Apprentissage séquentiel de préférence utilisateurs pour les systèmes de recommandation

Aleksandra Burashnikova, Yury Maximov, Marianne Clausel, Charlotte Laclau, Franck Iutzeler, Massih-Reza Amini

Apprentissage séquentiel de préférence utilisateurs pour les systèmes de recommandation

Aleksandra Burashnikova[1,5], Yury Maximov[1,2], Marianne Clausel[3], Charlotte Laclau[4], Franck Iutzeler[5], and Massih-Reza Amini[5]

[1]Skolkovo Institute of Science and Technology
[2]Los Alamos National Laboratory
[3]University of Lorraine
[4]Telecom Saint-Etienne
[5]University Grenoble Alpes

Résumé

Dans cet article, nous présentons une stratégie séquentielle pour l'apprentissage de systèmes de recommandation à grande échelle sur la base d'une rétroaction implicite, principalement sous la forme de clics. L'approche proposée consiste à minimiser l'erreur d'ordonnacement sur les blocs de produits consécutifs constitués d'une séquence de produits non cliqués suivie d'un produit cliqué pour chaque utilisateur. Afin d'éviter de mettre à jour les paramètres du modèle sur un nombre anormalement élevé de clics (principalement dus aux bots), nous introduisons un seuil supérieur et un seuil inférieur sur le nombre de mises à jour des paramètres pour chaque utilisateur. Ces seuils sont estimés sur la distribution du nombre de blocs dans l'ensemble d'apprentissage. Nous proposons une analyse de convergence de l'algorithme et démontrons empiriquement son efficacité sur six collections, à la fois en ce qui concerne les différentes mesures de performance et le temps de calcul.

1 Introduction

With the increasing number of products available online, there is a surge of interest in the design of automatic systems that provide personalized recommendations to users by adapting to their taste. The study of RS has become an active area of research these past years, especially since the Netflix Price [BL07]. One characteristic of online recommendation is the huge unbalance between the available number of products and those shown to the users. On the other hand, bots that interact with the system by providing too much feedback over some targeted items [KG16]. Contrariwise, many users do not interact with the system over the items that are shown to them. In this context, the main challenges concern the design of a scalable and an efficient online RS in the presence of noise and unbalanced data. These challenges have evolved in time with the continuous development of data collections released for competitions or issued from e-commerce. Recent approaches for RS [WZZ+20] now primarily consider feedback, mostly in the form of clicks known as *implicit* feedback, which is challenging to deal with as they do not depict the preference of a user over items, i.e., (no)click does not necessarily mean (dis)like. In this case, most of the developed approaches are based on the Learning-to-rank paradigm and focus on how to leverage the click information over the unclick one without considering the sequence of users' interactions. In this paper, we propose $\texttt{SAROS}_b$, a sequential strategy for recommender systems with implicit feedback that updates model parameters user per user over blocks of items constituted by a sequence of unclicked items followed by a clicked one. Model parameters are updated by minimizing the ranking loss over the blocks of unclicked items followed by a clicked one using a gradient descent approach. Updates are discarded for users who interact very little or a lot with the system. Furthermore, we provide empirical evaluation over six large publicly available datasets showing that the proposed approach is highly competitive compared to the state-of-the-art models in terms of quality metrics and, that are significantly faster than both the batch and the online versions of the algorithm.

2 Related work

Two main approaches have been proposed for recommender systems. The first one, Content-Based recommendation or cognitive filtering [PB07], makes use of existing contextual information about the users (e.g., demographic information) or items (e.g., textual description) for the recommendation. The second approach, Collaborative Filtering, is undoubtedly the most popular one [SK09], relies on past interactions and recommends items to users based on the feedback provided by other similar users.

Traditionally, collaborative filtering systems have been designed using *explicit* feedback, mostly in the form of rating [Kor08]. However, rating information is non-existent on most of e-commerce websites and is challenging to collect, and user interactions are often done sequentially. Recent RS systems focus on learning scoring functions using *implicit* feedback to assign higher scores to clicked items than to unclicked ones rather than to predict the clicks as it is usually the case when we deal with explicit feedback [STH+21]. The idea here is that even a clicked item does not necessarily express the preference of a user for that item, it has much more value than a set of unclicked items for which no action has been made.

