Sequential Learning over Implicit Feedback for Robust Large-Scale Recommender Systems

Aleksandra Burashnikova[1,2], Yury Maximov[3,1], and Massih-Reza Amini[2]

[1] Skolkovo Institute of Science and Technology, Russia
a.burahnikova@skoltech.ru, y.maximov@skoltech.ru
[2] Université Grenoble Alpes, Grenoble, France
Firstname.Lastname@univ-grenoble-alpes.fr
[3] Theoretical Division T-5 and CNLS, Los Alamos National Laboratory, USA
yury@lanl.gov

Abstract. In this paper, we propose a theoretically founded sequential strategy for training large-scale Recommender Systems (RS) over implicit feedback mainly in the form of clicks. The proposed approach consists in minimizing pairwise ranking loss over blocks of consecutive items constituted by a sequence of non-clicked items followed by a clicked one for each user. Parameter updates are discarded if for a given user the number of sequential blocks is below or above some given thresholds estimated over the distribution of the number of blocks in the training set. This is to prevent from updating the parameters for an abnormally high number of clicks over some targeted items, mainly due to bots; or very few user interactions. Both scenarios affect the decision of RS and imply a shift over the distribution of items that are shown to the users. We provide a proof of convergence of the algorithm to the minimizer of the ranking loss, in the case where the latter is convex. Furthermore, experimental results on five large-scale collections demonstrate the efficiency of the proposed algorithm concerning the state-of-the-art approaches, both regarding different ranking measures and computation time.

1 Introduction

With the increasing number of products available online, there is a surge of interest in the design of automatic systems — generally referred to as Recommender Systems (RS) — 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 Prize [1]. One characteristic of online recommendation is the huge unbalance between the available number of products and those shown to the users. Another aspect is the existence of bots that interact with the system by providing too many feedback over some targeted items; or many users that 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[4]. New approaches for RS now primarily consider *implicit* feedback, mostly in the form of clicks, that are easier to collect than *explicit* feedback which is in the form of scores. Implicit feedback is more 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 [10]. For 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 a new SequentiAl RecOmmender System for implicit feedback (called `SAROS`), that updates the model parameters user per user over blocks of items constituted by a sequence of unclicked items followed by a clicked one. The parameter updates are discarded for users who interact very little or a lot with the system. For other users, the update is done by minimizing the average ranking loss of the current model that scores the clicked item below the unclicked ones in a corresponding block. Recently, many other approaches that model the sequences of users feedback have been proposed, but they all suffer from a lack of theoretical analysis formalizing the overall learning strategy. In this work, we analyze the convergence property of the proposed approach and show that in the case where the global ranking loss estimated over all users and items is convex; then the minimizer found by the proposed sequential approach converges to the minimizer of the global ranking loss. Experimental results conducted on five large publicly available datasets show that our approach is highly competitive compared to the state-of-the-art models and, it is significantly faster than both the batch and the online versions of the algorithm.

The rest of this paper is organized as follows. Section 2 relates our work to previously proposed approaches. Section 3 introduces the general ranking learning problem that we address in this study. Then, in Section 3.3, we present the `SAROS` algorithm and provide an analysis of its convergence. Section 4 presents the experimental results that support this approach. Finally, in Section 5, we discuss the outcomes of this study and give some pointers to further research.

2 Related work

Two main approaches have been proposed for recommender systems. The first one, referred to as Content-Based recommendation or cognitive filtering [18], 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 referred to as Collaborative Filtering and undoubtedly the most popular one [26], relies on past interactions and recommends items to users based on the feedback provided by other similar users. Traditionally, collaborative filtering systems were designed using *explicit* feedback, mostly in the form of rating [12]. However, rating information is non-existent on most 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*

[4] `https://www.kaggle.com/c/outbrain-click-prediction`

feedback, in order 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 are dealing with explicit feedback [7, 19, 31]. 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. In most of these approaches, the objective is to rank the clicked item higher than the unclicked ones by finding a suitable representation of users and items in a way that for each user the ordering of the clicked items over unclicked ones is respected by dot product in the joint learned space. One common characteristic of publicly available collections for recommendation systems is the huge unbalance between positive (click) and negative feedback (no-click) in the set of items displayed to the users, making the design of an efficient online RS extremely challenging. Some works propose to weight the impact of positive and negative feedback directly in the objective function [17] to improve the quality. Another approach is to sample the data over a predefined buffer before learning [14], but these approaches do not model the shift over the distribution of positive and negative items, and the system's performance on new test data may be affected. 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) [3]. Recommender systems based on Markov Models, consider the sequential 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 [23]. Various strategies have been proposed to leverage the impact of sparse data, for example by considering only the last frequent sequences of items and using finite mixture models [23], or by combining similarity-based methods with high-order Markov Chains [20]. Although it has been shown that in some cases the proposed approaches can capture the temporal aspect of user interactions but these models suffer from high complexity and generally they do not pass the scale. Some other methods consider RS as a Markov decision process (MDP) problem and solve it using reinforcement learning (RL) [16, 28]. The size of discrete actions bringing the RL solver to a larger class of problems is also a bottleneck for these approaches. Very recently Recurrent neural networks such as GRU or LSTM, have been proposed for personalized recommendations [8, 27, 11], where the input of the network is generally the current state of the session, and the output is the predicted preference over items (probabilities for each item to be clicked next). Our proposed strategy differs from other sequential based approaches in the way that the model parameters are updated, at each time a block of unclicked items followed by a clicked one is constituted; and by controlling the number of blocks per user interaction. If for a given user, this number is below or above two predefined thresholds found over the distribution of the number of blocks, parameter updates for that particular user are discarded. Ultimately, we provide a proof of convergence of the proposed approach.

