation of each user and each item using the currently available ratings. As already argued by others (*e.g.* [16]), this problem fits perfectly into the sequential decision making framework, and more specifically, the bandit setting [20,10,9]. A sequential decision making problem under uncertainty faces an exploration vs. exploitation dilemma: the exploration is meant to acquire information in order to perform better subsequently by exploiting it; collecting the information has a cost that can not be merely zeroed, or simply left as an unimportant matter. However, in rather sharp contrast with the traditional bandit setting, here the set of bandits is constantly being renewed; the number of bandits is not small, though not being huge (from a few dozens to hundreds arms in general, up to dozens of millions in some applications): this makes the problem very different from the 2-armed bandit problem; we look for efficient and effective ways to address this task, since we want the proposed solution to be able to cope with real applications on the web. For obvious practical and economical reasons, the strategy can not merely consist in repeatedly presenting all available items to users until their appetences seem accurately estimated. We have to consider the problem as an exploration vs. exploitation problem in which exploration is a necessary evil to acquire information and eventually improve the performance of the recommendation system (RS for short). To summarize, we learn a latent representation of each user and each item, from which a recommendation policy is deduced, based on the available ratings. This learning process is continuous: the representation and the recommendation policy are updated regularly, as new ratings are observed, new items are introduced into the set of items, new users flow-in, and the preferences of already observed users change.

This being said, comes the problem of the objective function to optimize. Since the Netflix challenge, at least in the machine learning community, the recommendation problem is often reduced to a matrix factorization problem, performed in batch, learning on a training set, and minimizing the root mean squared error (RMSE) on a testing set. However, the RMSE comes with heavy flaws. Other objective functions have been considered to handle certain of these flaws [7,19].

Based on these ideas, our contribution in this paper is the following:

we propose an original way to handle new users and new items in recommendation systems: we cast this problem as a sequential decision making problem to be played online that selects items to recommend in order to optimize the exploration/exploitation balance; our solution is then to perform the rating matrix factorization driven by the policy of this sequential decision problem in order to focus on the most useful terms of the factorization. This is the core idea of the contributed algorithm we name BeWARE.

The reader familiar with the bandit framework can think of this work as a contextual bandit learning side information for each user and each item from the observed ratings, assuming the existence of a latent space of dimension k for both users and items. We stress the fact that learning and updating the representation of users and items at the same time recommendations are made is something very different from the traditional batch matrix factorization approach, or the traditional bandit setting.

We also introduce a methodology to use a classical partially filled rating matrices to assess the online performance of a bandit-based recommendation algorithm.

After introducing our notations in the next section, Sec. 3 briefly presents the matrix factorization approach. Sec. 4 introduces the necessary background in bandit theory. In Sec. 5 and Sec. 6, we present BeWARE considering in the case of new users and new items. Sec. 7 provides an experimental study on artificial data, and on real data. Finally, we conclude and draw some future lines of work in Sec. 8.

2 Notations and Vocabulary

$\mathbf{U}^T$ is the transpose of matrix $\mathbf{U}$, and $\mathbf{U}_i$ denotes its i^{th} row. For a vector $\mathbf{u}$ and a set of integers $\mathcal{S}$, $\mathbf{u}_\mathcal{S}$ is the sub-vector of $\mathbf{u}$ composed of the elements of $\mathbf{u}$ which indices belong to $\mathcal{S}$. Accordingly, $\mathbf{U}$ being a matrix, $\mathbf{U}_\mathcal{S}$ is the sub-matrix made of the rows of $\mathbf{U}$ which indices belong to $\mathcal{S}$. $\#\mathbf{u}$ is the number of components (dimension) of $\mathbf{u}$, and $\#\mathcal{S}$ is the number of elements of $\mathcal{S}$.

