Large-scale Bandit Recommender System

Frédéric Guillou[1], Romaric Gaudel[2], and Philippe Preux[2]

[1] Inria, Univ. Lille, CNRS, France
frederic.guillou@inria.fr
[2] Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL, Lille, France
{romaric.gaudel,philippe.preux}@univ-lille3.fr

Abstract. The main target of Recommender Systems (RS) is to propose to users one or several items in which they might be interested. However, as users provide more feedback, the recommendation process has to take these new data into consideration. The necessity of this update phase makes recommendation an intrinsically sequential task. A few approaches were recently proposed to address this issue, but they do not meet the need to scale up to real life applications. In this paper, we present a Collaborative Filtering RS method based on Matrix Factorization and Multi-Armed Bandits. This approach aims at good recommendations with a narrow computation time. Several experiments on large datasets show that the proposed approach performs personalized recommendations in less than a millisecond per recommendation.

1 Introduction

We consider Collaborative Filtering approaches based on Matrix Completion. Such Recommender Systems (RS) recommend items to users and adapt the recommendation to user tastes as inferred from past user behavior. Depending on the application, items can be ads, news, music, videos, movies, etc. This recommendation setting was popularized by the Netflix challenge [4]. Most of approaches model the taste of each user regarding each item as a matrix $\mathbf{R}^*$ [13]. In that matrix, only a few entries are known: the ones corresponding to feedback gathered in the past. In such a context, the RS recovers unknown values in $\mathbf{R}^*$ and the evaluation is done by splitting log data into two parts: the first part (aka. train-set) is used to define the training matrix which is completed by the RS algorithm; the second part (aka. test-set) is used to measure the quality of the matrix returned by the RS. Common measures of that quality are Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) on the test-set. While such a static batch evaluation makes sense to measure the quality of the matrix-completion step of Collaborative Filtering, it does not evaluate the quality of the final recommendation. A Collaborative Filtering based RS works in reality in a sequential manner and loops through the following steps:

1. Build a model of the users' tastes based on past feedback;
2. Recommend items to users using this model;
3. Gather feedback from users about recommended products.

Note that the model built at step 1 heavily depends on the feedback gathered at previous iterations. This feedback only exists for items which were chosen by the model itself. As such, at step 2, one faces the exploration/exploitation dilemma: either (i) recommend an item which led to the best feedback in the past (aka exploit) or (ii) recommend an item which hopefully brings information on the user's taste (aka explore). This dilemma is the core point of Multi-Armed Bandit Theory [3]. It is already studied in a sub-field of RS which has access to a representation of the context (the user, the webpage ...) [25]. Typical applications are the selection of news or ads to show on a web-page. The corresponding RS builds upon contextual bandits which are supported by strong theoretical results. In contrast with these studies, we focus on the setting where these representations are unknown and have to be inferred solely from users' feedback. In particular, we want to emphasize that we do not use any side information, neither about users, nor items. That field of research is almost empty and the few attempts therein leave out computational complexity constraints [12].

Our paper introduces the first sequential Collaborative Filtering based RS which (i) makes a good trade-off between exploration and exploitation and (ii) is able to properly scale. Extensive experiments are conducted on real word datasets of millions of ratings and convey three main conclusions: first, they highlight the need for a trade-off between exploration and exploitation for a RS to be optimal; second, they demonstrate that the proposed approach brings such optimal trade-off; third, they exhibit that the proposed approach can perform good recommendations in less than a millisecond per recommendation, the time a RS has to make its recommendation in a real, live system.

After introducing the setting in the next section, Sec. 3 recalls the standard matrix factorization approach and introduces the necessary background in bandit theory. In Sec. 4, we introduce an algorithm which fully takes into account the sequential aspect of RS. Sec. 5 provides an experimental study on real datasets. Finally, Sec. 6 reviews research results related to the proposed approach, and we conclude and draw some future lines of work in Sec. 7.

2 Sequential recommendation

Let us focus on a particular recommendation scenario, which illustrates more accurately how typical Recommender Systems work. We consider N users, M items, and the unknown matrix $\mathbf{R}^*$ of size $N \times M$ such that $r^*_{i,j}$ is the taste of user i with regards to item j. At each time-step t,

1. a user i_t requests a recommendation from the RS,
2. the RS selects an item j_t among the set of available items,
3. user i_t returns a feedback $r_t \sim \mathcal{D}(r^*_{i_t,j_t})$ for item j_t.