3 Framework and Problem Setting

Throughout, we use the following notation. For any positive integer n, $[n]$ denotes the set $[n] \doteq \{1, \dots, n\}$. We suppose that $\mathcal{I} \doteq [M]$ and $\mathcal{U} \doteq [N]$ are two sets of indexes defined over items and users. Further, we assume that each pair constituted by a user u and an item i is identically and independently distributed according to a fixed yet unknown distribution $\mathcal{D}_{\mathcal{U},\mathcal{I}}$.

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 $i \succcurlyeq_u i'$. 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 $i \succcurlyeq i'$. 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 $\mathcal{B}_u^t = \mathrm{N}_u^t \sqcup \mathit{\Pi}_u^t$, 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, $\mathit{\Pi}_u^t$. Hence, $\mathcal{I}_u^+ = \bigcup_t \mathit{\Pi}_u^t$ 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 :

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

and $\mathcal{L}(\boldsymbol{\omega}) = \mathbb{E}_u \left[\hat{\mathcal{L}}_u(\boldsymbol{\omega}) \right]$, where $\mathbb{E}_u$ is the expectation with respect to users chosen randomly according to the uniform distribution, and $\hat{\mathcal{L}}_u(\boldsymbol{\omega})$ is the pairwise ranking loss with respect to user u's interactions. As in other studies, we represent each user u and each item i respectively by vectors $\mathbf{U}_u \in \mathbb{R}^k$ and $\mathbf{V}_i \in \mathbb{R}^k$ in the same latent space of dimension k [13]. The set of weights to be found $\boldsymbol{\omega} = (\mathbf{U}, \mathbf{V})$, are then matrices formed by the vector representations of users $\mathbf{U} = (\mathbf{U}_u)_{u \in [N]} \in \mathbb{R}^{N \times k}$ and items $\mathbf{V} = (\mathbf{V}_i)_{i \in [M]} \in \mathbb{R}^{M \times k}$. The minimization of the ranking loss above in the batch mode with the goal of finding user and item embeddings, such that the dot product between these representations in the latent space reflects the best the preference of users over items, is a common approach. Other strategies have been proposed for the minimization of the empirical loss (1), among which the most popular one is perhaps the Bayesian Personalized Ranking (`BPR`) model [19]. In this approach, the instantaneous loss,

$\ell_{u,i,i'}$, is the surrogate regularized logistic loss for some hyperparameter $\mu \geq 0$:

$$\ell_{u,i,i'}(\boldsymbol{\omega}) = \log\left(1 + e^{-y_{i,u,i'}\mathbf{U}_u^\top(\mathbf{V}_i - \mathbf{V}_{i'})}\right) + \mu(\|\mathbf{U}_u\|_2^2 + \|\mathbf{V}_i\|_2^2 + \|\mathbf{V}_{i'}\|_2^2) \tag{2}$$

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 over the instantaneous loss (2). In this case, the expected number of rejected pairs is proportional to $O(|\mathcal{I}_u|^2)$ [22] which may be time-consuming in general. Another drawback is that user preference over items depend mostly on the context where these items are shown to the user. A user may prefer (or not) two items independently one from another, but within a given set of shown items, he or she may completely have a different preference over these items. By sampling items over the whole set of shown items, this effect of local preference is unclear.

3.2 Algorithm SAROS

Another particularity of online recommendation which is not explicitly taken into account by existing approaches is the bot attacks in the form of excessive clicks over some target items. They are made to force the RS to adapt its recommendations toward these target items, or a very few interactions which in both cases introduce biased data for the learning of an efficient RS. In order to tackle these points, our approach updates the parameters whenever the number of constituted blocks per user is lower and upper-bounded (Figure 1).

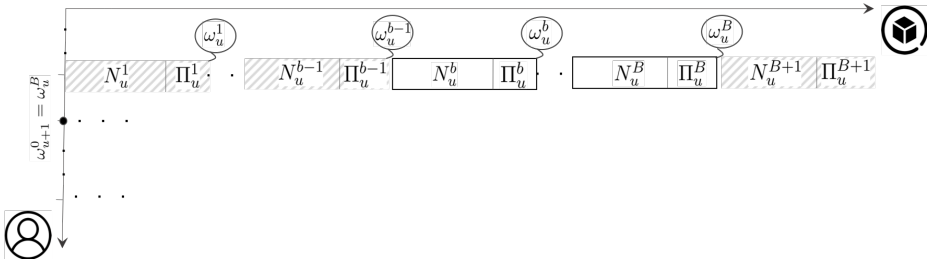


Fig. 1: A pictorial depiction of the sequential updates of weights $(\omega_u^t)_{1 \leq t \leq B}$ for a user $u \in \mathcal{U}$. The horizontal axis represents the sequence of interactions over items ordered by time. Each update of weights $\omega_u^t; t \in \{b, \ldots, B\}$ occurs whenever the corresponding sets of negative interactions, N_u^t, and positive ones, $\mathit{\Pi}_u^t$, exist, and that these number of interactions is lower and upper-bounded. For a new user $u+1$, the initial weights $\omega_{u+1}^0 = \omega_u^B$ are the ones obtained from the last update of the previous user's interactions.