Now, we introduce a set of notations dedicated to the RS problem. As we consider a time-evolving number of users and items, we will note n the current number of users, and m the current number of items. These should be indexed by a t to denote time, though often in this paper, t is dropped to simplify the notation. Without loss of generality, we assume $n < N$ and $m < M$, that is N and M are the maximal numbers of ever seen users and items (those figures may as large as necessary). $\mathbf{R}^*$ represents the ground truth, that is the matrix of ratings. $r^*_{i,j}$ is the rating given by user i to item j. We suppose that there exists an integer k and two matrices $\mathbf{U}$ of size $N \times k$ and $\mathbf{V}$ of size $M \times k$ such that $\mathbf{R}^* = \mathbf{U}\mathbf{V}^T$. We denote $\mathcal{S}$ the set of elements that have been observed, and $\mathbf{R}$ denote the matrix s.t. $r_{i,j} = r^*_{i,j} + \eta_{i,j}$ if $(i,j) \in \mathcal{S}$, where $\eta_{i,j}$ is a noise with zero mean and finite variance. The $\eta_{i,j}$ are i.i.d. In this paper, we assume that $\mathbf{R}^*$ is fixed during all the time; at a given moment, only a submatrix made of n rows and m columns is actually useful. This part of $\mathbf{R}^*$ that is observed is increasing along time. That is, the set $\mathcal{S}$ is growing along time. $\mathcal{J}(i)$ (resp. $\mathcal{I}(j)$) denotes the set of items rated by user i (resp. the set of users who rated item j). $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ denote estimates (with the statistical meaning) of the matrices $\mathbf{U}$ and $\mathbf{V}$ respectively. $\hat{\mathbf{U}}\hat{\mathbf{V}}^T$ is denoted by $\hat{\mathbf{R}}$. We use the term "observation" to mean a triplet $(i,\, j,\, r_{i,j})$. The RS receives a stream of observations. We use the term "rating" to mean the value associated by a user to an item. It can be a rating as in the Netflix challenge, or an information meaning click or not, sale or not, . . . For the sake of legibility, in the online setting we omit the t subscript for time dependency. $\mathcal{S}$, $\hat{\mathbf{U}}$, $\hat{\mathbf{V}}$, n, m should be subscripted with t.

3 Matrix Factorization

Since the Netflix challenge [4], many works in RS have been using matrix factorization: the matrix of observed ratings is assumed to be the product of two matrices of low rank k: $\hat{\mathbf{R}} = \hat{\mathbf{U}}\hat{\mathbf{V}}^T$ [11]. $\hat{\mathbf{U}}$ is a latent representation of users, while $\hat{\mathbf{V}}$ is a latent representation of items. As most of the values of the rating matrix are unknown, the decomposition can only be done using the set of observations. The classical approach is to solve the regularized minimization problem $(\hat{\mathbf{U}}, \hat{\mathbf{V}}) \overset{def}{=} \operatorname{argmin}_{\mathbf{U},\mathbf{V}} \zeta(\mathbf{U}, \mathbf{V})$, where $\zeta(\mathbf{U}, \mathbf{V}) \overset{def}{=} \sum_{\forall(i,j)\in\mathcal{S}} \left(r_{i,j} - \mathbf{U}_i \cdot \mathbf{V}_j^T\right)^2 + \lambda \cdot \Omega(\mathbf{U}, \mathbf{V})$, in which $\lambda \in \mathbb{R}^+$ and is a regularization term. ζ is not convex. The minimization is usually performed either by stochastic gradient descent (SGD), or by alternate least squares (ALS). Solving for $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ at once being non convex, ALS iterates and at iteration, ALS alternates an optimization of $\hat{\mathbf{U}}$ keeping $\hat{\mathbf{V}}$ fixed, and an optimization of $\hat{\mathbf{V}}$ keeping $\hat{\mathbf{U}}$ fixed.

In this paper we consider ALS-WR [22] whose regularization term $\Omega(\mathbf{U}, \mathbf{V}) \overset{def}{=} \sum_i \#\mathcal{J}(i)||\mathbf{U}_i||^2 + \sum_j \#\mathcal{I}(j)||\mathbf{V}_j||^2$ depends on users and items respective importance in the matrix of ratings.

This regularization is known to have a good empirical behavior — that is limited overfitting, easy tuning of λ and k, low RMSE.

4 Bandits

Let us consider a bandit machine with m independent arms. When pulling arm j, the player receives a reward drawn from $[0, 1]$ which follows a probability distribution ν_j. Let μ_j denote the mean of ν_j, $j^* \overset{def}{=} \operatorname{argmax}_j \mu_j$ be the best arm and $\mu^* \overset{def}{=} \max_j \mu_j = \mu_{j^*}$ be the best expected reward (we assume there is only one best arm). $\{\nu_j\}$, $\{\mu_j\}$, j^* and μ^* are unknown.