In current paper we assume the mean of distribution $\mathcal{D}(r^*_{i_t,j_t})$ to be $r^*_{i_t,j_t}$. See [14] for an example of a more refined observation/noise model.

We refer to applications where the feedback r_t corresponds to the quantity that has to be optimized, aka. *the reward*. In such a context, the aim of the RS

is to maximize the reward accumulated along time-steps $\text{CumRew}_T = \sum_{t=1}^{T} r_t$, or to minimize the *pseudo-regret* $\mathcal{R}_T$ (1) which measures how much the system loses in average by recommending a sub-optimal item:

$$\mathcal{R}_T = \sum_{t=1}^{T} \max_j r^*_{i_t,j} - \mathbb{E}[r_t] = \sum_{t=1}^{T} \max_j r^*_{i_t,j} - r^*_{i_t,j_t}. \quad (1)$$

Along the paper, we use the following notations. We denote $\mathbf{R}_t$ the partially known $N \times M$ matrix such that $r_{i_s,j_s} = r_s$ for any $s \leqslant t$. We note $\mathcal{S}_t$ the set of known entries of $\mathbf{R}_t$ and $\mathcal{I}_t(i)$ (respectively $\mathcal{J}_t(j)$) the set of items j (resp. users i) for which $(i,j) \in \mathcal{S}_t$. For the sake of readability, the subscript t is omitted in the following. Finally, for any matrix $\mathbf{M}$, we denote $\mathbf{M}_i$ the i-th row of $\mathbf{M}$.

3 Building blocks

We introduce in Sec. 4 a RS which handles sequential recommendations. This RS is composed of two main ingredients: (i) a model to infer an estimate $\widehat{\mathbf{R}}^*$ of the matrix $\mathbf{R}^*$ from known values in $\mathbf{R}$, and (ii) a strategy to choose the item to recommend given $\widehat{\mathbf{R}}^*$. This strategy aims at balancing exploration and exploitation. In this section we go over state of the art approaches for both tasks.

3.1 Matrix Factorization

Since the Netflix challenge [4], many works on RS focus on Matrix Factorization [13]: the unknown matrix $\mathbf{R}^*$ is assumed to be of low rank. Namely, there exist $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{R}^* = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ is a matrix of size $N \times k$ representing users features, $\mathbf{V}$ is a matrix of size $M \times k$ representing items features, k is the rank of $\mathbf{R}^*$, and $k \ll \max(N, M)$. Thereafter, the estimator of $\mathbf{R}^*$ is defined as

$$\widehat{\mathbf{R}}^* \overset{def}{=} \widehat{\mathbf{U}}\widehat{\mathbf{V}}^T, \text{ s.t. } (\widehat{\mathbf{U}}, \widehat{\mathbf{V}}) = \underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} \sum_{\forall (i,j) \in \mathcal{S}} \left(r_{i,j} - \mathbf{U}_i \mathbf{V}_j^T\right)^2 + \lambda \cdot \Omega(\mathbf{U}, \mathbf{V}), \quad (2)$$

in which $\lambda \in \mathbb{R}^+$, and the usual regularization term $\Omega(\mathbf{U}, \mathbf{V})$ is $||\mathbf{U}||^2 + ||\mathbf{V}||^2$. Eq. (2) corresponds to a non-convex optimization problem. The minimization is usually performed either by stochastic gradient descent (SGD), or by alternating least squares (ALS). As an example, ALS-WR [29] regularizes users and items according to their respective importance in the matrix of ratings: $\Omega(\mathbf{U}, \mathbf{V}) = \sum_i \#\mathcal{I}(i)||\mathbf{U}_i||^2 + \sum_j \#\mathcal{J}(j)||\mathbf{V}_j||^2$.

3.2 Multi-Armed Bandits

A RS works in a sequential context. As a consequence, while the recommendation made at time-step t aims at collecting a good reward at the present time, it affects the information that is collected, and therefore also the future recommendations and rewards. Specifically, in the context of sequential decision under uncertainty

problems, an algorithm which focuses only on short term reward loses w.r.t. expected long term reward. This section recalls standard strategies to handle this short term vs. long term dilemma. For ease of understanding, the setting used in this section is much simpler than the one faced in our paper.

