Journal of Artificial Intelligence Research 71 (2021) 121-142 Submitted 12/2020; published 05/2021

Learning over No-Preferred and Preferred Sequence of Items for Robust Recommendation

Aleksandra Burashnikova ALEKSANDRA.BURASHNIKOVA@SKOLTECH.RU
Skolkovo Institute of Science and Technology
Moscow, 125349, Russia, and
University Grenoble-Alpes
Grenoble, CS 40700 - 38058, France

Yury Maximov YURY@LANL.GOV
Theoretical Division, T-5
Los Alamos National Laboratory, USA
Los Alamos, NM 87545, USA

Marianne Clausel MARIANNE.CLAUSEL@UNIV-LORRAINE.FR
Department of Statistics
University of Lorraine
Nancy, F-54506, France

Charlotte Laclau CHARLOTTE.LACLAU@UNIV-ST-ETIENNE.FR
Hubert Curien Laboratory
Telecom Saint-Etienne
Saint-Etienne, CS 82301-42023, France

Franck Iutzeler FRANCK.IUTZELER@UNIV-GRENOBLE-ALPES.FR
Jean Kuntzmann Laboratory
University Grenoble Alpes
Grenoble, F-38000, France

Massih-Reza Amini MASSIH-REZA.AMINI@UNIV-GRENOBLE-ALPES.FR
Grenoble Informatics Laboratory
University Grenoble Alpes
Grenoble, CS 40700 - 38058, France

Abstract

In this paper, we propose a theoretically supported sequential strategy for training a large-scale Recommender System (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. We present two variants of this strategy where model parameters are updated using either the momentum method or a gradient-based approach. To prevent updating the parameters for an abnormally high number of clicks over some targeted items (mainly due to bots), we introduce an upper and a lower threshold on the number of updates for each user. These thresholds are estimated over the distribution of the number of blocks in the training set. They affect the decision of RS by shifting the distribution of items that are shown to the users. Furthermore, we provide a convergence analysis of both algorithms and demonstrate their practical efficiency over six large-scale collections with respect to various ranking measures and computational time.

1. Introduction

With the increasing number of online products, 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 Price (Bennett and Lanning, 2007a). 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 (Kaur and Goel, 2016). 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 accurate 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 (Outbrain Inc., 2016). Recent approaches for RS (Wang et al., 2020; Yi et al., 2019) now primarily consider feedback, mainly 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 (Hu et al., 2008). In this case, most of the developed approaches are based on the Learning-to-rank paradigm and focus on leveraging the click information over the unclick one without considering the sequence of users' interactions.

In this paper, we propose `SAROS`, a sequential learning 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. We present two variants of this strategy. The first approach, referred to as $\texttt{SAROS}_m$, updates the model parameters at each time a block of unclicked items followed by a clicked one is formed after a user's interaction. Parameters' updates are carried out by minimizing the average ranking loss of the current model that scores the clicked item below the unclicked ones using a momentum method (Polyak, 1964; Nesterov, 1983; Nesterov, 2018). The second strategy, which we refer to as $\texttt{SAROS}_b$, updates the model parameters by minimizing a ranking loss over the same blocks of unclicked items followed by a clicked one using a gradient descent approach; with the difference that parameter updates are discarded for users who interact very little or a lot with the system.

In this paper,

- We propose a unified framework in which we study the convergence properties of both versions of `SAROS` in the general case of non-convex ranking losses. This is an extension of our earlier results (Burashnikova et al., 2019), where only the convergence of $\texttt{SAROS}_b$ was studied in the case of convex ranking losses.

- Furthermore, we provide empirical evaluation over six large publicly available datasets showing that both versions of `SAROS` are 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.

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 both versions of the SAROS algorithm, $\texttt{SAROS}_b$ and $\texttt{SAROS}_m$, and provide an analysis of their convergence. Section 4 presents experimental results that support our approach. Finally, in Section 5, we discuss the outcomes of this study and give some pointers to further research.

2. Related Works

Two main approaches have been proposed for recommender systems. The first one, Content-Based recommendation or cognitive filtering (Pazzani and Billsus, 2007), 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 (Su and Khoshgoftaar, 2009), 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 (Koren, 2008). 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 (He et al., 2016; Rendle et al., 2009; Zhang et al., 2016; Sidana et al., 2021; Moura et al., 2018). 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 reweight the impact of positive and negative feedback directly in the objective function (Pan et al., 2008) to improve the quality. Another approach is to sample data over a predefined set of interactions before learning (Liu and Wu, 2016).