A player aims at maximizing the sum of rewards collected along T consecutive pulls. More specifically, by denoting j_t the arm pulled at time t and r_t the reward obtained at time t, the player wants to maximize the cumulative reward $\text{CumRew}_T = \sum_{t=1}^{T} r_t$. At each time-step but the last one, the player faces the dilemma:

- either *exploit* by pulling the arm which seems the best according to the estimated values of the parameters;
- or *explore* to improve the estimation of the parameters of the probability distribution of an arm by pulling it.

Li *et al.* [13] extend the bandit setting to contextual arms. They assume that a vector of real features $\mathbf{v} \in \mathbb{R}^k$ is associated to each arm and that the expectation of the reward associated to an arm is $\mathbf{u}^* \cdot \mathbf{v}$, where $\mathbf{u}^*$ is an unknown vector. The algorithm handling this setting is known as LinUCB. LinUCB consists in playing the arm with the largest upper confidence bound on the expected reward:

$$j_t = \underset{j}{\operatorname{argmax}}\, \hat{\mathbf{u}}.\mathbf{v}_j^T + \alpha\sqrt{\mathbf{v}_j\mathbf{A}^{-1}\mathbf{v}_j^T},$$

where $\hat{\mathbf{u}}$ is an estimate of $\mathbf{u}^*$, α is a parameter, and $\mathbf{A} = \sum_{t'=1}^{t-1} \mathbf{v}_{j_{t'}}.\mathbf{v}_{j_{t'}}^T + \mathbf{Id}$, where $\mathbf{Id}$ is the identity matrix. Note that $\hat{\mathbf{u}}.\mathbf{v}_j^T$ corresponds to an estimate of the expected reward, while $\sqrt{\mathbf{v}_j\mathbf{A}^{-1}\mathbf{v}_j^T}$ is an optimistic correction of that estimate.

While the objective of LinUCB is to maximize the cumulative reward, theoretical results [13,1] are expressed in term of *cumulative regret* (or regret for short) $\text{Regret}_T \overset{def}{=} \sum_{t=1}^{T}(r_t^* - r_t)$, where $r_t^* = \max_j \mathbf{u}^*.\mathbf{v}_{j_t}^T$ stands for the best expected reward at time t. Hence, the regret measures how much the player loses (in expectation), in comparison to playing the optimal strategy. Standard results prove regrets of order $\tilde{O}(\sqrt{T})$ or $O(\ln T)$, depending on the assumptions on the distributions and depending on the precise analysis[1].

Of course LinUCB and other contextual bandit algorithms require the context (values of features) to be provided. In real applications this is done using side information about the items and the users [17] $-$*i.e.* expert knowledge, categorization of items, Facebook profiles of users, implicit feedback ... The core idea of this paper is to use matrix factorization techniques to build a context online using the known ratings. To this end, one assumes that the items and the arms can be represented in the same space of dimension k and assuming that the rating of user u for item v is the scalar product of u and v.

We study the introduction of new items and/or new users into the RS. This is done without using any side information on users or items.

5 BeWARE of a new user

Let us consider a particular recommendation scenario. At each time-step t,

1. a user i_t requests a recommendation to the RS,
2. the RS selects an item j_t among the set of items that have never been recommended to user i_t beforehand,
3. user i_t returns a rating $r_t = r_{i_t,j_t}$ for item j_t.

Obviously, the objective of the RS is to maximize the cumulative reward $\text{CumRew}_T = \sum_{t=1}^{T} r_t$. In the context of such a scenario, the usual matrix factorization approach of RS recommends item j_t which has the best predicted rating for user i_t. This corresponds to a pure exploitation, or greedy, strategy which is well-known to be suboptimal to optimize CumRew_T: to be optimal, the RS has to balance the exploitation and exploration.