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

$$\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\boldsymbol{\omega}_u^t) = \frac{1}{|\mathit{\Pi}_u^t||\mathrm{N}_u^t|} \sum_{i \in \mathit{\Pi}_u^t} \sum_{i' \in \mathrm{N}_u^t} \ell_{u,i,i'}(\boldsymbol{\omega}_u^t). \tag{3}$$

The pseudo-code of SAROS is shown in the following. Starting from initial weights $\boldsymbol{\omega}_1^0$ chosen randomly for the first user. For each current user u, having been shown I_u items, the sequential update rule consists in updating the weights, block by block where after t updates; where the $(t+1)^{th}$ update over the current block $\mathcal{B}_u^t = \mathrm{N}_u^t \sqcup \mathit{\Pi}_u^t$ corresponds to one gradient descent step over the ranking loss estimated on these sets and which with the current weights $\boldsymbol{\omega}_u^t$ writes,

$$\boldsymbol{\omega}_u^{t+1} \leftarrow \boldsymbol{\omega}_u^t - \eta \nabla \hat{\mathcal{L}}_{\mathcal{B}_u^t}(\boldsymbol{\omega}_u^t) \tag{4}$$

To prevent from a very few interactions or from bot attacks, two thresholds b and B are fixed over the parameter updates. For a new user $u+1$, the parameters are initialized as the last updated weights from the previous user's interactions in the case where the corresponding number of updates t was in the interval $[b, B]$; i.e. $\omega_{u+1}^0 = \omega_u^t$. On the contrary case, they are set to the same previous initial parameters; i.e., $\omega_{u+1}^0 = \omega_u^0$.

3.3 Convergence analysis

We provide proofs of convergence for the SAROS algorithm under the typical hypothesis that the system is not instantaneously affected by the sequential learning of the weights. This hypothesis stipulates that the generation of items shown to users is independently and identically distributed with respect to some stationary in time underlying distribution $\mathcal{D}_\mathcal{I}$, and

Algorithm SAROS: SequentiAl RecOmmender System

Input: A time-ordered sequence (user and items) $\{(u, (i_1, \ldots, i_{|I_u|})\}_{u=1}^N$ drawn i.i.d. from $\mathcal{D}_{\mathcal{U},\mathcal{I}}$
Input: maximal B and minimal b number of blocks allowed per user u
Input: number of epochs E
Input: initial parameters ω_1^0, and (possibly non-convex) surrogate loss function $\ell(\omega)$
for $e \in E$ **do**
  for $u \in \mathcal{U}$ **do**
    Let $\mathrm{N}_u^t = \varnothing$, $\mathit{\Pi}_u^t = \varnothing$ be the sets of positive and negative items, counter $t = 0$
    for $i_k \in \mathcal{I}_u$ **do** ▷ Consider all items displayed to user u
      while $t \leq B$ **do**
        if u provides a negative feedback on item i_k **then**
          $\mathrm{N}_u^t \leftarrow \mathrm{N}_u^t \cup \{i_k\}$
        else
          $\mathit{\Pi}_u^t \leftarrow \mathit{\Pi}_u^t \cup \{i_k\}$
        end if
        if $\mathrm{N}_u^t \neq \varnothing$ and $\mathit{\Pi}_u^t \neq \varnothing$ and $t \leq B$ **then**
          $\boldsymbol{\omega}_u^{t+1} \leftarrow \boldsymbol{\omega}_u^t - \frac{\eta}{|\mathrm{N}_u^t||\mathit{\Pi}_u^t|} \sum_{i \in \mathit{\Pi}_u^t} \sum_{i' \in \mathrm{N}_u^t} \nabla \ell_{u,i,i'}(\boldsymbol{\omega}_u^t)$
          $t = t + 1, \mathrm{N}_u^t = \varnothing, \mathit{\Pi}_u^t = \varnothing$
        end if
      end while
    end for
    if $t \geq b$ **then**
      $\omega_{u+1}^0 = \omega_u^t$
    else
      $\omega_{u+1}^0 = \omega_u^0$
    end if
  end for
end for
Return: $\bar{\omega}_N = \sum_{u \in \mathcal{U}} \omega_u^0$

constitutes the main hypothesis of almost all the existing studies. Furthermore, we make the following technical assumption.