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) (Donkers et al., 2017). 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 (Shani et al., 2005) and choice of an appropriate order for the model (He and McAuley, 2016). Various strategies have been proposed to leverage the limitations of Markov Models, e.g., considering only the last frequent sequences of items and using finite mixture models (Shani et al., 2005) or combining similarity-based methods with high-order

Markov Chains (He and McAuley, 2016). 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) (Moling et al., 2012; Tavakol and Brefeld, 2014). 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 (Hidasi and Karatzoglou, 2018; Tang and Wang, 2018; Kang and McAuley, 2018). 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. Fang et al. recently presented a comprehensive survey of Neural Networks based sequential approaches for personalized recommendation (Fang et al., 2020). 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. Especially, we suppose that a user may not prefer an item in absolute but may click on it if the item is shown in a given context respectively to other items that were shown but not have been clicked. For this update scheme we further propose two variants by (1) either smoothing the parameter updates with the momentum technique; or (2) controlling the number of blocks per user interaction. That is, if for a given user the number of blocks is below or above two predefined thresholds found over the distribution of the number of blocks, parameter updates for the user are discarded. We further provide a proof of convergence of both variants of the proposed approach in the general case of non-convex loss functions.

3. Framework and Problem Setting

Throughout, we use the following notation. For any positive integer n, $[n]$ denotes the set $[n] = \{1, \dots, n\}$. 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.

$\mathcal{I} = [M]$	The set of item indexes
$\mathcal{U} = [N]$	The set of user indexes
$\mathcal{D}$	joint distribution over users and items
$\mathcal{D}_u$	conditional distribution of items for a fixed user u
N_u^t	Negative items in block t for user u
Π_u^t	Positive items in block t for user u
$\mathcal{B}_u^t = \mathrm{N}_u^t \sqcup \Pi_u^t$	Negative and positive items in block t for user u
$\mathcal{I}_u^+$	The set of all positive items for user u
$\mathcal{I}_u^-$	The set of all negative items for user u
$\ell_{u,i,i'}(\omega)$	Instantaneous loss for user u and a pair of items (i, i')
$\hat{\mathcal{L}}_u(\omega)$	Empirical ranking loss with respect to user u $\hat{\mathcal{L}}_u(\omega) = \frac{1}{\lvert\mathcal{I}_u^+\rvert\lvert\mathcal{I}_u^-\rvert}\sum_{i\in\mathcal{I}_u^+}\sum_{i'\in\mathcal{I}_u^-}\ell_{u,i,i'}(\omega)$
$\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega)$	Empirical ranking loss with respect to a block of items $\hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega) = \frac{1}{\lvert\Pi_u^t\rvert\lvert\mathrm{N}_u^t\rvert}\sum_{i\in\Pi_u^t}\sum_{i'\in\mathrm{N}_u^t}\ell_{u,i,i'}(\omega)$
$\mathcal{L}(\omega)$	Expected ranking loss $\mathcal{L}(\omega) = \mathbb{E}_{\mathcal{D}_u}\hat{\mathcal{L}}_u(\omega)$

Table 1: Notation

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 $\mathcal{I}_u^- = \bigcup_t \mathrm{N}_u^t; \forall u \in \mathcal{U}$. Notation is summarized in Table 1.

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:

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

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 (Koren et al., 2009). 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 (Rendle et al., 2009). In this approach, the instantaneous loss, $\ell_{u,i,i'}$, is the surrogate regularized logistic loss for some hyperparameter $\mu \geq 0$:

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

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).

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 $t \in \{1, \ldots, B_u\}$, and, two users is shown in Figure 1.

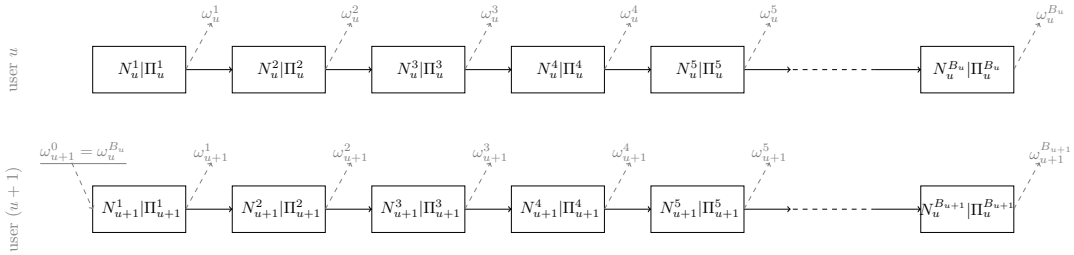


Figure 1: The horizontal axis represents the sequence of interactions over items ordered by time. Each update of weights $\omega_u^t; t \in \{1, \ldots, B_u\}$ occurs whenever the corresponding sets of negative interactions, N_u^t, and positive ones, Π_u^t, exist.

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 :

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

Note that this is different from session-based recommendations (Wang et al., 2019) 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.