Let us now describe the recommendation algorithm we propose at time-step t. We aim at recommending to user i_t an item j_t which leads to the best trade-off between exploration and exploitation in order to maximize CumRew_∞. We

[1] $\tilde{O}$ means O up to a logarithmic term on T.

assume that the matrix $\mathbf{R}$ is factored into $\hat{\mathbf{U}}\hat{\mathbf{V}}^T$ by ALS-WR which terminated by optimizing $\hat{\mathbf{U}}$ holding $\hat{\mathbf{V}}$ fixed. In such a context, the UCB approach is based on a confidence interval on the estimated ratings $\hat{r}_{i_t,j} = \hat{\mathbf{U}}_{i_t} \cdot \hat{\mathbf{V}}_j^T$ for any allowed item j.

We assume that we already observed a sufficient number of ratings for each item, but only a few ratings (possibly none) from user i_t. As a consequence the uncertainty on $\hat{\mathbf{U}}_{i_t}$ is much more important than on any $\hat{\mathbf{V}}_j$. In other words, the uncertainty on $\hat{r}_{i_t,j}$ mostly comes from the uncertainty on $\hat{\mathbf{U}}_{i_t}$. Let us express this uncertainty.

Let $\mathbf{u}^*$ denote the (unknown) true value of $\mathbf{U}_{i_t}$ and let us introduce the $k \times k$ matrix:

$$\mathbf{A} \overset{def}{=} (\hat{\mathbf{V}}_{\mathcal{J}(i_t)})^T \cdot \hat{\mathbf{V}}_{\mathcal{J}(i_t)} + \lambda \cdot \#\mathcal{J}(i_t) \cdot \mathbf{Id}.$$

As $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ comes from ALS-WR (which last iteration optimized $\hat{\mathbf{U}}$),

$$\hat{\mathbf{U}}_{j_t} = \mathbf{A}^{-1}\hat{\mathbf{V}}^T_{\mathcal{J}(i_t)}\mathbf{R}^T_{i_t,\mathcal{J}(i_t)}.$$

Using Azuma's inequality over the weighted sum of random variables (as introduced by [18] for linear systems), it follows that there exists a value $C \in \mathbb{R}$ such as, with probability $1 - \delta$:

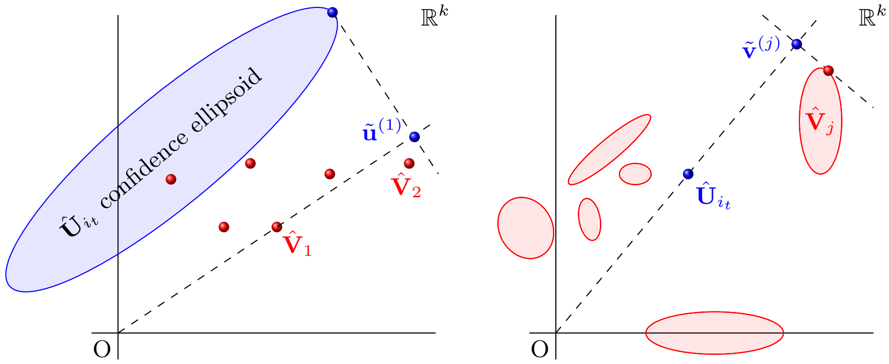
$$(\hat{\mathbf{U}}_{i_t} - \mathbf{u}^*)\mathbf{A}^{-1}(\hat{\mathbf{U}}_{i_t} - \mathbf{u}^*)^T \leq C\frac{\log(1/\delta)}{t}$$

This inequality defines the confidence bound around the estimate $\hat{\mathbf{U}}_{i_t}$ of $\mathbf{u}^*$. Therefore, a UCB strategy selects item j_t:

$$j_t \overset{def}{=} \underset{1 \leq j \leq m, j \notin \mathcal{J}(i_t)}{\operatorname{argmax}} \hat{\mathbf{U}}_{i_t} \cdot \hat{\mathbf{V}}_j^T + \alpha\sqrt{\hat{\mathbf{V}}_j\mathbf{A}^{-1}\hat{\mathbf{V}}_j^T},$$

where $\alpha \in \mathbb{R}$ is an exploration parameter to be tuned. Fig. 1(a) provides a graphical illustration of the link between the bound, and this choice of item j_t.