We consider the Multi-Armed Bandits (MAB) setting [3]: we face a bandit machine with M independent arms. At each time-step, we pull an arm j and receive a reward drawn from $[0, 1]$ which follows a probability distribution ν_j. Let μ_j denote the mean of ν_j, $j^* = \text{argmax}_j \mu_j$ be the best arm and $\mu^* = \max_j \mu_j = \mu_{j^*}$ be the best expected reward. The parameters $\{\nu_j\}$, $\{\mu_j\}$, j^* and μ^* are unknown.

We play T consecutive times and aim at minimizing the *pseudo-regret* $\mathcal{R}_T = \sum_{t=1}^{T} \mu^* - \mu_{j_t}$, where j_t denotes the arm pulled at time-step t. As the parameters are unknown, at each time-step, we face the dilemma: either (i) focus on short-term reward (aka. *exploit*) by pulling the arm which was the best at previous time-steps, or (ii) focus on long-term reward (aka. *explore*) by pulling an arm to improve the estimation of its parameters. Neither of these strategies is optimal. To be optimal, a strategy has to balance exploration and exploitation.

P. Auer [3] proposes a strategy based on an upper confidence bound (UCB1) to handle this exploration/exploitation dilemma. UCB1 balances exploration and exploitation by playing the arm $j_t = \text{argmax}_j \hat{\mu}_j(t) + \sqrt{\frac{2 \ln t}{T_j(t)}}$, where $T_j(t)$ corresponds to the number of pulls of arm j since the first time-step and $\hat{\mu}_j(t)$ denotes the empirical mean reward incurred from arm j up to time t. This equation embodies the exploration/exploitation trade-off: while $\hat{\mu}_j(t)$ promotes exploitation of the arm which looks optimal, the second term of the sum promotes exploration of less played arms. Other flavors of UCB-like algorithms [2,10,18] aim at a strategy closer to the optimal one or at a strategy which benefits from constraints on the reward distribution.

ε_n-greedy is another efficient approach to balance exploration and exploitation [3]. It consists in playing the greedy strategy ($j_t = \text{argmax}_j \hat{\mu}_j(t)$) with probability $1 - \varepsilon_t$ and in pulling an arm at random otherwise. Parameter ε_t is set to α/t with α a constant, so that there is more exploration at the beginning of the evaluation and then a decreasing chance to fall on an exploration step.

4 Explore-exploit Recommender System

This section introduces a RS which handles the sequential aspect of recommendation. More specifically, the proposed approach works in the context presented in Sec. 2 and aims at minimizing the pseudo-regret $\mathcal{R}_T$. As needed, the proposed approach balances exploration and exploitation.

Named SeALS (for *Sequential ALS-WR*), our approach is described in Alg. 1. It builds upon ALS-WR Matrix Completion approach and ε_n-greedy strategy to tackle the exploration/exploitation dilemma. At time-step t, for a given user i_t, ALS-WR associates an expected reward $\hat{r}_{i_t,j}$ to each item j. Then the item to recommend j_t is chosen by an ε_n-greedy strategy.

Algorithm 1 SeALS: recommend in a sequential context

Input: T_u, p, λ, α **Input/Output:** $\mathbf{R}$, $\mathcal{S}$

$(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}) \leftarrow$ ALS-WR$(\mathbf{R}, \mathcal{S}, \lambda)$

for $t = 1,\ 2,\ \ldots$ **do**

get user i_t and set $\mathcal{A}_t$ of allowed items

$$j_t \leftarrow \begin{cases} \text{argmax}_{j \in \mathcal{J}_t} \widehat{\mathbf{U}}_{i_t} \widehat{\mathbf{V}}_j^T & \text{, with probability } 1 - \min(\alpha/t, 1) \\ \text{random}(j \in \mathcal{A}_t) & \text{, with probability } \min(\alpha/t, 1) \end{cases}$$

recommend item j_t and receive rating $r_t = r_{i_t, j_t}$

update $\mathbf{R}$ and $\mathcal{S}$

if $t \equiv 0 \mod T_u$ **then** $(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}) \leftarrow$ mBALS-WR$(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}, \mathbf{R}, \mathcal{S}, \lambda, p)$ **end if**

end for

Algorithm 2 mBALS-WR: mini-batch version of ALS-WR

Input: $\mathbf{R}$, $\mathcal{S}$, λ, p, **Input/Output:** $\widehat{\mathbf{U}}$, $\widehat{\mathbf{V}}$