Many new approaches tackle the sequential learning problem for RS by taking into account the temporal aspect of interactions directly in the design of a dedicated model and are mainly based on Markov Models (MM), Reinforcement Learning (RL), and Recurrent Neural Networks (RNN) [DLZ17]. Recommender systems based on Markov Models, consider a subsequent interaction of users as a stochastic process over discrete random variables related to predefined user behavior. These approaches suffer from some limitations, mainly due to the sparsity of the data leading to a poor estimation of the transition matrix and choice of an appropriate order for the model [HM16]. Various strategies have been proposed to leverage the limitations of Markov Models. For instance, [HM16] suggests combining similarity-based methods with high-order Markov Chains. Although it has been shown that in some cases, the proposed approaches can capture the temporal aspect of user interactions, these models suffer from a high time-complexity and do not pass the scale. Some other methods consider RS as a Markov decision process (MDP) problem and solve it using reinforcement learning (RL) [TB14]. The size of discrete actions bringing the RL solver to a larger class of problems is also a bottleneck for these approaches. Recently many Recurrent Neural Networks (RNN) such as GRU or LSTM have been proposed for personalized recommendations [KM18]. In this approach, the input of the network is generally the sequence of user interactions consisted of a single behaviour type (click, adding to favourites, purchase, etc.) and the output is the predicted preference over items in the form of posterior probabilities of the considered behaviour type given the items. A comprehensive survey of Neural Networks based sequential approaches for personalized recommendation is presented in [FZSG20]. All these approaches do not consider negative interactions ; i.e. viewed items that are not clicked or purchased ; and the system's performance on new test data may be affected.

Our approach differs from other sequential based methods in the way that the model parameters are updated, at each time a block of unclicked items followed by a clicked one is constituted. This update scheme follows the hypothesis that user preference is not absolute over the items which were clicked, but it is relative with respect to items that were viewed. We further provide a proof of convergence of the proposed approach in the general case of non-convex loss functions in the case where the number of blocks per user interaction is controlled.

3 Framework

Throughout, we use the following notation. For any positive integer n, $[n]$ denotes the set $[n] = \{1, \ldots, n\}$. We suppose that $\mathcal{I} = [M]$ and $\mathcal{U} = [N]$ are two sets of indexes defined over respectively the items and the users. Further, we assume that each pair constituted by a user u and an item i is identically and independently distributed (i.i.d) according to a fixed yet unknown distribution $\mathcal{D}$. At the end of his or her session, a user $u \in \mathcal{U}$ has reviewed a subset of items $\mathcal{I}_u \subseteq \mathcal{I}$ that can be decomposed into two sets : the set of preferred and non-preferred items denoted by $\mathcal{I}_u^+$ and $\mathcal{I}_u^-$, respectively. Hence, for each pair of items $(i, i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-$, the user u prefers item i over item i' ; symbolized by the relation $i \underset{u}{\succ} i'$. From this preference relation a desired output $y_{u,i,i'} \in \{-1, +1\}$ is defined over the pairs $(u, i) \in \mathcal{U} \times \mathcal{I}$ and $(u, i') \in \mathcal{U} \times \mathcal{I}$, such that $y_{u,i,i'} = +1$ if and only if $i \underset{u}{\succ} i'$. We suppose that the indexes of users as well as those of items in the set $\mathcal{I}_u$, shown to the active user $u \in \mathcal{U}$, are ordered by time.

Finally, for each user u, parameter updates are performed over blocks of consecutive items where a block $\mathcal{B}_u^t = \mathrm{N}_u^t \sqcup \Pi_u^t$, corresponds to a time-ordered sequence (w.r.t. the time when the interaction is done) of no-preferred items, N_u^t, and at least one preferred one, Π_u^t. Hence, $\mathcal{I}_u^+ = \bigcup_t \Pi_u^t$ and $\mathcal{I}_u^- = \bigcup_t \mathrm{N}_u^t ; \forall u \in \mathcal{U}$.

3.1 Learning Objective

Our objective here is to minimize an expected error penalizing the misordering of all pairs of interacted items i and i' for a user u. Commonly, this objective is given under the Empirical Risk Minimization (ERM) principle, by minimizing the empirical ranking loss estimated over the items and the final set of users who interacted with the system :

$$\hat{\mathcal{L}}_u(\omega)=\frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|}\sum_{i\in\mathcal{I}_u^+}\sum_{i'\in\mathcal{I}_u^-}\ell_{u,i,i'}(\omega), \qquad (1)$$

where, $\ell_{u,i,i'}(.)$ is an instantaneous ranking loss defined over the triplet (u,i,i') with $i \underset{u}{\succ} i'$. Hence, $\hat{\mathcal{L}}_u(\omega)$ is the pairwise ranking loss with respect to user's interactions and $\mathcal{L}(\omega) = \mathbb{E}_u\left[\hat{\mathcal{L}}_u(\omega)\right]$ is the expected ranking loss, where $\mathbb{E}_u$ is the expectation with respect to users chosen randomly according to the marginal distribution.