We propose two strategies for the minimization of (Eq. 3) and the update of weights. In the first one, referred to as $\texttt{SAROS}_m$, the aim is to carry out an effective minimization of the ranking loss (3) by lessening the oscillations of the updates through the minimum. This is done by defining the updates as the linear combination of the gradient of the loss of (Eq. 3), $\nabla\widehat{\mathcal{L}}_{\mathcal{B}_u^t}(w_u^t)$, and the previous update as in the momentum technique at each iteration t :

$$v_u^{t+1} = \mu \cdot v_u^t + (1-\mu)\nabla\widehat{\mathcal{L}}_{\mathcal{B}_u^t}(w_u^t) \tag{4}$$

$$w_u^{t+1} = w_u^t - \alpha v_u^{t+1} \tag{5}$$

where α and μ are hyperparameters of the linear combination. In order to explicitly take into account bot attacks – in the form of excessive clicks over some target items – we propose a second variant of this strategy, referred to as $\texttt{SAROS}_b$. This variant consists in fixing two thresholds b and B over the parameter updates. For a new user u, model parameters are updated if and only if the number of blocks of items constituted for this user is within the interval $[b, B]$.

The pseudo-code of $\texttt{SAROS}_b$ is shown in Algorithm 1, 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, $\mathcal{B}_u^t = \mathrm{N}_u^t \sqcup \Pi_u^t$:

$$\omega_u^{t+1} = \omega_u^t - \frac{\eta}{|\mathrm{N}_u^t||\Pi_u^t|}\sum_{i\in\Pi_u^t}\sum_{i'\in\mathrm{N}_u^t}\nabla\ell_{u,i,i'}(\omega_u^t) \tag{6}$$

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 algorithms' convergence are 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.

Algorithm 1 SAROS$_b$: SequentiAl RecOmmender System for implicit feedback

Input: A sequence (user and items) $\{(u, (i_1, \dots, i_{|\mathcal{I}_u|})\}_{u=1}^{N}$ drawn i.i.d. from $\mathcal{D}$;
A maximal B and a minimal b number of blocks allowed per user u;
A maximal number of epochs E;
Initialization: Initialize parameters ω_1^0 randomly;
for $e = 1...E$ **do** ▷ Loop over all epochs
  for $u = 1...N$ **do** ▷ Loop over all users u
    $t \leftarrow 0$; $\mathcal{I}_u = (i_1, \dots, i_{|\mathcal{I}_u|})$ the sequence of items viewed by the user u;
    $j \leftarrow 1$; $\mathrm{N}_u^t \leftarrow \varnothing$; $\Pi_u^t \leftarrow \varnothing$;
    while $t \leq B$ and $j \leq |\mathcal{I}_u|$ **do** ▷ Loop over items displayed to user u
      while $feedback(u, i_j) = -1$ and $j \leq |\mathcal{I}_u|$ **do** ▷ While u has a negative feedback on i_j
        $\mathrm{N}_u^t \leftarrow \mathrm{N}_u^t \cup \{i\}$; $j \leftarrow j + 1$
      end while
      while $feedback(u, i_j) = +1$ and $j \leq |\mathcal{I}_u|$ **do** ▷ While u has a positive feedback on i_j
        if $\mathrm{N}_u^t \neq \varnothing$ **then** ▷ If there were negative feedback before the positive ones
          $\Pi_u^t \leftarrow \Pi_u^t \cup \{i_j\}$;
        end if
        $j \leftarrow j + 1$;
      end while
      if $\mathrm{N}_u^t \neq \varnothing$ and $\Pi_u^t \neq \varnothing$ **then** ▷ If negative and positive blocks are constituted
        $\omega_u^{t+1} \leftarrow \omega_u^t - \frac{\eta}{|\mathrm{N}_u^t||\Pi_u^t|} \sum_{i \in \Pi_u^t} \sum_{i' \in \mathrm{N}_u^t} \nabla \ell_{u,i,i'}(\omega_u^t)$;
        $t \leftarrow t + 1$; $\mathrm{N}_u^t \leftarrow \varnothing$; $\Pi_u^t \leftarrow \varnothing$;
      end if
    end while
    if $t \leq b$ **then** ▷ If the number of blocks is less than b, do not consider the updates
      $\omega_u^t \leftarrow \omega_u^0$;
    end if
    if $u < N$ **then** ▷ Initialize the weights for the next user with the current ones
      $\omega_{u+1}^0 \leftarrow \omega_u^t$;
    else ▷ The next user to the last one in the list of the users, is the first user
      $\omega_1^0 \leftarrow \omega_N^t$;
    end if
  end for
end for
Return: The last updated weights;

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 for both variants of the methods. 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 $a = 0$.

The following theorem establishes the convergence rate for the $\texttt{SAROS}_b$ algorithm.