Our algorithm, named BeWARE.User (BeWARE which stands for "Bandit WARms-up REcommenders") is described in Alg. 1. The presentation is optimized for clarity rather than for computational efficiency. Of course, if the exploration parameter α is set to 0 BeWARE.User makes a greedy selection for the item to recommend. The estimation of the center of the ellipsoid and its size can be influenced by the use of an other regularization term. BeWARE.User uses a regularization based on ALS-WR. It is possible to replace all $\#\mathcal{J}(.)$ by 1. This amounts to the standard regularization: we call this slightly different algorithm BeWARE.ALS.User. In fact one can use any regularization as long as $\hat{\mathbf{U}}_{i_t}$ is a linear combination of observed rewards.



(a) New user.

(b) New items

Figure 1. (a) The leftmost part of this figure illustrates the use of the upper confidence ellipsoid for item selection for the new user i_t who enters the game at time t. Items and users are vectors in $\mathbb{R}^k$. (One may suppose that $k = 2$ in this figure to make it in the plane.) Red dots represent items. The blue ellipse represents the confidence ellipsoid of the vector associated to the new user. The optimistic rating of the user for an item j is the maximum dot product between $\hat{\mathbf{V}}_j$ and any point in this ellipsoid. By a simple geometrical argument based on iso-contours of the dot product, this maximum value is equal to the dot product between $\hat{\mathbf{V}}_j$ and $\tilde{\mathbf{u}}_{i_t}^{(j)}$. Optimism leads to recommend the item maximizing the dot product $\langle \tilde{\mathbf{u}}_{i_t}^{(j)}, \hat{\mathbf{V}}_j \rangle$.
(b) This figure illustrates the use of the upper confidence ellipsoid for item selection in the context of a set of new items. The setting is similar to the case of a new user except that the vector associated to the user is known (represented by a blue dot) while each item now has its confidence ellipsoids. The optimistic RS recommends the item maximizing the scalar product $\langle \hat{\mathbf{U}}_{i_t}, \tilde{\mathbf{v}}^{(j)} \rangle$.

Algorithm 1 BeWARE.User: for a user i_t, recommends an item to this user.

```
    Input: $i_t$, $\lambda$, $\alpha$
    Input/Output: $\mathbf{R}$, $\mathcal{S}$
$(\hat{\mathbf{U}}, \hat{\mathbf{V}}) \leftarrow \text{MatrixFactorization}(\mathbf{R})$
$\mathbf{A} \leftarrow (\hat{\mathbf{V}}_{\mathcal{J}(i_t)})^T \cdot \hat{\mathbf{V}}_{\mathcal{J}(i_t)} + \lambda \cdot \#\mathcal{J}(i_t) \cdot \mathbf{Id}$.
$j_t \leftarrow \underset{j \notin \mathcal{J}(i_t)}{\text{argmax}}\, \hat{\mathbf{U}}_{i_t} \cdot \hat{\mathbf{V}}_j^T + \alpha \sqrt{\hat{\mathbf{V}}_j \mathbf{A}^{-1} \hat{\mathbf{V}}_j^T}$
Recommend item $j_t$ and receive rating $r_t = r_{i_t, j_t}$
Update $\mathbf{R}$, $\mathcal{S}$
```

6 BeWARE of new items

In general, a set of new items is introduced at once, not a single item. In this case, the uncertainty is more important on items. We compute a confidence bound around the items instead of the users, assuming ALS terminates with optimizing $\hat{\mathbf{V}}$ keeping $\hat{\mathbf{U}}$ fixed. With the same criterion and regularization on $\hat{\mathbf{V}}$ as above, at timestep t:

$$\hat{\mathbf{V}}_j = \mathbf{B}(j)^{-1}(\hat{\mathbf{U}}_{\mathcal{I}(j)})^T \mathbf{R}_{\mathcal{I}(j),j},$$
$$\text{with } \mathbf{B}(j) \overset{def}{=} (\hat{\mathbf{U}}_{\mathcal{I}(j)})^T \hat{\mathbf{U}}_{\mathcal{I}(j)} + \lambda \cdot \#\mathcal{I}(j) \cdot \mathbf{Id}.$$

So the upper confidence bound of the rating for user i on item j is:

$$\hat{\mathbf{U}}_i \cdot \hat{\mathbf{V}}_j^T + \alpha \sqrt{\hat{\mathbf{U}}_j \mathbf{B}(j)^{-1} \hat{\mathbf{U}}_j^T}.$$

This leads to the algorithm BeWARE.Items presented in Alg. 2. Again, the presentation is optimized for clarity rather than for computational efficiency. BeWARE.Items can be parallelized and has the complexity of one step of ALS. Fig. 1(b) gives the geometrical intuition leading to BeWARE.Items. Again, setting $\alpha = 0$ leads to a greedy selection. The regularization (line 4) can be modified.