As in other studies, we represent each user u and each item i respectively by vectors $\bar{U}_u \in \mathbb{R}^k$ and $\bar{I}_i \in \mathbb{R}^k$ in the same latent space of dimension k [KBV09]. The set of weights to be found $\omega = (\bar{U}, \bar{I})$, are then matrices formed by the vector representations of users $\bar{U} = (\bar{U}_u)_{u\in[N]} \in \mathbb{R}^{N\times k}$ and items $\bar{I} = (\bar{I}_i)_{i\in[M]} \in \mathbb{R}^{M\times k}$. A common approach is to minimize the above ranking loss in batch mode with the goal of finding user and item embeddings, so that the dot product between these representations in the latent space better reflects the preference of users over items. Other strategies have been proposed to minimize this empirical loss (1), among which the most popular one is perhaps the Bayesian Personalized Ranking (`BPR`) model [RFGST09]. In this approach, the instantaneous loss, $\ell_{u,i,i'}$, is the surrogate regularized logistic loss for some hyperparameter $\mu \geq 0$:

$$\begin{aligned}\ell_{u,i,i'}(\omega) = \log\left(1+e^{-y_{u,i,i'}\bar{U}_u^\top(\bar{I}_i-\bar{I}_{i'})}\right)\\ +\mu(\|\bar{U}_u\|_2^2+\|\bar{I}_i\|_2^2+\|\bar{I}_{i'}\|_2^2)\end{aligned} \qquad (2)$$

The `BPR` algorithm proceeds by first randomly choosing a user u, and then repeatedly selecting two pairs $(i,i') \in \mathcal{I}_u \times \mathcal{I}_u$. In the case where one of the chosen items is preferred over the other one (i.e., $y_{u,i,i'} \in \{-1,+1\}$), the algorithm then updates the weights using the stochastic gradient descent method for minimizing (1).

3.2 Algorithm `SAROS`

A key point in recommendation is that user preferences for items are largely determined by the context in which they are presented to the user. A user may prefer (or not) two items independently of one another, but he or she may have a totally different preference for these items within a given set of shown items. This effect of local preference is not taken into account by randomly sampling triplets formed by a user and corresponding clicked and unclicked items over the entire set of shown items to the user. Furthermore, triplets corresponding to different users are non uniformly distributed, as interactions vary from one user to another one, and for parameter updates ; triplets corresponding to low interactions have a small chance to be chosen. In order to tackle these points ; we propose to update the parameters sequentially over the blocks of non-preferred items followed by preferred ones for each user u. The constitution of sequences of non-preferred and preferred blocks of items respectively noted as N_u^t and $\Pi_u t$ for $t \in \{1,\dots,B_u\}$, and, two users is shown in Figure 1.

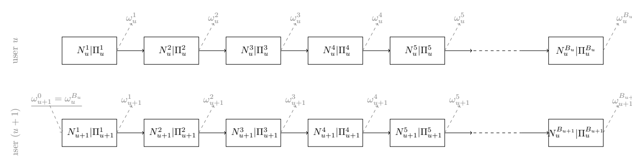

FIGURE 1 – The horizontal axis represents the sequence of interactions over items ordered by time. Each update of weights $\omega_u^t; t \in \{1,\dots,B_u\}$ occurs whenever the corresponding sets of negative interactions, N_u^t, and positive ones, Π_u^t, exist.

In this case, at each time t a block $\mathcal{B}_u^t = \mathrm{N}_u^t \sqcup \Pi_u^t$ is formed for user u ; weights are updated by miniminzing the ranking loss corresponding to this block :

$$\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t) = \frac{1}{|\Pi_u^t||\mathrm{N}_u^t|}\sum_{i\in\Pi_u^t}\sum_{i'\in\mathrm{N}_u^t}\ell_{u,i,i'}(\omega_u^t). \qquad (3)$$

Note that this is different from session-based recommendations [WCW+19] in which each session is also made up of a series of user-item interactions that take place over a period of time. However, session-based recommendations approaches capture both user's short-term preference from recent sessions and the preference dynamics representing the change of preferences from one session to the next by using each session as the basic input unit, which is not the case in our study.

Starting from initial weights ω_1^0 chosen randomly for the first user. The sequential update rule, for each current user u consists in updating the weights by making one step towards the opposite direction of the gradient of the ranking loss estimated on the current block, $\mathcal{B}_u^t = \mathrm{N}_u^t \sqcup \Pi_u^t$:

$$\omega_u^{t+1} = \omega_u^t - \frac{\eta}{|\mathrm{N}_u^t||\Pi_u^t|}\sum_{i\in\Pi_u^t}\sum_{i'\in\mathrm{N}_u^t}\nabla\ell_{u,i,i'}(\omega_u^t) \tag{4}$$

For a given user u, parameter updates are discarded if the number of blocks $(\mathcal{B}_u^t)_t$ for the current user falls outside the interval $[b, B]$. In this case, parameters are initialized with respect to the latest update before user u and they are updated with respect to a new user's interactions.