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*

$$\min_{u:\, 1 \leq u \leq N} \mathbb{E}\|\nabla \mathcal{L}(\omega_u^0)\|_2^2 \leq \frac{2(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_u^0)) + \beta\sigma^2 \sum_{u=1}^N \sum_{t=1}^{|\mathcal{B}_u|} (\eta_u^t)^2}{\sum_{u=1}^N \sum_{t=1}^{|\mathcal{B}_u|} \eta_u^t (1 - a^2 - \beta\eta_u^t(1/2 - a^2))}. \tag{7}$$

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

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

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

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

Following (Lan, 2020); by denoting $\delta_u^t = \nabla \hat{\mathcal{L}}_{\mathcal{B}_u^t}(\omega_u^t) - \nabla \mathcal{L}(\omega_u^t)$, we have:

$$\begin{aligned}
\mathcal{L}(\omega_u^{t+1}) &\leq \mathcal{L}(\omega_u^t) - \eta_u^t \langle \nabla \mathcal{L}(\omega_u^t), \nabla \mathcal{L}(\omega_u^t) + \delta_u^t \rangle + \frac{\beta}{2}(\eta_u^t)^2 \|\nabla \mathcal{L}(\omega_u^t) + \delta_u^t\|_2^2 \\
&= \mathcal{L}(\omega_u^t) + \frac{\beta(\eta_u^t)^2}{2}\|\delta_u^t\|_2^2 - \left(\eta_u^i - \frac{\beta(\eta_u^t)^2}{2}\right)\|\nabla \mathcal{L}(\omega_u^t)\|_2^2 \\
&\qquad - \left(\eta_u^t - \beta(\eta_u^t)^2\right) \langle \nabla \mathcal{L}(\omega_u^t), \delta_u^t \rangle
\end{aligned} \tag{8}$$

Our next step is to take the expectation on both sides of inequality (8). 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 (8), it comes:

$$\begin{aligned}
\mathcal{L}(\omega_u^{t+1}) &\leq \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2 \mathbb{E}\|\delta_u^t\|_2^2 - \eta_u^t(1 - \beta\eta_u^t/2 - a^2|1 - \beta\eta_u^t|)\|\nabla \mathcal{L}(\omega_u^t)\|_2^2 \\
&\leq \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2 \|\delta_u^t\|_2^2 - \eta_u^t \underbrace{(1 - a^2 - \beta\eta_u^t(1/2 - a^2))}_{:= z_u^t} \|\nabla \mathcal{L}(\omega_u^t)\|_2^2 \\
&= \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2 \|\delta_u^t\|_2^2 - \eta_u^t z_u^t \|\nabla \mathcal{L}(\omega_u^t)\|_2^2 \\
&= \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2 \sigma^2 - \eta_u^t z_u^t \|\nabla \mathcal{L}(\omega_u^t)\|_2^2,
\end{aligned} \tag{9}$$

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

$$\sum_{u=1}^{N}\sum_{t=1}^{|\mathcal{B}_u|}\eta_u^t z_u^t \|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \leq \mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*) + \frac{\beta\sigma^2}{2}\sum_{u=1}^{N}\sum_{t=1}^{|\mathcal{B}_u|}(\eta_u^t)^2.$$

and

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

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

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

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

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

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. □

Note that the stochastic gradient descent strategy implemented in the Bayesian Personalized Ranking model (`BPR`) (Rendle et al., 2009) also converges to the minimizer of the ranking loss $\mathcal{L}(\omega)$ (Eq. 1) with the same rate.

The analysis of momentum algorithm $\texttt{SAROS}_m$is slightly more involved. We say that a function $f(x)$ satisfies the Polyak-Łojsievich condition (Polyak, 1963; Lojasiewicz, 1963; Karimi et al., 2016) if the following inequality holds for some $\mu > 0$:

$$\frac{1}{2}\|\nabla f(x)\|_2^2 \geq \mu(f(x) - f^*),$$

where f^* is a global minimum of $f(x)$.

From this definition, we can derive an analysis on the convergence of $\texttt{SAROS}_m$ stated below.

Theorem 2. *Let $\mathcal{L}(\omega)$ be a (possibly non-convex) β-smooth function which satisfies the Polyak-Lojasievich condition with a constant $\mu > 0$. Moreover, assume the number of interactions per user belongs to an interval $[b, B]$ almost surely for some positive b and B, and Assumption 1 is satisfied with some σ^2 and a. Then, for $N = \sum_{u=1}^{N} |\mathcal{B}_u|$ and a constant step-size policy $\eta_u^t = \eta$ with $\eta\beta \leq 1$, one has*

$$\mathcal{L}(\omega_u^{t+1}) - \mathcal{L}(\omega_*) \leq \exp(-\mu\eta N)(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*)) + \frac{\beta\sigma^2\eta^2}{2(1-\mu/\beta)}, \quad \eta\beta \leq 1.$$

where the estimation is uniform for all a, $0 \leq a < 1/2$.

