Collaborative Filtering as a Multi-Armed Bandit

Frédéric Guillou
Inria Lille - Nord Europe
F-59650 Villeneuve d'Ascq, France
frederic.guillou@inria.fr

Romaric Gaudel & Philippe Preux
Univ. Lille, CNRS, Centrale Lille
UMR 9189 - CRIStAL
F-59000 Lille, France
romaric.gaudel@inria.fr
philippe.preux@inria.fr

Abstract

Recommender Systems (RS) aim at suggesting to users one or several items in which they might have interest. Following the feedback they receive from the user, these systems have to adapt their model in order to improve future recommendations. The repetition of these steps defines the RS as a sequential process. This sequential aspect raises an exploration-exploitation dilemma, which is surprisingly rarely taken into account for RS without contextual information. In this paper we present an explore-exploit collaborative filtering RS, based on Matrix Factorization and Bandits algorithms. Using experiments on artificial and real datasets, we show the importance and practicability of using sequential approaches to perform recommendation. We also study the impact of the model update on both the quality and the computation time of the recommendation procedure.

1 Introduction

We consider Collaborative Filtering approaches based on Matrix Completion; in particular, we want to emphasize that *we do not use any side information*, neither about users, nor items. 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. Such RS works 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 this feedback will be used to build the model at step 1 during next iterations.

In this paper, we investigate the consequences of this loop compared to a traditional batch setting.

This recommendation setting was popularized by the Netflix challenge [1]. Most of researches focus on step 1 of the recommendation, with a whole section of proposed approaches modeling the taste between users and items as a matrix $\mathbf{R}^*$ [2]. 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 RS, it does not evaluate the quality of the final recommendation. Namely, having a good matrix-completion algorithm does not imply having a good RS: for example, if the RS recommends

5 items at step 2, this RS only needs to know the identity of these items. It would be useless to know how well the ten worst items are predicted or ranked. Similar remarks open research to Matrix Completion algorithms targeting directly a good recommendation, such as rank-based matrix completion algorithms [3].

The paper brings three main contributions. First, it champions the necessity for a sequential evaluation process for RS. Second, it highlights the need of a trade-off between exploration and exploitation for an RS to be optimal. Last, it introduces a sequential RS based on Online Learning strategies which both (i) is feasible and (ii) makes a good trade-off between exploration and exploitation. These three contributions are supported by experiments on synthetic and real datasets.

Note that the exploration-exploitation dilemma is already studied in a sub-field of RS which has access to a representation of the context (the user, the item, the webpage, ...) [4]. Typical applications are the selection of news or ads to show on a web-page. In constrat with these studies, we focus on the setting where these representations are unknown and have to be inferred from users feedback. While contextual recommendation builds upon strong theoretical results on linear bandits, up to our knowledge, the field is empty as soon as there is no contextual information.We also hope that this paper will bring part of Reinforcement Learning community to focus on that promising field.

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 artificial data, and on real data. 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. 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 to the RS,
2. the RS selects an item j_t among the set of available items,
3. user i_t returns a feedback $r_t = r^*_{i_t,j_t} + \varepsilon_t$ for item j_t, where ε_t is a Gaussian noise.

We refer to applications where the feedback r_t corresponds to the quantity that has to be optimized, aka. *reward*. In such a context, the aim of the RS is to maximize the reward accumulated along time-steps $\mathrm{CumRew}_T = \sum_{t=1}^{T} r_t$, or to minimize the non-standard *cumulative regret* $\mathcal{R}_T = \sum_{t=1}^{T} \Delta_t$, where $\Delta_t = \mathrm{argmax}_j\, r^*_{i_t,j} - r_t$ is a kind of loss while playing sub-optimal arm j_t.

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 $I_t(j)$ (respectively $J_t(i)$) 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 vector corresponding to the i-th row of $\mathbf{M}$.

3 Building blocks

We introduce in Sec. 4 a RS which handles the sequential aspect of recommendation. 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 briefly go over state of the art approaches for both tasks.