Sample randomly p% of all users in a list l_{users}

Sample randomly p% of all items in a list l_{items}

$$\forall i \in l_{users},\ \widehat{\mathbf{U}}_i \leftarrow \text{argmin}_{\mathbf{U}} \sum_{j \in \mathcal{J}_t(i)} \left(r_{i,j} - \mathbf{U} \widehat{\mathbf{V}}_j^T \right)^2 + \lambda \cdot \#\mathcal{I}_t(i) \|\mathbf{U}\|$$

$$\forall j \in l_{items},\ \widehat{\mathbf{V}}_j \leftarrow \text{argmin}_{\mathbf{V}} \sum_{i \in \mathcal{I}_t(j)} \left(r_{i,j} - \widehat{\mathbf{U}}_i \mathbf{V}^T \right)^2 + \lambda \cdot \#\mathcal{J}_t(j) \|\mathbf{V}\|$$

Obviously, ALS-WR requires too large computation times to be run at each time-step to recompute user and item features. A solution consists in running ALS-WR every T_u time-steps. While such a strategy works well when T_u is small enough, $\mathcal{R}_T$ drastically increases otherwise (see Sec. 5.3). Taking inspiration from stochastic gradient approaches [6], we solve that problem by designing a mini-batch version of ALS-WR, denoted mBALS-WR (see Alg. 2).

mBALS-WR is designed to work in a sequential context where the matrix decomposition slightly changes between two consecutive calls. As a consequence, there are three main differences between ALS-WR and mBALS-WR. First, instead of computing $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ from scratch, mBALS-WR updates both matrices. Second, mBALS-WR performs only one pass on the data. And third, mBALS-WR updates only a fixed percentage of the line of $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$. When the parameter $p = 100\%$, mBALS-WR is a one-pass ALS-WR.

The main advantage of mBALS-WR is in spreading the computing budget along time-steps which means $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ are more often up to date. On one hand, ALS-WR consumes a huge computing budget every thousands of time-steps; in between two updates, it selects the items to recommend based on an outdated decomposition. On the other hand, mBALS-WR makes frequent updates of the decomposition. In the extreme case, updates can be done at each time-step.

5 Experimental investigation

In this section, we empirically evaluate the algorithms in the sequential setting on large real-world datasets. Two series of experiments emphasize two aspects

Table 1. Dataset characteristics.

	Movielens1M	Movielens20M	Douban	Yahoo!
Number of users	6,040	138,493	129,490	1,065,258
Number of items	3,706	26,744	58,541	98,209
Number of ratings	1,000,209	20,000,263	16,830,839	109,485,914

of the sequential RS: Sec. 5.2 shows that exploration improves the quality of the RS model and compares our model with several baselines, while Sec. 5.3 focuses on the influence of the method updating the matrix model on the pseudo-regret and the running time.

5.1 Experimental setting and remarks

We use the same setting as the one used in the paper by Kawale et al. [12]. For each dataset, we start with an empty matrix $\mathbf{R}$ to simulate an extreme cold-start scenario where no information is available at all. Then, for a given number of time-steps, we loop on the following procedure:

1. we select a user i_t uniformly at random,
2. the algorithm chooses an item j_t to recommend,
3. we reveal the value of $r_t = r^*_{i_t,j_t}$ and increment the pseudo-regret score (1).

We assume that $\mathbf{R}^*$ corresponds to the values in the dataset. To compute the regret, the maximization term $\max_j r^*_{i_t,j}$ is taken w.r.t. the known values. Note that it is allowed to play an arm several times: once an item has been recommended, it is not discarded from the set of future possible recommendations.

We consider four real-world datasets for our experiments: Movielens1M/20M [11] and Douban [19] for datasets on movies, and Yahoo! Music user ratings of musical artists[3]. Characteristics of these datasets are reported in Table 1. For the Yahoo! dataset, we remove users with less than 20 ratings.

Some difficulties arise when using real datasets: in most cases, the ground truth is unknown, and only a very small fraction of ratings is known since users gave ratings only to items they have purchased/listened/watched. This makes the evaluation of algorithms uneasy considering we need in advance the reward of items we include in the list of possible recommendations. This is the case in our experiments, as we do not have access to the full matrix $\mathbf{R}$ in all datasets.