In particular, if $\eta = c/\sqrt{N}$, under the same conditions one has

$$\mathcal{L}(\omega_u^t) - \mathcal{L}(\omega_*) \leq \exp(-\mu c\sqrt{N})(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*)) + \frac{\beta\sigma^2 c^2}{2(1-\mu/\beta)N}.$$

Proof. Similarly to the Theorem 1, From Ineq. (9) we have:

$$\mathcal{L}(\omega_u^{t+1}) \leq \mathcal{L}(\omega_u^t) + \frac{\beta}{2}(\eta_u^t)^2\sigma^2 - \eta_u^t z_u^t \|\nabla\mathcal{L}(\omega_u^t)\|_2^2$$

for $z_u^t = 1 - a^2 - \beta\eta_u^t(1/2 - a^2) > 0$. Further, using the Polyak-Lojasievich condition, it comes:

$$-\eta_u^t z_u^t \|\nabla\mathcal{L}(\omega_u^t)\|_2^2 \leq -2\mu\eta_u^t z_u^t(\mathcal{L}(\omega_u^t) - \mathcal{L}(\omega_*)),$$

and

$$\begin{aligned}
\mathcal{L}(\omega_u^{t+1}) - \mathcal{L}(\omega_*) &\leq \mathcal{L}(\omega_u^t) - \mathcal{L}(\omega_*) + \frac{\beta}{2}(\eta_u^t)^2\sigma^2 - 2\mu\eta_u^t z_u^t(\mathcal{L}(\omega_u^t) - \mathcal{L}(\omega_*)) \\
&\leq (\mathcal{L}(\omega_u^t) - \mathcal{L}(\omega_*))(1 - 2\mu\eta_u^t z_u^t) + \frac{\beta}{2}(\eta_u^t)^2\sigma^2 \\
&\leq \prod_u \prod_t (1 - 2\mu\eta_u^t z_u^t)(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*)) + \frac{\beta\sigma^2}{2}\sum_{v\leq u}(\eta_v^t)^2 \prod_v \prod_t (1 - 2\mu\eta_v^t z_v^t)
\end{aligned}$$

Finally, for a constant step-size policy, $\eta_u^t = \eta$, one has $z_u^t = z = 1 - a^2 - \beta\eta(1/2 - a^2)$ and

$$\mathcal{L}(\omega_u^{t+1}) - \mathcal{L}(\omega_*) \leq (1 - 2\mu\eta z)^N(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*)) + \frac{\beta\sigma^2\eta^2}{2(1-2\mu\eta z)},$$

where the last term is given by summing the geometric progression. As $\beta\eta \leq 1$ and $a < 1/2$ one has $z \geq 1/2$. Thus

$$\begin{aligned}
\mathcal{L}(\omega_u^{t+1}) - \mathcal{L}(\omega_*) &\leq (1 - \mu\eta)^N(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*)) + \frac{\beta\sigma^2\eta^2}{2(1-\mu/\beta)} \\
&\leq \exp(-\mu\eta N)(\mathcal{L}(\omega_1^0) - \mathcal{L}(\omega_*)) + \frac{\beta\sigma^2\eta^2}{2(1-\mu/\beta)}, \quad \eta\beta \leq 1.
\end{aligned}$$

Taking $\eta = c/\sqrt{N}$ for some positive c guarantees a rate of convergence $O(1/N)$. With a different choice of the step-size policy, rates almost up to $O(1/N^2)$ are possible; however, these rates imply $O(1/N)$ convergence for the norm of the gradient which matches the standard efforts of stochastic gradient descent under the Polyak-Lojasievich condition (Karimi et al., 2016). □

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 (Abadi et al., 2016). and computed on Skoltech CDISE HPC cluster ZHORES (Zacharov et al., 2019). 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 (Harper and Konstan, 2015) and Netflix (Bennett and Lanning, 2007b) datasets 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 (Bennett and Lanning, 2007a). For both datasets, we consider ratings greater or equal to 4 as positive feedback, and others as negative feedback.

- We get Outbrain dataset from of the Kaggle challenge (Outbrain Inc., 2016) 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(Sidana et al., 2018b) is another publicly available dataset for online recommendation (Sidana et al., 2018a) provided by Purch[1]. 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(Sidana et al., 2017a) dataset (Sidana et al., 2017b) contains 15,844,717 interactions of 2,158,859 users in Germany using Kelkoo's[2] online advertising platform.