Assumption 1 *Let the loss functions $\ell_{u,i,i'}(\omega)$ and $\mathcal{L}(\omega)$, $\omega \in \mathbb{R}^d$ be such that for some absolute constants $\gamma \geq \beta > 0$ and $\sigma > 0$:*

1. *$\ell_{u,i,i'}(\omega)$ is non-negative for any user u and a pair of items (i, i');*
2. *$\ell_{u,i,i'}(\omega)$ is twice continuously differentiable, and for any user u and a pair of items (i, i') one has $\gamma\|\omega - \omega'\|_2 \geq \|\nabla\ell_{u,i,i'}(\omega) - \nabla\ell_{u,i,i'}(\omega')\|_2$, as well as $\beta\|\omega - \omega'\|_2 \geq \|\nabla\mathcal{L}(\omega) - \nabla\mathcal{L}(\omega')\|_2$,.*
3. *Variance of the empirical loss is bounded $\mathbb{E}_{\mathcal{D}} \left\|\nabla\hat{\mathcal{L}}_u(\omega) - \nabla\mathcal{L}(\omega)\right\|_2^2 \leq \sigma^2$.*

Moreover, there exist some positive lower and upper bounds b and B, such that the number of updates for any u is within the interval $[b, B]$ almost surely.

Our main result is the following theorem which provides a bound over the deviation of the ranking loss with respect to the sequence of weights found by the SAROS algorithm and its minimum in the case where the latter is convex.

Theorem 1. *Let $\ell_{u,i,i'}(\omega)$ and $\mathcal{L}(\omega)$ satisfy Assumption 1. Then for any constant step size η, verifying $0 < \eta \leq \min(\frac{1}{\beta B}, 1/\sqrt{UB(\sigma^2 + 3\gamma^2/b)})$, and any set of users $\mathcal{U} \doteq [U]$; algorithm SAROS iteratively generates a sequence $\{\omega_u^0\}_{u\in\mathcal{U}}$ such that*

$$\frac{1}{\beta}\mathbb{E}\|\nabla\mathcal{L}(\omega_u^0)\|_2^2 \leq \frac{\beta B\Delta_{\mathcal{L}}^2}{u} + 2\Delta_{\mathcal{L}}\sqrt{\frac{B\sigma^2 + 3B\gamma^2/b}{u}},$$

where $\Delta_{\mathcal{L}}^2 = \frac{2}{\beta}(\mathcal{L}(\omega_0) - \mathcal{L}(\omega^*))$, and the expectation is taken with respect to users chosen randomly according to the uniform distribution $p_u = \frac{1}{N}$.

Furthermore, if the ranking loss $\mathcal{L}(\omega)$ is convex, then for the sequence $\{\omega_u^0\}_{u\in\mathcal{U}}$ generated by algorithm SAROS and $\bar{\omega}_u = \sum_{j\leq u}\omega_j^0$ we have

$$\mathcal{L}(\bar{\omega}_u) - \mathcal{L}(\omega_*) \leq \frac{\beta B\Delta_{\omega}^2}{u} + 2\Delta_{\omega}\sqrt{\frac{B\sigma^2 + 3B\gamma^2/b}{u}},$$

where $\Delta_{\omega} = \|\omega_0 - \omega_*\|_2^2$, and $\omega_* = \arg\min_{\omega}\mathcal{L}(\omega)$.

Proof. *Sketch.* Expectation of the empirical loss taken over a random block $\mathcal{B}_u^t$ for a user u, equals to the expected loss for this user. Then by the law of total expectation one has $\mathbb{E}_{\mathcal{D}_u}\left[k^{-1}\sum_{l=1}^k \nabla\hat{\mathcal{L}}_{\mathcal{B}_u^l}(\boldsymbol{\omega})\right] = \nabla\hat{\mathcal{L}}_u(\omega)$, where $\mathcal{D}_u$ is the

conditional distribution of items for a fixed user u. The variance of the gradient estimation over k blocks is bounded by $3\gamma^2/k$, as for any block after the next to $\mathcal{B}_u^t$ and before the previous to $\mathcal{B}_u^t$ are conditionally independent for any fixed $\mathcal{B}_u^t$.

Let g_u^t be a gradient of the loss function taken for user u over block $\mathcal{B}_u^t$:

$$g_u^t = \frac{1}{|N_u^t||\Pi_u^t|} \sum_{i \in N_u^t, i' \in \Pi_u^t} \nabla \ell_{u,i,i'}(\omega_u^{t-1}),$$

According to the notation of Algorithm SAROS let $\delta_u^t = g_u^t - \nabla\mathcal{L}(\omega_u^0)$ and $\omega_u^{t+1} = \omega_u^t - \eta g_u^t$, $\omega_{u+1}^0 = \omega_u^{|\mathcal{B}_u|}$, and $\omega_{u+1}^0 - \omega_u^0 = \eta \sum_{t \in \mathcal{B}_u} g_u^t$, where $\mathcal{B}_u$ is the set of all interacted blocks corresponding to user u. Using the smoothness of the loss function implied by Assumption 1, it comes :

$$\begin{aligned}\mathcal{L}(\omega_{u+1}^0) = \mathcal{L}(\omega_u^0) - \left(\hat{\eta}_u - \frac{\beta}{2}\hat{\eta}_u^2\right) \|\nabla\mathcal{L}(\omega_u^0)\|_2^2 \\ - (\hat{\eta}_u - \beta\hat{\eta}_u^2) \sum_{t \in \mathcal{B}_u} \left\langle \nabla\mathcal{L}(\omega_u^0), \frac{\delta_u^t}{|\mathcal{B}_u|} \right\rangle + \frac{\beta}{2}\hat{\eta}_u^2 \sum_{t \in \mathcal{B}_u} \left\| \frac{\delta_u^t}{|\mathcal{B}_u|} \right\|_2^2 \end{aligned} \tag{5}$$