Algorithm 2 BeWARE.Items: for a user i_t, recommends an item to this user in the case where a set of new items is made available.

```
   Input: $i_t$, $\lambda$, $\alpha$
   Input/Output: $\mathbf{R}$, $\mathcal{S}$
1: $(\hat{\mathbf{U}}, \hat{\mathbf{V}}) \leftarrow$ MatrixFactorization($\mathbf{R}$)
2: $\forall j \notin \mathcal{J}(i_t)$, $\mathbf{B}(j) \leftarrow (\hat{\mathbf{U}}_{\mathcal{I}(j)})^T \hat{\mathbf{U}}_{\mathcal{I}(j)} + \lambda \cdot \#\mathcal{I}(j) \cdot \mathbf{Id}$
3: $j_t \leftarrow \underset{j \notin \mathcal{J}(i_t)}{\text{argmax}}\, \hat{\mathbf{U}}_{i_t}.\hat{\mathbf{V}}_j^T + \alpha \sqrt{\hat{\mathbf{U}}_{i_t} \mathbf{B}(j)^{-1} \hat{\mathbf{U}}_{i_t}^T}$
4: Recommend item $j_t$ and receive rating $r_t = r_{i_t,j_t}$
5: Update $\mathbf{R}$, and $\mathcal{S}$
```

7 Experimental Investigation

In this section we evaluate empirically BeWARE on artificial data, and on real datasets. The BeWARE algorithms are compared to:

- greedy approaches (denoted Greedy.ALS and Greedy.ALS-WR) that always choose the item with the largest current estimated value (respectively given a decomposition obtained by ALS, or by ALS-WR),
- the UCB1 approach [3] (denoted UCB.on.all.users) that considers each reward r_{i_t,j_t} as an independent realization of a distribution ν_{j_t}. In other words, UCB.on.all.users recommends an item without taking into account the information on the user requesting the recommendation.

The comparison to greedy selection highlights the needs of exploration to have an optimal algorithm in the online context. The comparison to UCB.on.all.users assesses the benefit of personalizing recommendations.

7.1 Experimental Setting

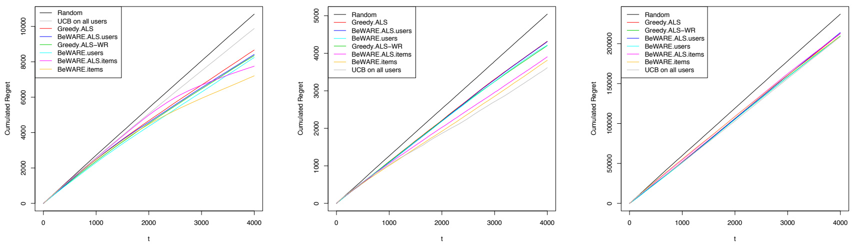
For each dataset, each algorithm starts with an empty $\mathbf{R}$ matrix of 100 items and 200 users. Then, the evaluation goes like this:

1. select a user uniformly at random among those who have not yet rated all the items,
2. request his favorite item among those he has not yet rated,
3. compute the immediate regret (the difference of rating between the best not yet selected item and the one selected by the algorithm),
4. iterate until all users have rated all items.

The difficulty with real datasets is that the ground truth is unknown, and actually, only a very small fraction of ratings is known. This makes the evaluation of algorithms uneasy. To overcome these difficulties, we also provide a comparison of the algorithms considering an artificial problem based on a ground truth matrix $\mathbf{R}^*$ considering m users and n items. This matrix is generated as in [6]. Each item belongs to either one of k genres, and each user belongs to either one of l types. For each item j of genre a and each user i of type b, $r^*_{i,j} = p_{a,b}$ is the ground truth rating of item j by user i, where $p_{a,b}$ is drawn uniformly at random in the set $\{1, 2, 3, 4, 5\}$. The observed rating $r_{i,j}$ is a noisy value of $r^*_{i,j}$: $r_{i,j} = r^*_{i,j} + \mathcal{N}(0, 0.5)$.