3.1 Alternating Least Square for Matrix Factorization

Since the Netflix challenge [1], many works on RS focus on Matrix Factorization [2]: 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 $N \times k$, $\mathbf{V}$ is a matrix of size $M \times k$, k is the rank of $\mathbf{R}^*$, and $k \ll \max(N, M)$.

Thereafter, the estimator of $\mathbf{R}^*$ is defined as $\widehat{\mathbf{R}}^* = \widehat{\mathbf{U}}\widehat{\mathbf{V}}^T$ where $(\widehat{\mathbf{U}}, \widehat{\mathbf{V}})$ is the solution of

$$\underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} \sum_{\forall (i,j) \in \mathcal{S}} \left(r_{i,j} - \mathbf{U}_i \mathbf{V}_j^T\right)^2 + \lambda \cdot \Omega(\mathbf{U}, \mathbf{V}), \tag{1}$$

in which $\lambda \in \mathbb{R}^+$, and the usual regularization term is $\Omega(\mathbf{U}, \mathbf{V}) = ||\mathbf{U}||^2 + ||\mathbf{V}||^2$. Eq. (1) corresponds to a non-convex optimization problem. The minimization is usually performed either by stochastic gradient descent (SGD), or by alternating least squares (ALS). ALS-WR [5] regularizes users and items according to their respective importance in the matrix of ratings: $\Omega(\mathbf{U}, \mathbf{V}) = \sum_i \#\mathcal{J}(i)||\mathbf{U}_i||^2 + \sum_j \#\mathcal{I}(j)||\mathbf{V}_j||^2$. This regularization is known to have a good empirical behavior — that is limited overfitting, easy tuning of λ and k, low RMSE. This is also the default one in Mahout[1].

3.2 Multi-Armed Bandits

The RS we focus on works in a sequential context. As a consequence, while the recommendation made at time-step t aims at collecting a good reward now, it also affects the information that is collected, and therefore 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 with regard to expected long term reward. This section recalls standard strategies to handle this short term vs. long term dilemma.

We consider the well-studied Multi-Armed Bandits (MAB) setting [6, 7]: 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, $j^* = \operatorname{argmax}_j \mu_j$ 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* $\bar{\mathcal{R}}_T = \sum_{t=1}^{T}(\mu^* - r_t)$, where r_t denotes the reward collected at time-step t while pulling arm j_t. As the parameters are unknown, at each time-step (except the last one), 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 [6] proposes a strategy based on an upper confidence bound (UCB1) to handle this exploration vs. exploitation dilemma. UCB1 balances exploration and exploitation by playing the arm $j_t = \operatorname{argmax}_j \hat{\mu}_j(t) + \sqrt{\frac{2 \ln t}{T_j(t)}}$, 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 [8, 9, 10] 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 an even simpler but efficient approach to balance exploration and exploitation [6]. It consists in playing the greedy strategy ($j_t = \operatorname{argmax}_j \hat{\mu}_j(t)$) 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 which depends on M.

Thompson sampling is another well-known approach to solve the dilemma in a Bayesian way [11].

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 cumulative 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. At time-step t, for a given user i_t, ALS-

[1] https://mahout.apache.org/

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, m, 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 $\mathcal{J}_t$ of allowed items

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

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$(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}, \mathbf{R}, \mathcal{S}, \lambda, m, i_t, j_t)$ **end if**

end for

Obviously, ALS-WR requires too large computation time to be run at each time-step. A solution consists in running ALS-WR from times to times, say every T_u time-steps. While such 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 [12], we solve that problem by designing a mini-batch version of ALS-WR, denoted mBALS-WR. Alg. 2 describes that new algorithm.