3.3 Convergence analysis

The proofs of convergence is given under a common hypothesis that the sample distribution is not instantaneously affected by learning of the weights, i.e. the samples can be considered as i.i.d. More precisely, we assume the following hypothesis.

Assumption 1. *For an i.i.d. sequence of user and any $u, t \geq 1$, we have*

1. $\mathbb{E}_{(u,\mathcal{B}_u^t)}\|\nabla\mathcal{L}(\omega_u^t) - \nabla\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t)\|_2^2 \leq \sigma^2$,
2. *For any u,*

$$\left|\mathbb{E}_{\mathcal{B}_u^t|u}\langle\nabla\mathcal{L}(\omega_u^t), \nabla\mathcal{L}(\omega_u^t) - \nabla\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t)\rangle\right| \leq a^2\|\nabla\mathcal{L}(\omega_u^t)\|_2^2$$

for some parameters $\sigma > 0$ and $a \in [0, 1/2)$ independent of u and t.

The first assumption is common in stochastic optimization and it implies consistency of the sample average approximation of the gradient. However, this assumption is not sufficient to prove the convergence because of interdependency of different blocks of items for the same user. The second assumption implies that in the neighborhood of the optimal point, we have $\nabla\mathcal{L}(\omega_u^t)^\top\nabla\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t) \approx \|\nabla\mathcal{L}(\omega_u^t)\|_2^2$, which greatly helps to establish consistency and convergence rates. In particular, if an empirical estimate of the loss over a block is unbiased, e.g. $\mathbb{E}_{\mathcal{B}_u^t}\nabla\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega) = \nabla\mathcal{L}(\omega)$, the second assumption is satisfied with $a = 0$.

Theorem 1. *Let ℓ be a (possibly non-convex) β-smooth loss function. Assume, moreover, that the number of interactions per user belongs to an interval $[b, B]$ almost surely and assumption 1 is satisfied with some constants σ^2 and a, $0 < a < 1/2$. Then, for a step-size policy $\eta_u^t \equiv \eta_u$ with $\eta_u \leq 1/(B\beta)$ for any user u, one has*

$$\min_{u:\,1\leq u\leq N}\mathbb{E}\|\nabla\mathcal{L}(\omega_u^0)\|_2^2 \leq \frac{2(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_u^0)) + \beta\sigma^2\sum_{u=1}^N\sum_{t=1}^{|\mathcal{B}_u|}(\eta_u^t)^2}{\sum_{u=1}^N\sum_{t=1}^{|\mathcal{B}_u|}\eta_u^t(1 - a^2 - \beta\eta_u^t(1/2 - a^2))}.$$

In particular, for a constant step-size policy $\eta_u^t = \eta = c/\sqrt{N}$ satisfies $\eta\beta \leq 1$, one has

$$\min_{t,u}\|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \leq \frac{2}{b(1-4a^2)}\frac{2(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*))/c + \beta c\sigma^2 B}{\sqrt{N}}.$$

Démonstration. Since ℓ is a β smooth function, we have for any u and t :

$$\begin{aligned}
\mathcal{L}(\omega_u^{t+1}) &\leq \mathcal{L}(\omega_u^t) + \langle\nabla\mathcal{L}(\omega_u^t), \omega_u^{t+1} - \omega_u^t\rangle \\
&\quad + \frac{\beta}{2}(\eta_u^t)^2\|\nabla\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t)\|_2^2 \\
&= \mathcal{L}(\omega_u^t) - \eta_u^t\langle\nabla\mathcal{L}(\omega_u^t), \nabla\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t)\rangle \\
&\quad + \frac{\beta}{2}(\eta_u^t)^2\|\nabla\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t)\|_2^2
\end{aligned}$$

Following [Lan20]; by denoting $\delta_u^t = \nabla\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t) - \nabla\mathcal{L}(\omega_u^t)$, we have :

$$\begin{aligned}
\mathcal{L}(\omega_u^{t+1}) &\leq \mathcal{L}(\omega_u^t) + \frac{\beta(\eta_u^t)^2}{2}\|\delta_u^t\|_2^2 \\
&\quad -\left(\eta_u^i - \frac{\beta(\eta_u^t)^2}{2}\right)\|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \\
&\quad - \left(\eta_u^t - \beta(\eta_u^t)^2\right)\langle\nabla\mathcal{L}(\omega_u^t), \delta_u^t\rangle
\end{aligned} \tag{5}$$