We also consider real datasets, the NetFlix dataset [4] and the Yahoo!Music dataset [8]. Of course, the major issue with real data is that there is no dataset with a complete matrix, which means we do no longer have access to the ground truth $\mathbf{R}^*$, which makes the evaluation of algorithms more complex. This issue is usually solved in the bandit literature by using a method based on reject sampling [14]. For a well constructed dataset, this kind of estimators has no bias and a known bound on the decrease of the error rate [12]. For all the algorithms, we restrict the possible choices for a user at time-step t to the items with a known rating in the dataset. However, a minimum amount of ratings per user is needed to be able to have a meaningful comparison of the algorithms (otherwise, a random strategy is the only reasonable one). As a consequence, with both datasets, we focus on the 5000 heaviest users for the top ∼250 movies/songs. This leads to a matrix $\widetilde{\mathbf{R}}^*$ with only 10% to 20% of missing ratings. We insist on the fact that this is necessary for performance evaluation of the algorithms; obviously, this is not required to use the algorithms on a live RS.

We would like to advertize that this experimental methodology has a unique feature: this methodology allows us to turn any matrix of ratings into an online problem which can be used to test bandit recommendation algorithms. We think that this methodology is an other contribution of this paper.



(a) Artificial dataset. (b) Netflix dataset. (c) Yahoo!Music dataset.

Figure 2. Cumulated regret (the lower, the better) for a set of 100 new items and 200 users with no prior information. Figures are averaged over 20 runs (for Netflix and artificial data, $k = 5$, $\lambda = 0.05$, $\alpha = 0.12$ whereas for Yahoo!Music, $k = 8$, $\lambda = 0.2$, $\alpha = 0.05$). On the artificial dataset (a), BeWARE.items is better than the other strategies in terms of regret. On the Netflix dataset (b), `UCB on all users` is the best approach and BeWARE.items is the second best. On the Yahoo!Music dataset (c), BeWARE.items, Greedy.ALS-WR and UCB all 3 lead to similar performances.

7.2 Experimental Results

Figures 2(a) and 2(b) show that given a fixed factorization method, BeWARE strategies outperform greedy item selection. Looking more closely at the results, BeWARE.items performs better than BeWARE.user, and BeWARE.user is the only BeWARE strategy beaten by its greedy counterpart (Greedy.ALS-WR) on the Netflix dataset. These results demonstrate that an online strategy has to care about exploration to tend towards optimality.

While UCB.on.all.users is almost the worst approach on artificial data (Fig. 2(a)), it surprisingly performs better than all other approaches on the Netflix dataset. We feel that this difference is strongly related to the preprocessing of the Netflix dataset we have done to be able to follow the experimental protocol (and have an evaluation at all). By focusing on the top ∼250 movies, we only keep blockbusters that everyone enjoys. With that particular subset of movies, there is no need to adapt the recommendation user per user. As a consequence, UCB.on.all.users suffers a smaller regret than other strategies, as it considers users as n independent realizations of the same distribution. It is worth noting that the regret of UCB.on.all.users would increase with the number of items while the regret of BeWARE scales with the dimensionality of the factorization, which makes BeWARE a better candidates for real applications with much more items to deal with.

Last, on the Yahoo! Music datatset (Fig. 2(c)), all algorithms suffer the same regret.

7.3 Discussion

In a real setting, BeWARE.items has a desirable property: it tends to favor new items with regards to older ones because they simply have less ratings than the others, hence larger confidence bounds. So the algorithm gives them a boost which is exactly what a webstore is willing. Moreover, the RS then uses at its best the novelty effect associated to new items. This natural attraction of users for new items can be very strong as it has been shown during the Exploration & Exploitation challenge at ICML'2012 which was won by a context free algorithm [15].

The computational cost of BeWARE is the same as doing an additional step of alternate least squares; moreover some intermediate calculations of the QR factorization can be re-used to speed up the computation. So the total cost of BeWARE.Items is almost the same as ALS-WR. Even better, while the online setting requires to recompute the factorization at each time-step, this factorization changes only slightly from one iteration to the other. As a consequence, only a few ALS-WR iterations are needed to update the factorization. Overall the computational cost remains reasonable even in a real application.