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

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

for $i \in \{i_t\} \cup \{(m-1) \text{ randomly drawn users}\}$ **do**

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

end for

for $j \in \{j_t\} \cup \{(m-1) \text{ randomly drawn items}\}$ **do**

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

end for

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 number m of lines of $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$. When the parameter m tends towards $\max(N, M)$, mBALS-WR is similar to a one-pass ALS-WR. To the opposite, when $m = 1$, mBALS-WR only updates the row of $\widehat{\mathbf{U}}$ (respectively $\widehat{\mathbf{V}}$) corresponding to the user which asked for a recommendation (resp. the item which was recommended) at current time-step t.

mBALS-WR spreads the computing budget along time-steps: ALS-WR consumes a huge computing budget every thousands of time-steps, and selects the items to recommend based on an outdated decomposition at remaining time-steps. To the opposite, 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 evaluate empirically the algorithms on artificial data, and on real datasets in the sequential setting. Two series of experiments emphasize two aspects of the sequential RS: Sec. 5.2 show that exploration helps to improve the quality of the RS model and Sec. 5.3 focuses on the influence of the method updating the matrix model.

5.1 Experimental setting and remarks

For each dataset, we start with an empty matrix $\mathbf{R}$. Then, for $T = 200,000$ consecutive time-steps

1. we select a user i_t uniformly at random among those who have not yet rated all the items,
2. the algorithm chooses an item j_t to recommend among the unrated ones,
3. we reveal the value of r_{i_t,j_t} and increment the cumulative regret score.

Some difficulties arise when using real datasets: first, in most cases, the ground truth is unknown, and only a very small fraction of ratings is known since users give ratings only to items they purchase/listen/watch. This makes the evaluation of algorithms uneasy considering we need in advance the reward of items we include in the list of possible recommendations. Second, a bias is often observed on the known ratings, due to the tendency of users to mostly rate items they like.

To overcome these difficulties, we consider four datasets for our experiments: an artificial one, and sub-datasets from three real datasets. Datasets characteritics are reported in Table 1.

Table 1: Dataset characteristics.

	Artificial	Jester	Movielens	LibimSeTi
Number of users	1,000	7,200	6,000	5,000
Number of items	1,000	100	4,000	5,000
Number of ratings	1,000,000	720,000	4,915,949	1,985,992

The first dataset is generated artificially based on a ground truth matrix $\mathbf{R}^*$. This matrix is generated as in [13]. 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 $r^*_{i,j}$: $r_{i,j} = r^*_{i,j} + \mathcal{N}(0, 0.5)$. The second dataset has been created from Jester data by extracting only the users who have rated all items. While this dataset includes few items, it is a real-life dataset and any entry of the corresponding matrix is known. The third dataset is a subset of Movielens20M, created by first selecting the top 6,000 users who have most rated movies, and then by selecting the 4,000 most rated movies. The matrix is of similar size with the well-studied Movielens1M, but it contains almost five times more ratings. The last dataset is a subset of LibimSeTi dataset, created in a similar fashion by first selecting the top 5,000 users who gave the largest number of ratings, and then by selecting the 5,000 people who have been most rated. This is an interesting case as "items" here are actually other users of the website.

In the last two cases, we do not have access to the full matrix $\mathbf{R}$, which makes the evaluation of algorithms more complex. This issue is solved in the case of contextual bandits by using reject sampling [14]: the algorithm chooses an arm (item), and if the arm does not appear in logged data, the choice is discarded, as if the algorithm has 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 no more need 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.