Our next step is to take the expectation on both sides of inequality (5). According to Assumption 1, one has for some $a \in [0, 1/2)$:

$$\left(\eta_u^t - \beta(\eta_u^t)^2\right)\left|\mathbb{E}\langle\nabla\mathcal{L}(\omega_u^t), \delta_u^t\rangle\right| \leq \left(\eta_u^t - \beta(\eta_u^t)^2\right)a^2\|\nabla\mathcal{L}(\omega_u^t)\|_2^2$$

where the expectation is taken over the set of blocks and users seen so far.

Finally, taking the same expectation on both sides of inequality (5), it comes :

$$
\begin{aligned}
\mathcal{L}(\omega_u^{t+1}) &\le \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2\mathbb{E}\|\delta_u^t\|_2^2 \\
&\quad - \eta_u^t(1-\beta\eta_u^t/2 - a^2|1-\beta\eta_u^t|)\|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \\
&\le \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2\|\delta_u^t\|_2^2 \\
&\quad - \eta_u^t \underbrace{(1-a^2-\beta\eta_u^t(1/2-a^2))}_{:=z_u^t}\|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \\
&= \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2\|\delta_u^t\|_2^2 - \eta_u^t z_u^t\|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \\
&= \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2\sigma^2 - \eta_u^t z_u^t\|\nabla\mathcal{L}(\omega_u^t)\|_2^2, \quad (6)
\end{aligned}
$$

where the second inequality is due to $|\eta_u^t\beta| \le 1$. Also, as $|\eta_u^t\beta| \le 1$ and $a^2 \in [0, 1/2)$ one has $z_u^t > 0$ for any u, t. Rearranging the terms, one has

$$
\begin{aligned}
&\min_{t,u}\|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \\
&\le \frac{\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*) + \frac{\beta}{2}\sum_{u=1}^N\sum_{t=1}^{|\mathcal{B}_u|}(\eta_u^t)^2\sigma^2}{\sum_{u=1}^N\sum_{t=1}^{|\mathcal{B}_u|}\eta_u^t z_u^t} \\
&\le \frac{\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*) + \frac{\beta}{2}\sum_{u=1}^N\sum_{t=1}^{|\mathcal{B}_u|}(\eta_u^t)^2\sigma^2}{\sum_{u=1}^N\sum_{t=1}^{|\mathcal{B}_u|}\eta_u^t(1-a^2-\beta\eta_u^t(1/2-a^2))}
\end{aligned}
$$

Where, ω_* is the optimal point. Then, using a constant step-size policy, $\eta_u^i = \eta$, and the bounds on a block size, $b \le |\mathcal{B}_u| \le B$, we get :

$$
\begin{aligned}
&\min_{t,u}\|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \\
&\le \frac{\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*) + \frac{\beta\sigma^2}{2}N\sum_{u=1}^N\eta_u^2}{b\sum_{u=1}^N\eta_u(1-a^2-\beta\eta_u(1/2-a^2))} \\
&\le \frac{4\mathcal{L}(\omega_1^0) - 4\mathcal{L}(\omega_*) + 2\beta\sigma^2 B\sum_{u=1}^N\eta^2}{b(1-4a^2)\sum_{u=1}^N\eta} \\
&\le \frac{2}{b(1-4a^2)}\left\{\frac{2\mathcal{L}(\omega_1^0) - 2\mathcal{L}(\omega_*)}{N\eta} + \beta\sigma^2 B\eta\right\}.
\end{aligned}
$$

Taking $\eta = c/\sqrt{N}$ so that $0 < \eta \le 1/\beta$, one has

$$
\begin{aligned}
&\min_{t,u}\|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \\
&\le \frac{2}{b(1-4a^2)}\frac{2(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*))/c + \beta c\sigma^2 B}{\sqrt{N}}.
\end{aligned}
$$

If $b = B = 1$, this rate matches up to a constant factor to the standard $O(1/\sqrt{N})$ rate of the stochastic gradient descent. □

4 Experiments

In this section, we provide an empirical evaluation of our optimization strategy on some popular benchmarks proposed for evaluating RS. All subsequently discussed components were implemented in Python3 using the TensorFlow library [1]. We first proceed with a presentation of the general experimental set-up, including a description of the datasets and the baseline models.