Table 2 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. Table 3 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,

1. http://www.purch.com/
2. http://www.kelkoo.fr/

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

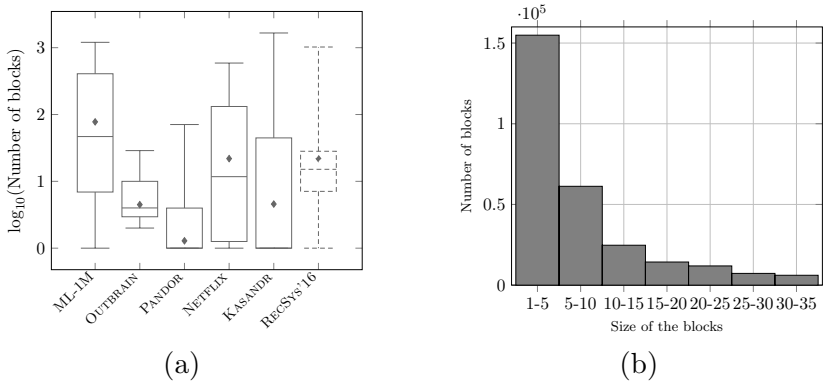
Table 2: Statistics on the # of users and items; as well as the sparsity and the average number of + (preferred) and − (non-preferred) items on ML-1M, Netflix, Outbrain, Kasandr and Pandor collections after preprocessing.

attaining 3% for the largest, Kasandr collection. Finally, we have used 10% of the most recent interactions of users in the training set as validation set for hyperparameter tuning (see below).

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 in both variants of `SAROS` positive and negative items are not sampled for updating the weights, these updates are performed on blocks of small size, and are made very often.

Dataset	$\lvert S_{train}\rvert$	$\lvert S_{test}\rvert$	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 3: Number of interactions used for train and test on each dataset, and the percentage of positive feedback among these interactions.



(a) (b)

Figure 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 learning approach described in the previous sections, we compared the proposed `SAROS` algorithm[3] 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`) (Koren, 2008), 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` (Rendle et al., 2009) 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` (Grbovic et al., 2015), learns the representation of items using a Neural Networks based model, called word2vec (Mikolov et al., 2013), and performs next-items recommendation using the similarity between the representations of items.
- `GRU4Rec+` (Hidasi and Karatzoglou, 2018) is an extended version of `GRU4Rec` (Hidasi et al., 2016) 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

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

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` (Tang and Wang, 2018) 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` (Kang and McAuley, 2018) 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` (He et al., 2020) 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. For more details regarding the exact values for the parameters, see the Table 4.

Parameter	ML	Outbrain	Pandor	Netflix	Kasandr	RecSys'16
B	78	5	2	22	5	22
b	1	2	1	1	1	1
Learning rate	.05	.05	.05	.05	.4	.3

Table 4: Hyperparameter values of $\texttt{SAROS}_b$.

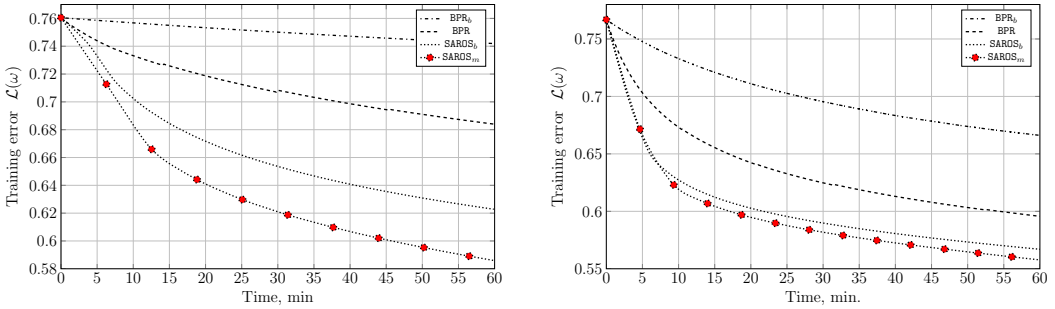
Evaluation Setting and Results. We begin our comparisons by testing $\texttt{BPR}_b$, `BPR` and `SAROS` approaches over the logistic ranking loss (Eq. 2) which is used to train them. Results on the test, after training the models till the convergence are shown in Table 5. $\texttt{BPR}_b$ (resp. `SAROS`) techniques have the worse (resp. best) test loss on all collections, and the difference between their performance is larger for bigger size datasets.

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 as it is done ine `BPR` without this sequence information of *viewed but not clicked* and *viewed and clicked* over items in user's sessions. 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 the three approaches converge to the same minimizer of the ranking loss (Eq. 1). This is an empirical evidence of the convergence of $\texttt{SAROS}_b$ and $\texttt{SAROS}_m$, showing

Dataset	Test loss at convergence, Eq. (1)			
	BPR$_b$	BPR	SAROS$_b$	SAROS$_m$
ML-1M	0.744	0.645	**0.608**	0.637
Outbrain	0.747	0.638	0.635	**0.634**
Pandor	0.694	0.661	**0.651**	0.666
Netflix	0.694	0.651	**0.614**	0.618
Kasandr	0.631	0.393	**0.212**	0.257
RecSys'16	0.761	0.644	0.640	**0.616**

Table 5: Comparison between BPR, BPR$_b$ and SAROS approaches in terms of test loss at convergence.

that the sequence of weights found by the algorithm minimizing (Eq. 3) allows to minimize the general ranking loss (Eq. 1) as it is stated in Theorems 1 and 2.