This issue is solved in the case of contextual bandits by using reject sampling [17]: the algorithm chooses an arm (item), and if the arm does not appear in logged data, the choice is discarded, as if the algorithm had not been called at all. For a well collected dataset, this estimator has no bias and has a known bound on the decrease of the error rate [15]. With our setting, we need no more to rely on reject sampling: we restrict the possible choices for a user at time-step t to the items with a known rating in the dataset.

The SeALS algorithm is compared to the following baselines:

[3] https://webscope.sandbox.yahoo.com/

- Random: at each iteration, a random item is recommended to the user.
- Popular: this approach assumes we know the most popular items based on the ground truth matrix. At each iteration, the most popular item (restricted to the items rated by the user on the dataset) is recommended. This is a strong baseline as it knows beforehand which items have the highest average ratings.
- UCB1: this bandit approach considers each reward r_{i_t,j_t} as an independent realization of a distribution ν_{j_t}. In other words, it recommends an item without taking into account the identity of the user requesting the recommendation.
- PTS [12]: this approach builds upon a statistical model of the matrix $\mathbf{R}$. The recommendations are done after a Thompson Sampling strategy [7] which is implemented with a Particle Filter. We present the results obtained with the non-Bayesian version, as it obtains very similar results with the Bayesian one.

5.2 Impact of exploration

The first set of experiments compares two strategies to recommend an item: SeALS with $\alpha > 0$, and SeALS with $\alpha = 0$ (denoted Greedy) which corresponds to the greedy strategy. Both strategies use the maximum possible value for p ($p = 100\%$), which means we update all the users and items every T_u time-steps. The value of T_u used is the same for SeALS and Greedy, and α is set to 2,000 for Movielens1M, 10,000 for Douban and Movielens20M and 250,000 for Yahoo!. We set $\lambda = 0.1$ for Greedy and $\lambda = 0.15$ for SeALS. Parameter k is set to 15.

We also compare these two approaches with PTS. By fixing the value of the parameters of PTS as mentioned in [12] (30 particles and $k = 2$), the experimental results we obtain are not as good as the ones presented in that paper. However, we recover results similar to the ones presented in [12] by setting k to 15. So, we use that value of k for the results of the PTS approach we are displaying.

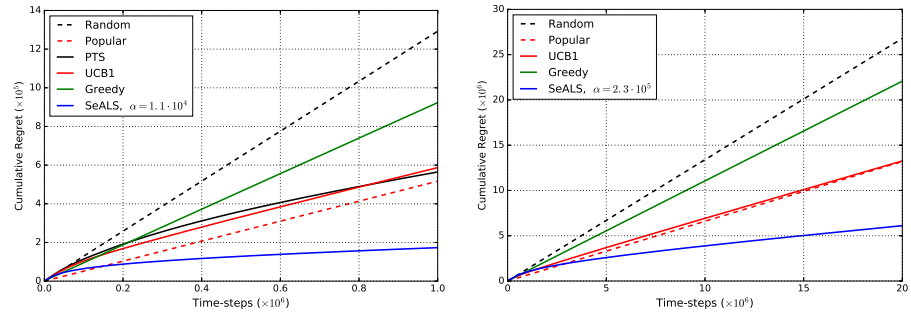
Fig. 1 displays the pseudo-regret $\mathcal{R}_T$ obtained by the Recommender System after a given number of iterations (all results are averaged over 50 runs).

Results on all datasets demonstrate the need of exploration during the recommendation process: by properly fixing α, SeALS gets lower pseudo-regret value than Greedy. SeALS also obtains the best results on all datasets.

The PTS method which also tackles the exploration/exploitation dilemma appears only on the Movielens1M evaluation as this method does not scale well on large datasets (see Sec. 5.3 for the running time of PTS on Movielens1M). However, it is important to note that on the original PTS paper, this approach performs only comparably or slightly better than the Popular baseline on the evaluation provided on all small datasets, while our approach consistently performs much better than this baseline. One can reasonably assume that SeALS would perform better than PTS even if the latter one didn't have a scaling issue.