Datasets. We report results obtained on five publicly available datasets, for the task of personalized Top-N recommendation on the following collections. ML-1M [HK15] and NETFLIX consist of user-movie ratings, on a scale of one to five, collected from a movie recommendation service and the Netflix company. The latter was released to support the Netflix Prize competition [BL07]. For both datasets, we consider ratings greater or equal to 4 as positive feedback, and others as negative feedback. We extracted a subset out of the OUTBRAIN dataset from of the Kaggle challenge that consisted in the recommendation of news content to users based on the 1,597,426 implicit feedback collected from multiple publisher sites in the United States. PANDOR is another publicly available dataset for online recommendation [SLA18] provided by Purch. The dataset records 2,073,379 clicks generated by 177,366 users of one of the Purch's high-tech website over 9,077 ads they have been shown during one month. RECSYS'16 is a sample based on historic XING data provided 6,330,562 feedback given by 39,518 users on the job posting items and the items generated by XING's job recommender system. KASANDR dataset [SLA+17] contains 15,844,717 interactions of 2,158,859 users in Germany using Kelkoo's online advertising platform. Table 1 presents some detailed statistics about each collection. Among these, we report the average number of clicks (positive feedback) and the average number of items that were viewed but not clicked (negative feedback). As we see from the table, OUTBRAIN, KASANDR, and PANDOR datasets are the most unbalanced ones in regards to the number of preferred and non-preferred items. To construct the training and the test sets, we discarded users who did not interact over the shown items and sorted all interactions according to time-based on the existing time-stamps related to each dataset. Furthermore, we considered 80% of each user's first interactions (both positive and negative) for training, and the remaining for the test. Finally, we have used 10% of the most recent interactions of users in the training set as validation set for hyperparameter tuning.

Compared approaches. To validate the sequential learning approach described in the previous sections,

1. `https://www.tensorflow.org/`.

Data	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	Sparsity	Avg. # of +	Avg. # of −
ML-1M	6,040	3,706	.9553	95.2767	70.4690
OUTBRAIN	49,615	105,176	.9997	6.1587	26.0377
PANDOR	177,366	9,077	.9987	1.3266	10.3632
NETFLIX	90,137	3,560	.9914	26.1872	20.2765
RECSYS'16	39,518	28,068	.9943	26.2876	133.9068
KASANDR	2,158,859	291,485	.9999	2.4202	51.9384

TABLE 1 – Statistics on the # of users and items ; as well as the sparsity and the average number of + (preferred) and − (non-preferred) items on different collections after preprocessing.

we compared the proposed `SAROS` algorithm [2] with the following approaches. ; `MostPop` is a non-learning based approach which consists in recommending the same set of popular items to all users ; Matrix Factorization (`MF`) [Kor08], is a factor model which decomposes the matrix of user-item interactions into a set of low dimensional vectors in the same latent space, by minimizing a regularized least square error between the actual value of the scores and the dot product over the user and item representations. `BPR` [RFGST09] corresponds to the model described in the problem statement above (Section 3.1), a stochastic gradient-descent algorithm, based on bootstrap sampling of training triplets, and $\texttt{BPR}_b$ the batch version of the model which consists in finding the model parameters $\omega = (\bar{U}, \bar{I})$ by minimizing the ranking loss over all the set of triplets simultaneously (Eq. 1). `Prod2Vec` [GRD+15], learns the representation of items using a Neural Networks based model, called word2vec [MCCD13], and performs next-items recommendation using the similarity between the representations of items. `GRU4Rec+` [HK18] is an extended version of `GRU4Rec` [HKBT16] adopted to different loss functions, that applies recurrent neural network with a GRU architecture for session-based recommendation. The approach considers the session as the sequence of clicks of the user and learns model parameters by optimizing a regularized approximation of the relative rank of the relevant item which favors the preferred items to be ranked at the top of the list. `Caser` [TW18] is a CNN based model that embeds a sequence of clicked items into a temporal image and latent spaces and find local characteristics of the temporal image using convolution filters. `SASRec` [KM18] uses an attention mechanism to capture long-term semantics in the sequence of clicked items and then predicts the next item to present based on a user's action history. `LightGCN` [HDW+20] is a graph convolution network which learns user and item embedding by linearly propagating them on the user-item interaction graph. The final representations are the weighted sum of the embeddings learned at all layers.

Hyper-parameters of different models and the dimension of the embedded space for the representation of users and items ; as well as the regularisation parameter over the norms of the embeddings for all approaches were found using grid search on the validation set.

We fixed b and B, used in $\texttt{SAROS}_b$, to respectively the minimum and the average number of blocks found on the training set of each corresponding collection. With the average number of blocks being greater than the median on all collections, the motivation here is to consider the maximum number of blocks by preserving the model from the bias brought by the too many interactions of the very few number of users.