where $\hat{\eta}_u = |\mathcal{B}_u|\eta$. Then by re-arranging and summing up, we get

$$\begin{aligned}\sum_{u=1}^{N} \left(\hat{\eta}_u - \frac{\beta}{2}\hat{\eta}_u^2\right) \|\nabla\mathcal{L}(\omega_u)\|_2^2 \leq \mathcal{L}(\bar{\omega}_u) - \mathcal{L}(\omega^*) \\ - \sum_{u=1}^{N} (\hat{\eta}_u - \beta\hat{\eta}_u^2) \left\langle \nabla\mathcal{L}(\omega_u), \sum_{t \in \mathcal{B}_u} \frac{\delta_u^t}{|\mathcal{B}_u|} \right\rangle + \frac{\beta}{2} \sum_{u=1}^{N} \hat{\eta}_u^2 \left\| \sum_{t \in \mathcal{B}_u} \frac{\delta_u^t}{|\mathcal{B}_u|} \right\|_2^2 \end{aligned}$$

As the stochastic gradient taken with respect to a block of items gives an unbiased estimate of the gradient, thus

$$\mathbb{E}_{\mathcal{D}_u} \left[\left\langle \nabla\mathcal{L}(\omega_u), \sum_{t \in \mathcal{B}_u} \frac{\delta_u^t}{|\mathcal{B}_u|} \right\rangle \middle| \xi_u \right] = 0, \tag{6}$$

where ξ_u is a set of users preceding u. As in the conditions of the theorem $b \leq |\mathcal{B}_u|$ almost surely, by the law of total variation, $\operatorname{Var}\psi = \mathbb{E}[\operatorname{Var}(\psi|\eta)] + \operatorname{Var}[\mathbb{E}[\psi|\eta]]$:

$$\mathbb{E}_{\mathcal{D}_u} \left\| \sum_{t \in \mathcal{B}_u} \frac{\delta_u^t}{|\mathcal{B}_u|} \right\|_2^2 \leq \sigma^2 + \frac{3\gamma^2}{b} \tag{7}$$

where the first term on the right-hand side of Eq. (7) comes from Assumption 1, and the second term is due to the variance estimate. Condition $\beta\eta B \leq 1$ implies $\hat{\eta}_u - \beta\hat{\eta}_u^2/2 \geq \hat{\eta}_u/2$, thus

$$\frac{1}{\beta}\mathbb{E}_{\mathcal{D}} \|\nabla\mathcal{L}(\omega)\|_2^2 \leq \frac{1}{\sum_{u=1}^{N} \hat{\eta}_u} \left[\frac{2(\mathcal{L}(\omega_0) - \mathcal{L}(\omega_*))}{\beta} + \left(\sigma^2 + 3\frac{\gamma^2}{b}\right) \sum_{u=1}^{N} \hat{\eta}_u^2 \right]$$

The rest of the proof of the theorem comes along the same lines according to the randomized stochastic gradient descent analysis [4]. □

The full proof is provided in the Supplementary. This result implies that the loss over a sequence of weights $(\omega_u^0)_{u\in\mathcal{U}}$ generated by the algorithm converges to the true minimizer of the ranking loss $\mathcal{L}(\omega)$ with a rate proportional to $O(1/\sqrt{u})$. The stochastic gradient descent strategy implemented in the Bayesian Personalized Ranking model (`BPR`) [19] also converges to the minimizer of the ranking loss $\mathcal{L}(\omega)$ with the same rate. However, the main difference between `BPR` and `SAROS` is their computation time. As stated in section 3.1, the expected number of rejected random pairs sampled by algorithm `BPR` before making one update is $O(|\mathcal{I}_u|^2)$ while with `SAROS`, blocks are created sequentially as and when users interact with the system. For each user u, weights are updated whenever a block is created, with the overall complexity of $O(\max_t(|\Pi_u^t| \times |\mathrm{N}_u^t|))$, with $\max_t(|\Pi_u^t| \times |\mathrm{N}_u^t|) \ll |\mathcal{I}_u|^2$.

4 Experimental Setup and Results

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 [5] and computed on Skoltech CDISE HPC cluster "Zhores" [30]. 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 [6] and NETFLIX [2] 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 [2].For both datasets, we consider ratings greater or equal to 4 as positive feedback, and negative feedback otherwise.
- We extracted a subset out of the OUTBRAIN dataset from of the Kaggle challenge[6] 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.
- KASANDR[7] dataset [25] contains 15,844,717 interactions of 2,158,859 users in Germany using Kelkoo's (`http://www.kelkoo.fr/`) online advertising platform.
- PANDOR[8] is another publicly available dataset for online recommendation [24] provided by Purch (`http://www.purch.com/`). The dataset records

[5] `https://www.tensorflow.org/`.
[6] `https://www.kaggle.com/c/outbrain-click-prediction`
[7] `https://archive.ics.uci.edu/ml/datasets/KASANDR`
[8] `https://archive.ics.uci.edu/ml/datasets/PANDOR`

Data	$\|\mathcal{U}\|$	$\|\mathcal{I}\|$	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
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 ML-1M, Netflix, Outbrain, Kasandr and Pandor collections after preprocessing.