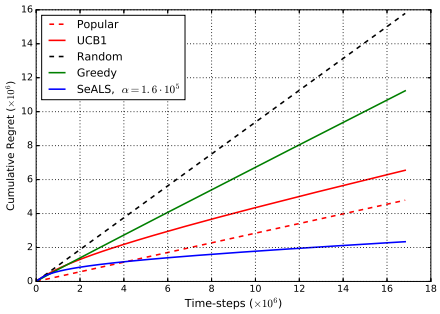
5.3 Impact of the update strategy

To evaluate the impact of the method used to update the model as well as the period at which updates take place, we set up a different evaluation display for

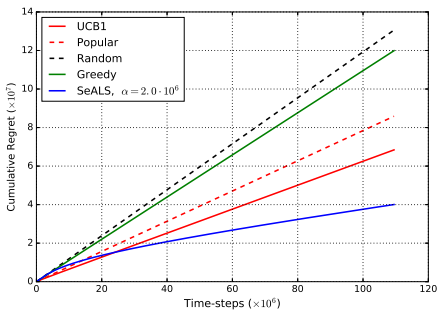


(a) Movielens1M

(b) Movielens20M



(c) Douban



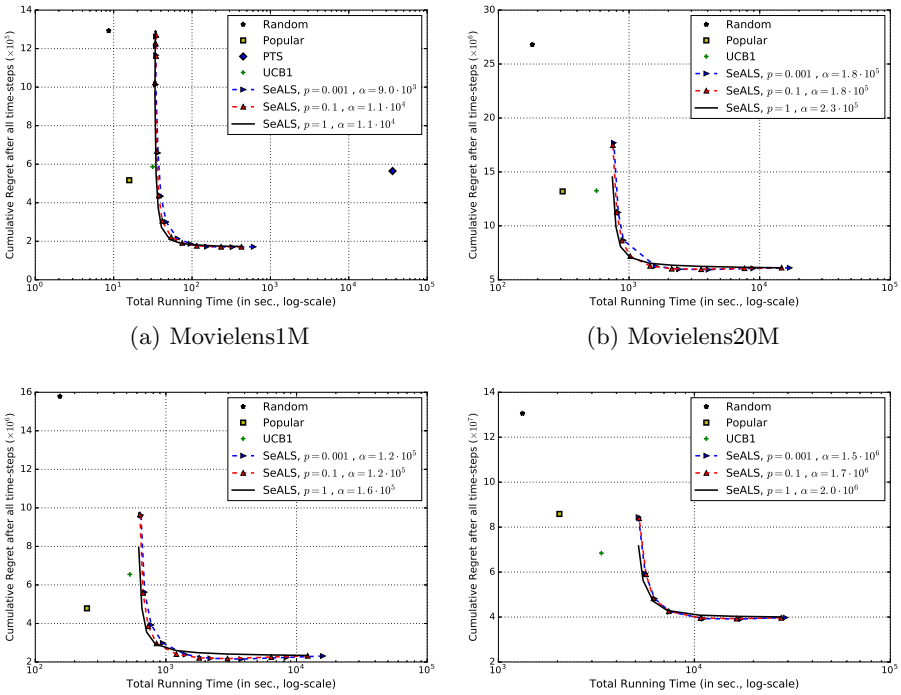
(d) Yahoo

Fig. 1. Impact of exploration on four datasets.

this second experiment. For each RS, we run experiments as presented in Sec. 5.1, and we store the final running time as well as the pseudo-regret at the end of all iterations. Such evaluation methodology allows finding which size of mini-batch and update period leads to the best trade-off between the final pseudo-regret score and the running time. This is an important point as a recommendation should both (i) be accurate and (ii) be quickly provided to the user in a real-word RS.

Fig. 2 displays the results of this experiment. Each curve corresponds to a fixed size of mini-batch p, and every point of a same curve represents a specific value of the update period T_u. A point located at a high running time results from a small value of T_u (meaning the model was updated very often). For SeALS with $p = 100\%$, the period T_u of the updates varies in the range $[2\,10^3; 2\,10^5]$ for Movielens1M, $[10^4; 5\,10^6]$ for Douban and Movielens20M, and $[2.5\,10^5; 10^7]$ for Yahoo!. For $p = 10\%$ and $p = 0.1\%$, the considered periods are the same ones as for $p = 100\%$, but respectively divided by 10 and 10^3 in order to obtain comparable running times. Indeed, since we update a smaller portion of the matrix, it is possible to run this update more often and choose a small period T_u.

For each value of p, we display the curve with the value of α (for the exploration) which reaches the lowest pseudo-regret.