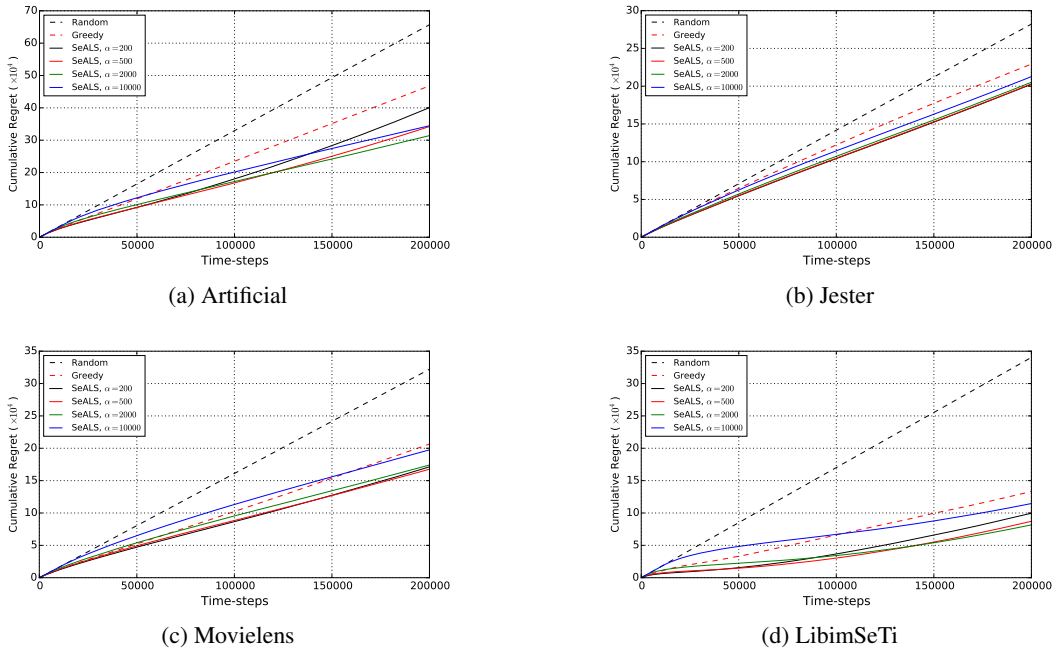
However, a minimum amount of ratings per user is needed to compare the algorithms. This is the reason why we restrict experiments to subsets of real datasets with a sufficient number of ratings per user. We insist on the fact that this is necessary to perform a meaningful evaluation of the algorithms; obviously, this is not required to use the algorithms as a real-life RS.

Note that we compare the algorithms based on the cumulative regret for the sake of easy comparaison to MAB results. However, this metric is prone to discussion. Indeed, for each user, the items are removed from the list of future recommendations as soon as they are rated. As such, the optimal arm fluctuates, which affects the cumulative regret.

5.2 Impact of exploration

These experiments compare two strategies to recommend an item: (i) SeALS with $\alpha > 0$, and (ii) SeALS with $\alpha = 0$ (denoted Greedy) which corresponds to the greedy strategy. Both strategies use the maximum possible value for m and update the estimator of $\mathbf{R}^*$ every $T_u = 200$ time-steps. Fig. 1 displays the cumulative regret $\mathcal{R}_T$ obtained by the RS during the $200,000$ consecutive recommendations. In order to bound the behavior of the algorithms, we also display the results of an algorithm selecting the item randomly (denoted Random). Results on all datasets demonstrate

the need of exploration during the recommendation process: SeALS gets a lower $\mathcal{R}_T$ than Greedy. We also show experiments with different values on α to highlight the influence of exploration on the cumulative reward. While extreme values of α increase $\mathcal{R}_T$, the results are the best (and equivalent) for a large range of values for α ($500 \leq \alpha \leq 2,000$).



(a) Artificial

(b) Jester

(c) Movielens

(d) LibimSeTi

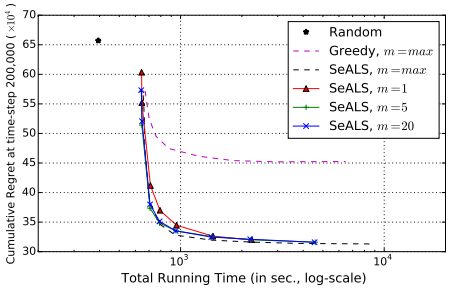
Figure 1: Cumulative regret as a function of the time-step. Figures are averaged over 40 runs. For the artificial dataset, $\lambda = 0.01$ for Greedy and $\lambda = 0.05$ for SeALS. For other datasets, $\lambda = 0.1$ for Greedy and $\lambda = 0.15$ for SeALS. Parameter k is set to 15 for ALS-WR.

