Peer-to-peer Online Collaborative Filtering

Andrea N. Bán
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
ban.andrea@sztaki.mta.hu

Levente Kocsis
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
kocsis@sztaki.mta.hu

Róbert Pálovics
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
palovics.robert@sztaki.mta.hu

ABSTRACT

Recommender systems often deal with a large amount of sequential data. For these scenarios, online matrix factorization techniques based on online prediction and incremental updates are often the most promising approaches. Decentralizing the system and keeping the user data on their devices is an important step in the direction of preserving user privacy. In this paper we propose a peer-to-peer online matrix factorization algorithm that stores the ratings of a user and her private data local. Additionally, the users have a local copy of the common part of the factor model and communicate with other users to advance towards a consensus on it. The algorithm is proven to converge to a set of local optima in the stationary case, while we show empirically that the algorithm performs well in the non-stationary case, both in terms of ranking performance and privacy preservation.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information filtering; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Recommender systems, Collaborative Filtering, Online learning, Peer-to-peer networks

1. INTRODUCTION

Recommender systems are now ubiquitous in most online applications. Approaches based on collaborative filtering have enjoyed significant success in several competitions, and became an area of extensive research. In the competitions it is natural to separate the collected data into a training set and a test set. However, in most real applications the data arrives continuously, and the systems need to recommend some items on the spot. While several authors argued for an online testing scenario [21, 11], it has received less attention. Incremental processing also becomes a necessity due to the size of the data. In this article we consider the scenario where users connect to the system sequentially, they request a recommendation, and the system is updated depending on the preference of the user.

Most recommender systems have a centralized structure, negating privacy requirements of the users. It is argued in [24] that it is paramount for privacy that users keep their data (e.g. ratings and user specific models) locally. Instead of storing such data on cloud servers, it is a natural idea to keep it only on personal devices. Consequently, the processing is fully distributed. This would increase the level of privacy and remove the dependence on a central infrastructure. An early approach in this direction is [20] that adapts a neighborhood algorithm to a fully distributed architecture. More recently, [28] used contextual bandit framework for social recommender systems. While these are fairly interesting approaches, we prefer matrix factorization models that have been proven perhaps the most successful approaches to collaborative filtering [18].

In this paper we consider an approach to online matrix factorization, where the user preferences and user latent vectors are kept locally, along with an instance of the item matrix. Consensus on the item matrix is reached by exchanging parts of the item matrix with other users. This way the most sensitive data stays local. We assume that revealing parts of the local copies of the item matrix will not reveal too much about the users preference. While we believe that our approach is a right step in improving the privacy of the users, and we provide an empirical evaluation against a deanonymization algorithm, the main focus of the paper is on the prediction quality of the peer-to-peer algorithm.

For non-stationary stochastic optimization, regret bounds on the online performance are provided mostly when the objective function is convex (see e.g., [6]). For batch settings, [8, 7] show that peer-to-peer stochastic approximation algorithms converge to some local optima when the complete model is sent. In a stationary online collaborative filtering framework, we extend their work on two aspects: (1) we assume that the model has a private component, and this component is not sent between peers; and (2) only parts of the common component of the model is sent at a certain moment. The first aspect is natural from privacy point of view, while the second may be beneficial for reducing the necessary communication.

The rest of the paper is organized as follows. In Section 3 we provide a skeleton for peer-to-peer online prediction algorithms. Further, we prove that the algorithm converges to a set of local optima. In this section we consider a more

general framework than matrix factorization: a model that has a private and a common component. Centralized online matrix factorization algorithms are described in Section 4 for rating prediction and for top-K recommendation. The peer-to-peer version of these matrix factorization algorithms are described in Section 5. Section 6 discusses the privacy preserving properties of these algorithms, and describes a standard de-anonymization algorithm adapted against the online peer-to-peer algorithms. The empirical performance of these algorithms in non-stationary setting is evaluated in Section 7. Conclusions and future work are provided in Section 8.

2. RELATED RESEARCH

There have been several techniques to parallelize matrix factorization algorithms, including [3, 13]. However, the cited techniques require a central component that has access to the complete model, and all the user preferences (ratings). This drawback is circumvented to some respect in [16] by assigning computational nodes to both users and items, with the items having access to the private data of users that interact with. In their setting the space of items overlaps with that of users, and if items are trusted users, this approach is sufficient. However, in most cases items are handled by one or a few service providers, and the privacy would be compromised using this technique. An approach very similar to ours was proposed in [15] for singular value decomposition (SVD). Each row of the matrix is assigned to a user, along with the corresponding user vectors, and the item matrices are communicated between the items in order to achieve consensus. Our approach is different in that we focus on online ranking prediction (as opposed to SVD on a fixed data set), which enables a more selective broadcasting of item vectors. A theoretical analysis of such selective communication strategies is also provided.