(a) Outbrain (b) Pandor

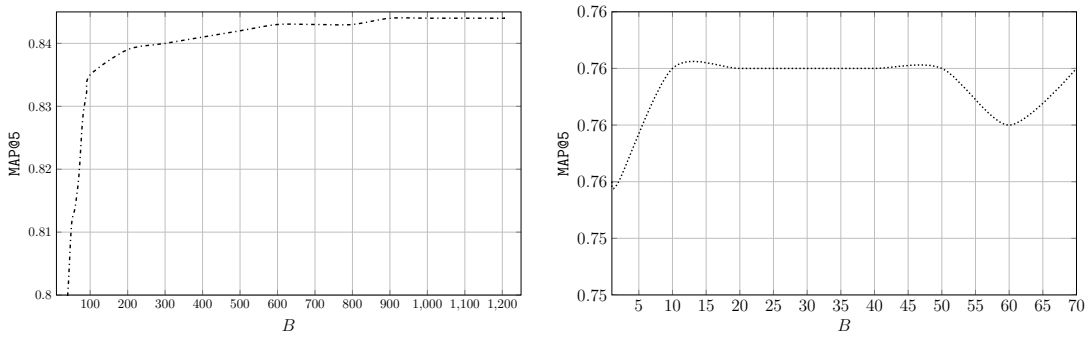
(c) Netflix (d) Kasandr

Figure 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.

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 $rel_i \in [0, 1]$ (Schutze et al., 2008).

To estimate the importance of the maximum number of blocks (B) for $\texttt{SAROS}_b$, we explore the dependency between quality metrics MAP@K and NDCG@K on ML-1M and Pandor collections (Figure 4). The latter records the clicks generated by users on one of Purch's high-tech website and it was subject to bot attacks (Sidana et al., 2018a). For this collection, large values of B affects MAP@K while the measure reaches a plateau on ML-1M. The choice of this hyperparameter may then have an impact on the results. As future work, we are investigating the modelling of bot attacks by studying the effect of long memory in the blocks of no-preferred and preferred items in small and large sessions with the aim of automatically fixing this threshold B.



(a) ML-1M (b) Pandor

Figure 4: Evolution of MAP@5 with respect to largest number of allowed blocks, B.

Table 6 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 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 (Usunier et al., 2005), 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

	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
$SAROS_m$	.763	.674	.885	.857	.735	.492	.858	.726	.899	.909	.765	.603
$SAROS_b$	**.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
$SAROS_m$	.816	.577	.720	.857	.644	.495	.787	.567	.723	.837	.651	.494
$SAROS_b$	.832	**.619**	.756	**.866**	.732	.570	**.808**	**.607**	.759	**.846**	.747	.561

Table 6: Comparison between MostPop, Prod2Vec, MF, BPR_b, BPR, GRU4Rec+, SASRec, Caser, and SAROS 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.

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. The main hypothesis of the approach is that the preferred and no-preferred items within a local sequence of user interactions express better the user preference than when considering the whole set of preferred and no-preferred items independently one from another. We presented two variants of the approach; in the first model parameters are updated 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 variant, is based on the momentum technique as a means of damping oscillations. 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 both variants of the algorithm and its minimum in the general case of non-convex ranking loss. Empirical results conducted on six 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.

Acknowledgments

This work was partly supported by MIAI@Grenoble Alpes (ANR-19-P3IA-0003); and the U.S. Department of Energy through the Los Alamos National Laboratory as part of LANL LDRD ER and DR projects. 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

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., and Isard, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

Bennett, J. and Lanning, S. (2007a). The netflix prize. In *Proceedings of KDD Cup and Workshop*.

Bennett, J. and Lanning, S. (2007b). The Netflix prize. In *The ACM SIGKDD Cup and Workshop, 2007*.

Burashnikova, A., Maximov, Y., and Amini, M.-R. (2019). Sequential Learning over Implicit Feedback for Robust Large-Scale Recommender Systems. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*.

Donkers, T., Loepp, B., and Ziegler, J. (2017). Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 152–160.

Fang, H., Zhang, D., Shu, Y., and Guo, G. (2020). Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *ACM JournalsACM Transactions on Information Systems*, 39(1).

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., and Sharp, D. (2015). E-commerce in your inbox: Product recommendations at scale. In *Proceedings of SIGKDD*, pages 1809–1818.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. In *ACM Transactions of Interaction Intelligent Systems*, pages 1–19.

He, R. and McAuley, J. (2016). Fusing similarity models with markov chains for sparse sequential recommendation. In *IEEE 16th International Conference on Data Mining (ICDM)*, pages 191–200.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. (2020). 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.

He, X., Zhang, H., Kan, M.-Y., and Chua, T.-S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 549–558.