5.3 Impact of the update strategy

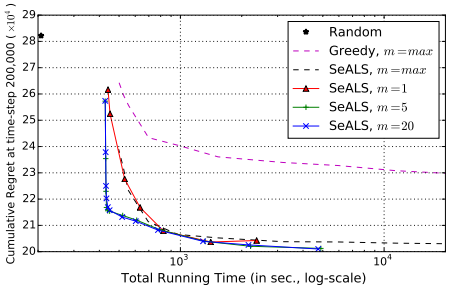
To evaluate the impact of the method used to update the model as well as the period at which updates take place, we set up a different evaluation display. For each RS, we run experiments as presented in Sec. 5.1 and we store the final running time as well as $\mathcal{R}_T$ at the end of the $200,000$ iterations. Such evaluation methodology allows to find which size of mini-batch (respectively update period) leads to the best trade-off between the final score and the running time. Fig. 2 displays the results for all datasets. Each curve corresponds to a fixed size of mini-batch m, and every point of a same curve represents a specific value of the update period T_u. A point with a large running time results from a small value T_u. The value of T_u varies in $[200; 60,000]$ for SeALS with $m = max$ and in $[1; 200]$ for SeALS with $m = 1$. On the artificial dataset, we notice that $\mathcal{R}_T$ decreases very rapidly and reaches a plateau. In this case, an efficient strategy to obtain the best results is to spend a bit more computation time in order to reach that plateau. There is no need for clever update strategy based on mini-batch optimization. On the real-life datasets, SeALS with the full update again quickly reaches its upper limit. However, the mini-batch approach still finds room for improvement and reaches the best cumulative regret within a smaller running time. Note that best results are obtained for a total running time of a few thousands seconds which corresponds to a dozen milli-seconds per recommendation. These experiments do not assert there is a universal size of mini-batch to achieve the best trade-off. They yet illustrate the importance this parameter can play in making good recommendation using the least computing power as possible for the model update.

6 Related work

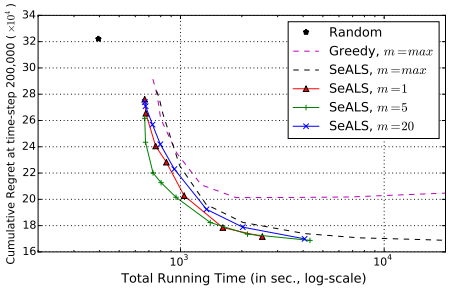
To the best of our knowledge, very few papers consider a similar RS setting [16, 17, 18, 19, 20, 21]. In [21], we extend linear bandits to the Matrix Factorization context. Other papers propose a solution



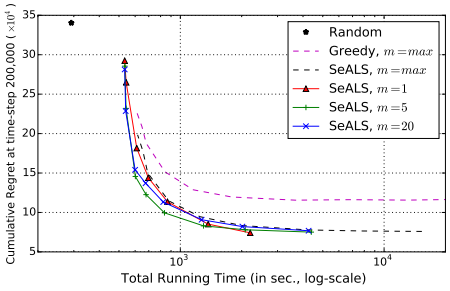
(a) Artificial, λ=0.05 and $\alpha = 2,000$



(b) Jester, λ=0.15 and $\alpha = 500$



(c) Movielens, λ=0.15 and $\alpha = 500$



(d) LibimSeTi, λ=0.15 and $\alpha = 2,000$

Figure 2: Cumulative regret at time-step 200,000 as a function of running time. Figures are averaged over 40 runs. Parameter k of ALS-WR is set to 15.

based on a Bayesian model. SeALS is the first approach which scales up while handling matrices of rank > 2. On the contrary, the exploration-exploitation trade-off is already well-studied in the field of RS which has access to a representation of the context[4]. Compared to them, we focus on the setting where these representations are unknown and have to be inferred from users feedback. The cold-start setting also focuses on the need for exploration [22, 23]. The aim is to deal with new users (or new items). While some approaches look at external information on the user [22], some papers rewrite the problem as an Active Learning problem [23]: *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 [7]. Finally, while [24] and [25] focus on online Matrix Factorization, they omit the exploration-exploitation dilemma, and they focus on stochastic gradient descent to solve (1).