Experimental results. We compare the performance of all the approaches on the basis of the common ranking metrics, which are the Mean Average Precision at rank K (`MAP@K`) over all users defined as $\texttt{MAP@K} = \frac{1}{N}\sum_{u=1}^{N} \texttt{AP@K}(u)$, where $\texttt{AP@K}(u)$ is the average precision of preferred items of user u in the top K ranked ones ; and the Normalized Discounted Cumulative Gain at rank K (`NDCG@K`) that computes the ratio of the obtained ranking to the ideal case and allow to consider not only binary relevance as in Mean Average Precision, $\texttt{NDCG@K} = \frac{1}{N}\sum_{u=1}^{N} \frac{\texttt{DCG@K}(u)}{\texttt{IDCG@K}(u)}$, where $\texttt{DCG@K}(u) = \sum_{i=1}^{K} \frac{2^{rel_i}-1}{\log_2(1+i)}$, rel_i is the graded relevance of the item at position i ; and $\texttt{IDCG@K}(u)$ is $\texttt{DCG@K}(u)$ with an ideal ordering equals to $\sum_{i=1}^{K} \frac{1}{\log_2(1+i)}$ for $rel_i \in [0, 1]$ [SMR08].

Table 2 presents `NDCG@5` and `NDCG@10` (top), and `MAP@5` and `MAP@10` (down) of all approaches over the test sets of the different collections. The non-machine learning method, `MostPop`, gives results of an order of magnitude lower than the learning based approaches. Moreover, the factorization model `MF` which predicts clicks by matrix completion is less effective when dealing with implicit feedback than ranking based models especially on large datasets where there are fewer interactions. We also found that embeddings found by ranking based models, in the way that the user preference over the pairs of items is preserved in the embedded space by the dot product, are more robust than the ones found by `Prod2Vec`. When comparing `GRU4Rec+` with `BPR` that also minimizes the same surrogate ranking loss, the former outperforms it in case of KASANDR with a huge imbalance between positive and negative interactions.

This is mainly because `GRU4Rec+` optimizes an ap-

2. The source code is available at `https://github.com/SashaBurashnikova/SAROS`.

	NDCG@5						NDCG@10					
	ML-1M	Outbrain	Pandor	Netflix	Kasandr	RecSys'16	ML-1M	Outbrain	Pandor	Netflix	Kasandr	RecSys'16
MostPop	.090	.011	.005	.056	.002	.004	.130	.014	.008	.096	.002	.007
Prod2Vec	.758	.232	.078	.712	.012	.219	.842	.232	.080	.770	.012	.307
MF	.684	.612	.300	.795	.197	.317	.805	.684	.303	.834	.219	.396
BPR_b	.652	.583	.874	.770	.567	.353	.784	.658	.890	.849	.616	.468
BPR	.776	.671	.889	.854	.603	**.575**	.863	.724	.905	.903	.650	**.673**
GRU4Rec+	.721	.633	.843	.777	.760	.507	.833	.680	.862	.854	.782	.613
Caser	.665	.585	.647	.750	.241	.225	.787	.658	.666	.834	.276	.225
SASRec	.721	.645	.852	.819	.569	.509	.832	.704	.873	.883	.625	.605
LightGCN	.784	.652	.901	.836	**.947**	.428	**.874**	.710	.915	.895	**.954**	.535
$SAROS_b$	**.788**	**.710**	**.904**	**.866**	.791	.563	**.874**	**.755**	**.917**	**.914**	.815	.662

	MAP@5						MAP@10					
	ML-1M	Outbrain	Pandor	Netflix	Kasandr	RecSys'16	ML-1M	Outbrain	Pandor	Netflix	Kasandr	RecSys'16
MostPop	.074	.007	.003	.039	.002	.003	.083	.009	.004	.051	.3e-5	.004
Prod2Vec	.793	.228	.063	.669	.012	.210	.772	.228	.063	.690	.012	.220
MF	.733	.531	.266	.793	.170	.312	.718	.522	.267	.778	.176	.306
BPR_b	.713	.477	.685	.764	.473	.343	.688	.477	.690	.748	.488	.356
BPR	.826	.573	.734	.855	.507	**.578**	.797	.563	.760	.835	.521	**.571**
GRU4Rec+	.777	.513	.673	.774	.719	.521	.750	.509	.677	.757	.720	.500
Caser	.718	.471	.522	.749	.186	.218	.694	.473	.527	.733	.197	.218
SASRec	.776	.542	.682	.819	.480	.521	.751	.534	.687	.799	.495	.511
LightGCN	**.836**	.502	**.793**	.835	**.939**	.428	.806	.507	**.796**	.817	**.939**	.434
$SAROS_b$	.832	**.619**	.756	**.866**	.732	.570	**.808**	**.607**	.759	**.846**	.747	.561