Hidasi, B. and Karatzoglou, A. (2018). Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 843–852.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations (ICLR)*.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM)*, pages 263–272.

Kang, W. and McAuley, J. (2018). Self-attentive sequential recommendation. In *International Conference on Data Mining, ICDM*, pages 197–206.

Karimi, H., Nutini, J., and Schmidt, M. (2016). Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer.

Kaur, P. and Goel, S. (2016). Shilling attack models in recommender system. In *2016 International Conference on Inventive Computation Technologies (ICICT)*.

Koren, Y. (2008). 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.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. In *IEEE Computer Journal*, pages 30–37.

Lan, G. (2020). *First-order and Stochastic Optimization Methods for Machine Learning*. Springer.

Liu, C.-L. and Wu, X.-W. (2016). Large-scale recommender system with compact latent factor model. In *Expert Systems and Applications*, pages 467–475.

Lojasiewicz, S. (1963). *Une propriété topologique des sous-ensembles analytiques réels*. Éditions du Centre National de la Recherche Scientifique.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, (ICLR)*.

Moling, O., Baltrunas, L., and Ricci, F. (2012). Optimal radio channel recommendations with explicit and implicit feedback. In *Proceedings of the 6th ACM Conference on Recommender Systems*, pages 75–82.

Moura, S., Asarbaev, A., Amini, M.-R., and Maximov, Y. (2018). Heterogeneous dyadic multi-task learning with implicit feedback. In *International Conference on Neural Information Processing*, pages 660–672. Springer.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate o(kˆ2). *Doklady Akademii Nauk*, 269(3):543–547.

Nesterov, Y. (2018). *Lectures on convex optimization*, volume 137. Springer.

Outbrain Inc. (2016). Outbrain click prediction. Retrieved from `https://www.kaggle.com/c/outbrain-click-prediction`.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *8th IEEE International Conference on Data Mining (ICDM)*, pages 502–511.

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 325–341. Springer.

Polyak, B. T. (1963). Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 452–461.

Schutze, H., Manning, C. D., and Raghavan, P. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Shani, G., Heckerman, D., and Brafman, R. I. (2005). An MDP-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295.

Sidana, S., Laclau, C., and Amini, M.-R. (2017a). KASANDR data set. Retrieved from `https://archive.ics.uci.edu/ml/datasets/KASANDR`.

Sidana, S., Laclau, C., and Amini, M.-R. (2018a). Learning to recommend diverse items over implicit feedback on PANDOR. In *Proceedings of the* 12^{th} *ACM Conference on Recommender Systems*, pages 427–431.

Sidana, S., Laclau, C., and Amini, M.-R. (2018b). PANDOR data set. Retrieved from `https://archive.ics.uci.edu/ml/datasets/PANDOR`.

Sidana, S., Laclau, C., Amini, M.-R., Vandelle, G., and Bois-Crettez, A. (2017b). KASANDR: A Large-Scale Dataset with Implicit Feedback for Recommendation. In *Proceedings SIGIR*, pages 1245–1248.

Sidana, S., Trofimov, M., Horodnytskyi, O., Laclau, C., Maximov, Y., and Amini, M.-R. (2021). User preference and embedding learning with implicit feedback for recommender systems. *Data Mining and Knowledge Discovery*, 35:1–25.

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. In *Advances in artificial intelligence*, volume 2009, page Article ID 421425.

Tang, J. and Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 565–573.

Tavakol, M. and Brefeld, U. (2014). Factored MDPs for detecting topics of user sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 33–40.

Usunier, N., Amini, M.-R., and Gallinari, P. (2005). A data-dependent generalisation error bound for the AUC. In *ICML workshop on ROC Analysis in Machine Learning*.

Wang, C., Zhu, H., Zhu, C., Qin, C., and Xiong, H. (2020). Setrank: A setwise bayesian approach for collaborative ranking from implicit feedback. In *The 34th AAAI Conference on Artificial Intelligence*.

Wang, S., Cao, L., Wang, Y., Sheng, Q. Z., Orgun, M., and Lian, D. (2019). A survey on session-based recommender systems. *arXiv preprint*, 1902.04864.

Yi, B., Shen, X., Liu, H., Zhang, Z., Zhang, W., Liu, S., and Xiong, N. (2019). Deep matrix factorization with implicit feedback embedding for recommendation system. In *IEEE Transactions on Industrial Informatics*, pages 4591–4601.

Zacharov, I., Arslanov, R., Gunin, M., Stefonishin, D., Bykov, A., Pavlov, S., Panarin, O., Maliutin, A., Rykovanov, S., and Fedorov, M. (2019). Zhores - Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology. *Open Engineering*, 9(1):512–520.

Zhang, R., Bao, H., Sun, H., Wang, Y., and Liu, X. (2016). Recommender systems based on ranking performance optimization. In *Frontiers of Computer Science*, pages 270—280.