(a) Movielens1M (b) Movielens20M

(c) Douban (d) Yahoo

Fig. 2. Impact of the update strategy on four datasets.

Three main conclusions are drawn from this experiment: first, the results on Movielens1M highlight the non-scalability of the PTS algorithm, which takes several hours to complete 1 million iterations while SeALS only takes a few minutes. PTS does not seem to be an appropriate algorithm to provide quick recommendations as it takes too long updating the model. Second, on each dataset, each curve concerning SeALS quickly decreases: there is a rapid transition from a poor score to a good pseudo-regret. This means finding the appropriate period of update is sufficient to obtain a good RS. Third, on the large datasets, decreasing the portion of the users and items to update with a smaller p results in a worse score when using large update periods, but leads to a slightly better trade-off between the pseudo-regret and the running time at some point, when the updates are happening more often. One has to notice the best results for smaller values of p are obtained with a smaller value of α, which implies less steps of exploration has been done. We guess some sort of exploration is added

in the system by not updating each user and each item at each time-step as it happens when $p = 100\%$ is chosen: while the whole model is not updated, it does not play optimally, which means it explores.

6 Related work

To the best of our knowledge, only few papers consider a RS setting similar to the one presented in the current paper, where exploration/exploitation dilemma is tackled with ratings only [28,27,21,12,20]. [20] focus on a different recommendation setting where an item can be recommended only one time per user. Their approach builds upon ALS Matrix Factorization framework and extends linear bandits [16] to handle the exploration/exploitation dilemma. The use of linear bandit framework prevents this approach from scaling.

Other papers propose a solution based on a Bayesian model. The approach PTS, introduced in [12], tackles the exact same problem as our paper. PTS is based on Thompson Sampling [7] and Particle Filtering. However, their approach requires a huge computing time which scales with k^3. As a consequence, [12] only provides experiments on "small" datasets, with $k = 2$. SeALS is the first approach which both tackles the exploration/exploitation dilemma and scales up well to large datasets and high number of recommendations, while building an accurate representation of users and items ($k \gg 2$).

Contrary to the non-contextual case, the exploration/exploitation dilemma is already well-studied in the field of RS which has access to a representation of the context [25]. Compared to them, we focus on the setting where these representations are unknown and have to be inferred from users feedback.

Some papers focusing on the cold-start setting also focuses on the need for exploration [1,5]: the goal in this case is to deal with new users or new items. While some approaches look at external information on the user [1], some papers rewrite the problem as an Active Learning problem [5]: perform recommendation in order to gather information on the user as fast as possible. Targeted applications would first "explore" the user and then "exploit" him. Unlike Active Learning strategies, (i) we spread the cost of exploration along time, and (ii) we handle the need for a never-ending exploration to reach optimality [3].

Finally, some researches focus on a ranking algorithm instead of trying to target a good RMSE score. Cremonesi et al. [8] compare state of the art RS algorithms with respect to a rank-based scoring and shows that winning algorithms are not the ones which reach the smallest RMSE score. Following the same guideline, [22,24,26] propose RS which directly target a good ranking of the top items instead of a full-completion of the matrix. During the training phase, they replace the L2 loss of Eq. (2) by rank-based losses (AUC, MRR, NDCG...). While targeting a rank-based measure during the training phase could increase the accuracy of the RS, [9] and [23] show that the cumulative reward/regret is the unique good metric to evaluate the RS algorithm.

7 Conclusion and future work

In this paper we handle Recommender Systems based on Matrix Completion in the suitable context: an endless loop which alternates (i) learning of the model and (ii) recommendations given the model. Our proposed approach, SeALS, meets both challenges which arise in such a context. First, SeALS handles both short-term and long-term reward by balancing exploration and exploitation. Second, SeALS handles constraints on computing budget by adapting mini-batch strategy to alternating least square optimization. Experiments on real-life datasets show that (i) exploration is a necessary evil to acquire information and eventually improve the performance of the RS, and (ii) SeALS runs in an acceptable amount of time (less than a millisecond per iteration). SeALS paves the way to many extensions. SeALS builds upon ALS-WR which is intrinsically parallel; implementations of SeALS in real-life systems should benefit from parallel computing. SeALS could also be extended to mix user feedback and contextual information. All in all, we hope SeALS is revealing a new playground for other bandit algorithms beyond ε_n-greedy.