Dataset	$\|S_{train}\|$	$\|S_{test}\|$	pos_{train}	pos_{test}
ML-1M	797,758	202,451	58.82	52.39
Outbrain	1,261,373	336,053	17.64	24.73
Pandor	1,579,716	493,663	11.04	12.33
Netflix	3,314,621	873,477	56.27	56.70
RecSys'16	5,048,653	1,281,909	17.07	13.81
Kasandr	12,509,509	3,335,208	3.36	8.56

Table 2: Number of interactions used for train and test on each dataset, and the percentage of positive feedback among these interactions.

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.

Table 1 presents some detailed statistics about each collection. Among these, we report the average number of positive (click, like) feedback and the average number of 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. Table 2 presents the size of the training and the test sets as well as the percentage of positive feedback (preferred items) for all collections ordered by increasing training size. The percentage of positive feedback is inversely proportional to the size of the training sets, attaining 3% for the largest, Kasandr collection.

We also analyzed the distributions of the number of blocks and their size for different collections. Figure 2 (left) shows boxplots representing the logarithm of the number of blocks through their quartiles for all collections. From these plots, it comes out that the distribution of the number of blocks on Pandor, Netflix and Kasandr are heavy-tailed with more than the half of the users interacting no more than 10 times with the system. Furthermore, we note that on Pandor the average number of blocks is much smaller than on the two

other collections; and that on all three collections the maximum numbers of blocks are 10 times more than the average. These plots suggest that a very small number of users (perhaps bots) have an abnormal interaction with the system generating a huge amount of blocks on these three collections. To have a better understanding, Figure 2 (right) depicts the number of blocks concerning their size on KASANDR. The distribution of the number of blocks follows a power law distribution and it is the same for the other collections that we did not report for the sake of space. In all collections, the number of blocks having more than 5 items drops drastically. As the SAROS does not sample positive and negative items for updating the weights, these updates are performed on blocks of small size, and are made very often.

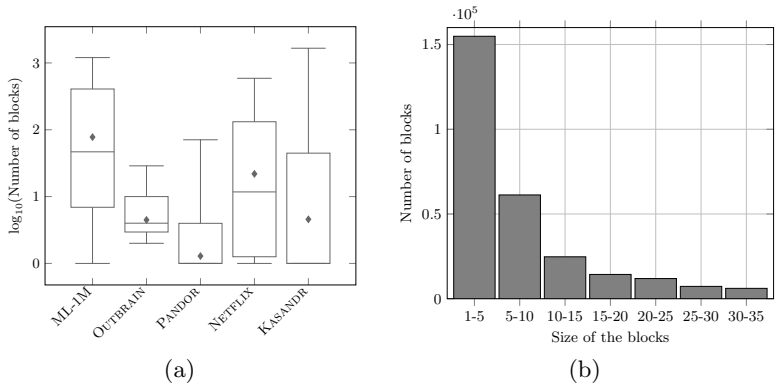


Fig. 2: (a) Boxplots depicting the logarithm of the number of blocks through their quartiles for all collections. The median (resp. mean) is represented by the band (resp. diamond) inside the box. The ends of the whiskers represent the minimum and the maximum of the values. (b) Distributions of negative feedback over the blocks in the training set on KASANDR.

Compared approaches. To validate the sequential approach described earlier, we compared the proposed SAROS algorithm[9] with the following methods:

- MostPop is a non-learning based approach which consists in recommending the same set of popular items to all users.
- Matrix Factorization (MF) [12], 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.

[9] The code is available on https://github.com/SashaBurashnikova/SAROS.

- `BPR` [19] 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 $\boldsymbol{\omega} = (\mathbf{U}, \mathbf{V})$ by minimizing the global ranking loss over all the set of triplets simultaneously (Eq. 1).
- `Prod2Vec` [5], learns the representation of items using a Neural Networks based model, called word2vec [15], and performs next-items recommendation using the similarity between the representations of items.
- `GRU4Rec+` [8] is an extended version of `GRU4Rec` [9] 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 that depends on all the previous ones for learning the 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` [27] is a CNN based model that embeds a sequence of interactions into a temporal image and latent spaces and find local characteristics of the temporal image using convolution filters.
- `SASRec` [11] uses an attention mechanism to capture long-term semantics and then predicts the next item to present based on a user's action history.

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 `BPR`, $\texttt{BPR}_b$, `MF`, `Caser` and `SAROS` approaches were found by cross-validation. We fixed b and B, used in `SAROS`, 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. For more details regarding the exact values for the parameters, see the Table 3.

Parameter	ML	Outbrain	Pandor	Netflix	Kasandr
B	78	5	2	22	5
b	1	2	1	1	1
Learning rate	.05	.05	.05	.05	.4

Table 3: Values for the `SAROS` parameters.

Evaluation setting and results. We begin our comparisons by testing $\texttt{BPR}_b$, `BPR` and `SAROS` approaches over the logistic ranking loss (Eq. 2) which is used to train them. Results on the test, after training the models 30 minutes and at convergence are shown in Table 4. $\texttt{BPR}_b$ (resp. `SAROS`) techniques have the worse (resp. best) test loss on all collections, and the difference between their performance is larger for bigger size datasets.