7 Conclusion and future work

In this paper we handle Recommender Systems based on Matrix Completion in the suitable context: a never-ending alternation between (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 a convenient time (a dozen milli-seconds 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 than 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] J. Bennett, S. Lanning, and Netflix, "The Netflix prize," in *KDD Cup and Workshop*, 2007.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30–37, Aug. 2009.

[3] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-N recommendation tasks," in *Proc. of RecSys'10*, pp. 39–46, ACM Press, 2010.

[4] L. Tang, Y. Jiang, L. Li, and T. Li, "Ensemble contextual bandits for personalized recommendation," in *Proc. of RecSys'14*, 2014.

[5] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Proc. of Alg. Aspects in Information and Management (AAIM'08)*, pp. 337–348, 2008.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, pp. 235–256, May 2002.

[7] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *CoRR*, vol. abs/1204.5721, 2012.

[8] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Exploration-exploitation tradeoff using variance estimates in multi-armed bandits," *Theor. Comput. Sci.*, vol. 410, pp. 1876–1902, Apr. 2009.

[9] A. Garivier and O. Cappé, "The KL-UCB algorithm for bounded stochastic bandits and beyond," in *Proc. of COLT'11*, pp. 359–376, 2011.

[10] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. of World Wide Web (WWW'10)*, pp. 661–670, 2010.

[11] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Proc. of the 25^{th} Annual Conf. on Neural Information Processing Systems (NIPS'11)*, pp. 2249–2257, 2011.

[12] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. of NIPS*, pp. 161–168, 2007.

[13] S. Chatterjee, "Matrix estimation by universal singular value thresholding," *pre-print*, 2012. `http://arxiv.org/abs/1212.1247`.

[14] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms.," in *Proc. of WSDM'11*, pp. 297–306, 2011.

[15] J. Langford, A. Strehl, and J. Wortman, "Exploration scavenging," in *Proc. of ICML*, pp. 528–535, 2008.

[16] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A Contextual-Bandit Approach to Personalized News Article Recommendation," *CoRR*, vol. abs/1003.0146, 2010.

[17] X. Zhao, W. Zhang, and J. Wang, "Interactive collaborative filtering," in *CKIM'13*, pp. 1411–1420, 2013.

[18] Z. Xing, X. Wang, and Y. Wang, "Enhancing Collaborative Filtering Music Recommendation by Balancing Exploration and Exploitation," in *Proc. of int. soc. for Music Inf. Retr. (ISMIR)*, pp. 445–450, 2014.

[19] A. Nakamura, "A ucb-like strategy of collaborative filtering," in *Proc. of ACML'14*, 2014.

[20] J. Kawale, H. Bui, B. Kveton, L. T. Thanh, and S. Chawla, "Efficient thompson sampling for online matrix-factorization recommendation," in *Proc. of NIPS'15*, 2015.

[21] J. Mary, R. Gaudel, and P. Preux, "Bandits and Recommender Systems," in *Proc. of Mach. Learn., Optimization and big Data (MOD'15)*, 2015.

[22] D. Agarwal, B.-C. Chen, P. Elango, N. Motgi, S.-T. Park, R. Ramakrishnan, S. Roy, and J. Zachariah, "Online models for content optimization," in *Proc. of NIPS'08*, pp. 17–24, 2008.

[23] S. Bhagat, U. Weinsberg, S. Ioannidis, and N. Taft, "Recommending with an agenda: Active learning of private attributes using matrix factorization," in *Proc. of RecSys'14*, pp. 65–72, 2014.

[24] S. Rendle and L. Schmidt-Thieme, "Online-updating Regularized Kernel Matrix Factorization Models for Large-scale Recommender Systems," in *Proc. of RecSys'08*, pp. 251–258, 2008.

[25] X. Luo, Y. Xia, and Q. Zhu, "Incremental Collaborative Filtering recommender based on Regularized Matrix Factorization," *Knowledge-Based Systems*, vol. 27, pp. 271 – 280, 2012.