Aknowledgments The authors would like to acknowledge the stimulating environment provided by SequeL research group, Inria and CRIStAL. This work was supported by French Ministry of Higher Education and Research, by CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, and by FUI Hermès. Experiments were carried out using Grid'5000 testbed, supported by Inria, CNRS, RENATER and several universities as well as other organizations.

References

1. Agarwal, D., Chen, B.C., Elango, P., Motgi, N., Park, S.T., Ramakrishnan, R., Roy, S., Zachariah, J.: Online models for content optimization. In: Proc. of NIPS'08. pp. 17–24 (2008)
2. Audibert, J.Y., Munos, R., Szepesvári, C.: Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. Theoretical Computer Science 410(19), 1876–1902 (2009)
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47, 235–256 (May 2002)
4. Bennett, J., Lanning, S., Netflix: The Netflix prize. In: KDD Cup and Workshop (2007)
5. Bhagat, S., Weinsberg, U., Ioannidis, S., Taft, N.: Recommending with an agenda: Active learning of private attributes using matrix factorization. In: Proc. of RecSys'14. pp. 65–72 (2014)
6. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: Proc. of NIPS. pp. 161–168 (2007)
7. Chapelle, O., Li, L.: An empirical evaluation of thompson sampling. In: Proc. of NIPS'11. pp. 2249–2257 (2011)
8. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-N recommendation tasks. In: Proc. of RecSys'10. pp. 39–46 (2010)

9. Garcin, F., Faltings, B., Donatsch, O., Alazzawi, A., Bruttin, C., Huber, A.: Offline and online evaluation of news recommender systems at swissinfo.ch. In: Proc. of RecSys'14. pp. 169–176. ACM (2014)
10. Garivier, A., Cappé, O.: The KL-UCB algorithm for bounded stochastic bandits and beyond. In: Proc. of COLT'11. pp. 359–376 (2011)
11. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5(4), 19 (2015)
12. Kawale, J., Bui, H., Kveton, B., Thanh, L.T., Chawla, S.: Efficient thompson sampling for online matrix-factorization recommendation. In: NIPS'15 (2015)
13. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (Aug 2009)
14. Koren, Y., Sill, J.: Ordrec: An ordinal model for predicting personalized item rating distributions. In: Proc. of RecSys'11. pp. 117–124 (2011)
15. Langford, J., Strehl, A., Wortman, J.: Exploration scavenging. In: Proc. of ICML. pp. 528–535 (2008)
16. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th international conference on World wide web. pp. 661–670. ACM (2010)
17. Li, L., Chu, W., Langford, J., Wang, X.: Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In: Proc. of WSDM'11. pp. 297–306 (2011)
18. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proc. of World Wide Web (WWW'10). pp. 661–670 (2010)
19. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the fourth ACM international conference on Web search and data mining. pp. 287–296. ACM (2011)
20. Mary, J., Gaudel, R., Preux, P.: Bandits and Recommender Systems. In: Proc. of Mach. Learn., Optimization and big Data (MOD'15) (2015)
21. Nakamura, A.: A ucb-like strategy of collaborative filtering. In: Proc. of ACML'14 (2014)
22. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proc. of UAI'09. pp. 452–461 (2009)
23. Said, A., Bellogín, A.: Comparative recommender system evaluation: benchmarking recommendation frameworks. In: Proc. of RecSys'14. pp. 129–136 (2014)
24. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., Hanjalic, A.: CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In: Proc. of RecSys'12. pp. 139–146 (2012)
25. Tang, L., Jiang, Y., Li, L., Li, T.: Ensemble contextual bandits for personalized recommendation. In: Proc. of RecSys'14 (2014)
26. Weston, J., Yee, H., Weiss, R.J.: Learning to rank recommendations with the k-order statistic loss. In: Proc. of RecSys'13. pp. 245–248 (2013)
27. Xing, Z., Wang, X., Wang, Y.: Enhancing Collaborative Filtering Music Recommendation by Balancing Exploration and Exploitation. In: Proc. of int. soc. for Music Inf. Retr. (ISMIR). pp. 445–450 (2014)
28. Zhao, X., Zhang, W., Wang, J.: Interactive collaborative filtering. In: CKIM'13. pp. 1411–1420 (2013)
29. Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R.: Large-scale parallel collaborative filtering for the netflix prize. In: Proc. of Alg. Aspects in Information and Management (AAIM'08). pp. 337–348 (2008)