Dataset	Test Loss, Eq. (1)					
	30 min			At convergence		
	$\texttt{BPR}_b$	BPR	SAROS	$\texttt{BPR}_b$	BPR	SAROS
ML-1M	0.751	0.678	**0.623**	0.744	0.645	**0.608**
Outbrain	0.753	0.650	**0.646**	0.747	0.638	**0.635**
Pandor	0.715	0.671	**0.658**	0.694	0.661	**0.651**
Netflix	0.713	0.668	**0.622**	0.694	0.651	**0.614**
Kasandr	0.663	0.444	**0.224**	0.631	0.393	**0.212**

Table 4: Comparison between BPR, $\texttt{BPR}_b$ and SAROS approaches in terms on test loss after 30 minutes of training and at convergence.

These results suggest that the local ranking between preferred and no-preferred items present in the blocks of the training set reflects better the preference of users than the ranking of random pairs of items or their global ranking without this contextual information. Furthermore, as in SAROS updates occur after the creation of a block, and that the most of the blocks contain very few items (Figure 2 - right), weights are updated more often than in BPR or $\texttt{BPR}_b$. This is depicted in Figure 3 which shows the evolution of the training error over time for $\texttt{BPR}_b$, BPR and SAROS on all collections. As we can see, the training error decreases in all cases, and theoretically, the three approaches converge to the same minimizer of the ranking loss (Eq. 1). However, the speed of convergence is much faster with SAROS.

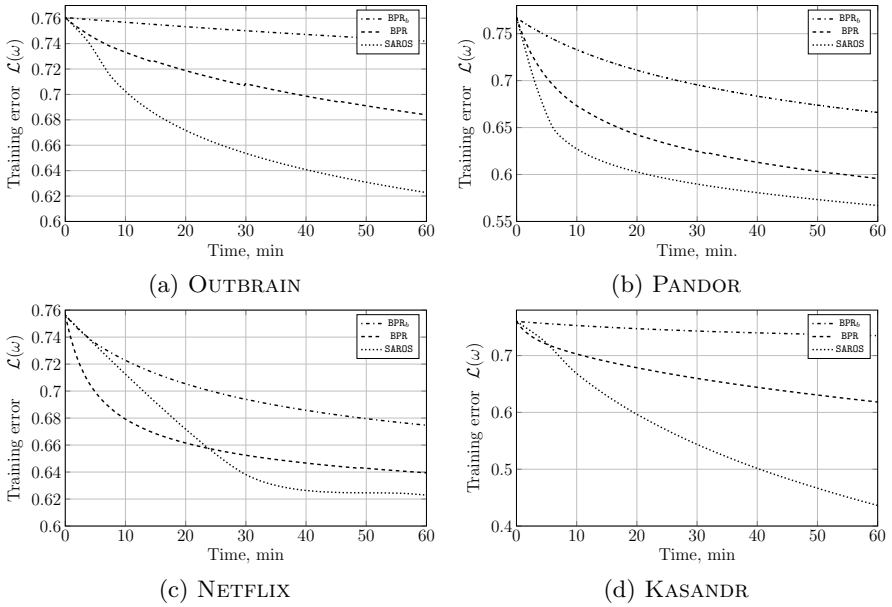
We also compare the performance of all the approaches on the basis of the common ranking metrics, which are the Mean Average Precision at rank K (MAP@K) over all users defined as $\texttt{MAP@K} = \frac{1}{N}\sum_{u=1}^{N} \texttt{AP@K}(u)$, where $\texttt{AP@K}(u)$ is the average precision of preferred 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 $rel_i \in [0, 1]$ [21].

Table 5 presents MAP@5 and MAP@10 (top), and NDCG@5 and NDCG@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 approximation 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 [29], 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 approaches over all collections even such as last published Caser and SASRec.

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. The main hypothesis of the approach is that the preferred and no-preferred items within a local sequence of user interactions express better the



(a) OUTBRAIN (b) PANDOR

(c) NETFLIX (d) KASANDR

Fig. 3: Evolution of the loss on training sets for both BPR$_b$, BPR and SAROS as a function of time in minutes for all collections.

	MAP@5					MAP@10				
	ML-1M	Outbrain	Pandor	Netflix	Kasandr	ML-1M	Outbrain	Pandor	Netflix	Kasandr
MostPop	.074	.007	.003	.039	.002	.083	.009	.004	.051	.3e-5
Prod2Vec	.793	.228	.063	.669	.012	.772	.228	.063	.690	.012
MF	.733	.531	.266	.793	.170	.718	.522	.267	.778	.176
BPR_b	.713	.477	.685	.764	.473	.688	.477	.690	.748	.488
BPR	<u>.826</u>	<u>.573</u>	<u>.734</u>	<u>.855</u>	.507	<u>.797</u>	<u>.563</u>	<u>.760</u>	<u>.835</u>	.521
GRU4Rec+	.777	.513	.673	.774	<u>.719</u>	.750	.509	.677	.757	<u>.720</u>
Caser	.718	.471	.522	.749	.186	.694	.473	.527	.733	.197
SASRec	.776	.542	.682	.819	.480	.751	.534	.687	.799	.495
SAROS	**.837**	**.619**	**.750**	**.866**	**.732**	**.808**	**.607**	**.753**	**.846**	**.747**