8 Conclusion and Future Work

In this paper, we have bridged matrix factorization with bandits to address in a principled way the balance between exploration and exploitation faced by online recommendations systems when considering new users or new items. We think that this contribution is conceptually rich, and opens ways to many different studies. We showed on large, publicly available datasets that this approach is also effective, leading to efficient algorithms able to work online, under the expected computational constraints of such systems. Furthermore, the algorithms are quite easy to implement.

Many extensions are currently under study. First, we work on extending these algorithms to use contextual information about users, and items. This will require combining the similarity measure with confidence bounds; this might be translated into a Bayesian prior. We also want to analyze regret bound for large enough number of items and users. This part can be tricky as LinUCB still does not have a full formal analysis, though some insights are available in [1].

An other important point is to work on the recommendation of several items at once and get feedback only for the one. There has been some work in the non contextual bandits on this point [5].

Finally, we plan to combine confidence ellipsoid about both users and items. We feel that such a combination has low odds of providing better results for real applications, but it is interesting from a theoretical perspective, and should lead to even better results on artificial problems.

Acknowledgements: authors acknowledge the support of INRIA, and the stimulating environment of the research group SequeL.

References

1. Y. Abbasi-yadkori, D. Pal, and Cs. Szepesvari. Improved algorithms for linear stochastic bandits. In *Proc. NIPS*, pages 2312–2320, 2011.
2. D. Agarwal, B-Ch. Chen, P. Elango, N. Motgi, S-T. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *Proc. NIPS*, pages 17–24, 2008.
3. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, May 2002.
4. J. Bennett, S. Lanning, and Netflix. The Netflix prize. In *KDD Cup and Workshop*, 2007.
5. N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *J. Comput. Syst. Sci.*, 78(5):1404–1422, 2012.
6. Sourav Chatterjee. Matrix estimation by universal singular value thresholding. *pre-print*, 2012. http://arxiv.org/abs/1212.1247.
7. Ch. Dhanjal, R. Gaudel, and S. Clémençon. Collaborative filtering with localised ranking. In *Proc. AAAI*, 2015.
8. G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The Yahoo! music dataset and kdd-cup'11. In *Proceedings of KDD Cup*, 2011.
9. S. Feldman. Personalization with contextual bandits. http://engineering.richrelevance.com/author/sergey-feldman/.
10. P. Kohli, M. Salek, and G. Stoddard. A fast bandit algorithm for recommendations to users with heterogeneous tastes. In *Proc. AAAI*, pages 1135–1141, 2013.
11. Y. Koren, R. Bell, and Ch. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.
12. J. Langford, A. Strehl, and J. Wortman. Exploration scavenging. In *Proc. ICML*, pages 528–535. Omnipress, 2008.
13. L. Li, W. Chu, J. Langford, and R.E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. WWW*, pages 661–670, New York, NY, USA, 2010. ACM.
14. L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. WSDM*, pages 297–306. ACM, 2011.
15. J. Mary, A. Garivier, L. Li, R. Munos, O. Nicol, R. Ortner, and P. Preux. Icml exploration and exploitation 3 - new challenges, 2012.
16. G. Shani, D. Heckerman, and I. Brafman Ronen. An MDP-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, September 2005.
17. P.K. Shivaswamy and Th. Joachims. Online learning with preference feedback, 2011. NIPS workshop on choice models and preference learning.
18. Th. J. Walsh, I. Szita, C. Diuk, and Michael L. Littman. Exploring compact reinforcement-learning representations with linear regression. *CoRR*, abs/1205.2606, 2012.
19. J. Weston, H. Yee, and R.J. Weiss. Learning to rank recommendations with the k-order statistic loss. In *Proc. of RecSys*, pages 245–248. ACM, 2013.
20. J. M. White. *Bandit algorithms for website optimization.* O'Reilly, 2012.
21. Y. Yue, S. A. Hong, and C. Guestrin. Hierarchical exploration for accelerating contextual bandits. In *Proc. ICML*, pages 1895–1902, 2012.
22. Y. Zhou, D. Wilkinson, R. Schreiber, and R. 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.