Table 2 – Comparison of different approaches in terms of NDCG@5 and NDCG@10(top), and MAP@5 and MAP@10(down). Best performance is in bold and the second best is underlined.

proximation of the relative rank that favors interacted items to be in the top of the ranked list while the logistic ranking loss, which is mostly related to the Area under the ROC curve [UAG05], pushes up clicked items for having good ranks in average. However, the minimization of the logistic ranking loss over blocks of very small size pushes the clicked item to be ranked higher than the no-clicked ones in several lists of small size and it has the effect of favoring the clicked item to be at the top of the whole merged lists of items. Moreover, it comes out that SAROS is the most competitive approach ; performing better than other techniques, or, is the second best performing method over all collections.

5 Conclusion

The contributions of this paper are twofold. First, we proposed SAROS, a novel learning framework for large-scale Recommender Systems that sequentially updates the weights of a ranking function user by user over blocks of items ordered by time where each block is a sequence of negative items followed by a last positive one. We bounded the deviation of the ranking loss concerning the sequence of weights found by the algorithm and its minimum in the general case of non-convex ranking loss, and showed the efficiency of the approach on six real-life implicit feedback datasets.

Références

[BL07] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD Cup and Workshop*, 2007.

[DLZ17] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *RecSYS*, pages 152–160, 2017.

[FZSG20] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. Deep Learning for Sequential Recommendation : Algorithms, Influential Factors, and Evaluations. *ACM Transactions on Information Systems*, 39(1), 2020.

[GRD+15] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox : Product recommendations at scale. In *Proceedings of SIGKDD*, pages 1809–1818, 2015.

[HDW+20] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN : Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 639–648, 2020.

[HK15] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets : History and context. In *ACM Transactions of Interaction Intelligent Systems*, pages 1–19, 2015.

[HK18] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, pages 843–852, 2018.

[HKBT16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[HM16] Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*, pages 191–200, 2016.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. In *IEEE Computer Journal*, pages 30–37, 2009.

[KG16] P. Kaur and S. Goel. Shilling attack models in recommender system. In *ICICT*, 2016.

[KM18] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *International Conference on Data Mining, ICDM*, pages 197–206, 2018.

[Kor08] Yehuda Koren. Factorization meets the neighborhood : a multifaceted collaborative filtering model. In *SIGKDD*, pages 426–434, 2008.

[Lan20] Guanghui Lan. *First-order and Stochastic Optimization Methods for Machine Learning*. Springer, 2020.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, (ICLR)*, 2013.

[PB07] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web : Methods and Strategies of Web Personalization*, pages 325–341. Springer, 2007.

[RFGST09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR : Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[SK09] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. In *Advances in artificial intelligence*, volume 2009, page Article ID 421425, 2009.

[SLA+17] Sumit Sidana, Charlotte Laclau, Massih-Reza Amini, Gilles Vandelle, and André Bois-Crettez. KASANDR : A Large-Scale Dataset with Implicit Feedback for Recommendation. In *Proceedings SIGIR*, pages 1245–1248, 2017.

[SLA18] Sumit Sidana, Charlotte Laclau, and Massih-Reza Amini. Learning to recommend diverse items over implicit feedback on PANDOR. In *Proceedings of the* 12^{th} *ACM Conference on Recommender Systems*, pages 427–431, 2018.

[SMR08] Hinrich Schutze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[STH+21] Sumit Sidana, Mikhail Trofimov, Oleh Horodnytskyi, Charlotte Laclau, Yury Maximov, and Massih-Reza Amini. User preference and embedding learning with implicit feedback for recommender systems. *Data Mining and Knowledge Discovery*, 35 :1–25, 2021.

[TB14] Maryam Tavakol and Ulf Brefeld. Factored MDPs for detecting topics of user sessions. In *RecSYS*, pages 33–40, 2014.

[TW18] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573, 2018.

[UAG05] Nicolas Usunier, Massih-Reza Amini, and Patrick Gallinari. A data-dependent generalisation error bound for the AUC. In *ICML workshop on ROC Analysis in Machine Learning*, 2005.

[WCW+19] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet Orgun, and Defu Lian. A survey on session-based recommender systems. *arXiv preprint*, 1902.04864, 2019.

[WZZ+20] Chao Wang, Hengshu Zhu, Chen Zhu, Chuan Qin, and Hui Xiong. Setrank : A setwise bayesian approach for collaborative ranking from implicit feedback. In *AAAI*, 2020.