	NDCG@5					NDCG@10				
	ML-1M	Outbrain	Pandor	Netflix	Kasandr	ML-1M	Outbrain	Pandor	Netflix	Kasandr
MostPop	.090	.011	.005	.056	.002	.130	.014	.008	.096	.002
Prod2Vec	.758	.232	.078	.712	.012	.842	.232	.080	.770	.012
MF	.684	.612	.300	.795	.197	.805	.684	.303	.834	.219
BPR_b	.652	.583	.874	.770	.567	.784	.658	.890	.849	.616
BPR	<u>.776</u>	<u>.671</u>	<u>.889</u>	<u>.854</u>	.603	<u>.863</u>	<u>.724</u>	<u>.905</u>	<u>.903</u>	.650
GRU4Rec+	.721	.633	.843	.777	<u>.760</u>	.833	.680	.862	.854	<u>.782</u>
Caser	.665	.585	.647	.750	.241	.787	.658	.666	.834	.276
SASRec	.721	.645	.852	.819	.569	.832	.704	.873	.883	.625
SAROS	**.788**	**.710**	**.903**	**.865**	**.791**	**.874**	**.755**	**.913**	**.914**	**.815**

Table 5: Comparison between MostPop, Prod2Vec, MF, BPR_b, BPR, GRU4Rec+, SASRec, Caser and SAROS approaches in terms of MAP@5 and MAP@10(top), and NDCG@5 and NDCG@10(down). Best performance is in bold and the second best is underlined.

user preference than when considering the whole set of preferred and no-preferred items independently one from another. The approach updates the model parameters user per user over blocks of items constituted by a sequence of unclicked items followed by a clicked one. The parameter updates are discarded for users who interact very little or a lot with the system. The second contribution is a theoretical analysis of the proposed approach which bounds the deviation of the ranking loss concerning the sequence of weights found by the algorithm and its minimum in the case where the loss is convex. Empirical results conducted on five real-life implicit feedback datasets support our founding and show that the proposed approach is significantly faster than the common batch and online optimization strategies that consist in updating the parameters over the whole set of users at each epoch, or after sampling random pairs of preferred and no-preferred items. The approach is also shown to be highly competitive concerning state of the art approaches on MAP and NDCG measures.

6 Acknowledgements

This work at Los Alamos was supported by the U.S. Department of Energy through the Los Alamos National Laboratory as part of LDRD and the DOE Grid Modernization Laboratory Consortium (GMLC). Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001).

References

1. J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD Cup and Workshop*, 2007.
2. J. Bennett, S. Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA., 2007.
3. T. Donkers, B. Loepp, and J. Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 152–160, 2017.
4. S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.
5. M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of SIGKDD*, pages 1809–1818, 2015.
6. F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions of Interaction Intelligent Systems*, 5(4):1–19, 2015.
7. X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558, 2016.
8. B. Hidasi and A. Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of CIKM*, pages 843–852, 2018.
9. B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.
10. Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE ICDM*, pages 263–272, 2008.
11. W. Kang and J. McAuley. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining, ICDM*, pages 197–206, 2018.
12. Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.
13. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 8:30–37, 2009.
14. C.-L. Liu and X.-W. Wu. Large-scale recommender system with compact latent factor model. *Expert Systems and Applications*, 64(C):467–475, Dec. 2016.
15. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
16. O. Moling, L. Baltrunas, and F. Ricci. Optimal radio channel recommendations with explicit and implicit feedback. In *RecSys '12 Proceedings of the sixth ACM conference on Recommender systems*, pages 75–82. ACM, 2012.
17. R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, 2008.

18. M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.
19. S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.
20. H. Ruining and M. Julian. Fusing similarity models with markov chains for sparse sequential recommendation. In *IEEE ICDM*, 2016.
21. H. Schutze, C. D. Manning, and P. Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press, 2008.
22. D. Sculley. Large scale learning to rank. In *In NIPS 2009 Workshop on Advances in Ranking*, 2009.
23. G. Shani, D. Heckerman, and R. I. Brafman. An MDP-based recommender system. *Journal of Machine Learning Research*, 6, 2005.
24. S. Sidana, C. Laclau, and M. R. 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.
25. S. Sidana, C. Laclau, M. R. Amini, G. Vandelle, and A. Bois-Crettez. KASANDR: A Large-Scale Dataset with Implicit Feedback for Recommendation. In *Proceedings SIGIR*, pages 1245–1248, 2017.
26. X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009.
27. J. Tang and K. Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *ACM International Conference on Web Search and Data Mining*, 2018.
28. M. Tavakol and U. Brefeld. Factored MDPs for detecting topics of user sessions. In *RecSys '14 Proceedings of the 8th ACM Conference on Recommender systems*, pages 33–40. ACM, 2014.
29. N. Usunier, M.-R. Amini, and P. Gallinari. A data-dependent generalisation error bound for the auc. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.
30. I. Zacharov, R. Arslanov, M. Gunin, D. Stefonishin, S. Pavlov, O. Panarin, A. Maliutin, S. G. Rykovanov, and M. Fedorov. Zhores - petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology. *CoRR*, abs/1902.07490, 2019.
31. R. Zhang, H. Bao, H. Sun, Y. Wang, and X. Liu. Recommender systems based on ranking performance optimization. *Frontiers of Computer Science*, 10(2):270—280, 2016.