As mentioned before, we are interested in (non-stationary) online prediction with matrix factorization, which has non-convex objective functions. There have been very few papers with theoretical guarantees for non-stationary online optimization for non-convex functions. The few exceptions, such as [22], use some form of a global search, which do not scale very well with the dimensionality of the problem (factor models have relatively high dimensions). For non-stationary stochastic optimization, there exists regret bounds on the online performance when the objective function is convex (see e.g., [6]). Regret bounds were proven for distributed optimization of stationary convex functions as well [9, 12, 29]. Most of these approaches have a periodical consensus step to ensure that the local models are identical. Of these, [9] is perhaps the most interesting on using variance based dynamic communication. Unfortunately, the objective functions are not convex in the parameters of a matrix factorization model, and therefore, these results can not be applied. There have been several papers on optimization of non-convex functions in batch settings, including peer-to-peer stochastic approximation algorithms. The papers that have the mildest assumptions on the communication protocol include [7, 8], which are also the closest to our analysis. They show that the approximation algorithms converge to some local optima when the complete model is sent. We extend their work by facilitating a private part of the model (the user vectors for matrix factorization), and allowing the algorithms to send only a part of the common model (e.g., only the vector corresponding to the rated item).

One of the aims for peer-to-peer online collaborative filtering is improving the privacy of the users. The vulnerability of centrally collected data was also shown by [23]. There have been several attempts to improve privacy of nearest neighborhood (e.g., [20, 10]) or contextual bandit approaches [28], but we choose to focus on matrix factorization. [24] suggests extending the server with a crypto-service provider, which imposes restrictions on the collaborative filtering algorithm, and still requires some trust in the server. An algorithm designed to improve privacy of collaborative filtering data against attacks described in [23] is the k-CoRating algorithm [31]. It extends the rating matrix such that each user has at least $k-1$ peers that rated exactly the same items. The cost of the improved privacy can be a deterioration of the collaborative filtering algorithm that uses that extended data (as shown in our experiments). The peer-to-peer computational architecture adopted by our approach improves privacy without harming prediction performance as shown [30] for convex optimization, and as illustrated by our experiments described in Section 7.3. However, further techniques can be applied to harden the privacy requirements.

3. CONVERGENCE

In this section we show that peer-to-peer online prediction algorithms converge to the same set of local optima as their corresponding centralized algorithms. We consider a more general framework than matrix factorization: a model that has a private and a common component.

Assume that we have N users. At every time step n, user u connects to its local recommender system (RS), and requests some prediction that can be either a rating of an item or a list of top items. The local RS observes some context that includes the item itself in case of rating prediction, as well as any other relevant information. After receiving the prediction the user reveals its preference (i.e. an item and/or the rating of an item). The triplet of user activity, context and user preference will be identified by the random variables X_n, which are supposed to be independent and identically distributed (i.i.d.). X_n may represent the activity of more than one user, if several users are active at the same time.

We consider RS's represented by a model consisting of a user specific vector $\varphi^u \in \mathbb{R}^{d_2}$, and a common part $\theta \in \mathbb{R}^{d_1}$. For instance, in the case of matrix factorization, the former is represented by the user vectors, and the latter by the item vectors. In a peer-to-peer variant of the RS, the local RS will store the vector specific to its user φ^u and a local instance, θ^u, of the common vector.

The performance of the RS is evaluated by a loss function that can be decomposed into some local functions $f^u(\theta^u, \varphi^u, X)$. The loss functions of inactive users are assumed 0. The function depends only on the local component of X (the local context, activity and user preference). The aim is to minimize the global loss function

$$f^*(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}\boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X),$$

where $\boldsymbol{\theta} = (\theta^{1T}, \ldots, \theta^{NT})^T$, $\boldsymbol{\varphi} = (\varphi^{1T}, \ldots, \varphi^{NT})^T$ and $\boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) = \sum_{u=1}^{N} f^u(\theta^u, \varphi^u, X)$.

In order to minimize the loss function, after observing the user preference, for each user the local RS will update its parameters in the direction of the negative gradient, as

follows:

$$\begin{aligned}\tilde{\theta}_n^u &= \theta_{n-1}^u - \gamma_n \nabla_\theta f^u(\theta_{n-1}^u, \varphi_{n-1}^u, X_n) \\ \varphi_n^u &= \varphi_{n-1}^u - \gamma_n \nabla_\varphi f^u(\theta_{n-1}^u, \varphi_{n-1}^u, X_n).\end{aligned} \quad (1)$$

Note that for inactive users the local gradients are zero.

To obtain generalization, the local gradient steps are followed by a communication step aimed at improving the consensus on the common part of the model. The communication among users can be represented by a sequence of i.i.d. random matrices $W_n \in \mathbb{R}^{Nd_1 Nd_1}$, and using the notation $\tilde{\boldsymbol{\theta}}_n = (\tilde{\theta}_n^{1T}, \dots, \tilde{\theta}_n^{NT})^T$ we update $\boldsymbol{\theta}_n$ as follows:

$$\boldsymbol{\theta}_n = W_n \tilde{\boldsymbol{\theta}}_n. \quad (2)$$

In the following, first we prove that every θ^u converges to the same limit θ^*. Then we prove that the pair consisting of θ^* and the limit of $\boldsymbol{\varphi}_n$ is a local minimum of $f^*(\theta, \boldsymbol{\varphi})$. Let us introduce some further notations: $\mathbf{1} = (1, 1, \dots, 1)^T \in \mathbb{R}^N$. For $\boldsymbol{x} \in \mathbb{R}^{Nd}$ we write $\langle \boldsymbol{x} \rangle = 1/N(x^1 + \dots + x^N)$ so that

$$\langle \boldsymbol{x} \rangle = \frac{1}{N}(\mathbf{1} \otimes I_d)\boldsymbol{x}, \quad (3)$$

where $\otimes$ denotes the Kronecker product. Further, J stands for the projection onto the 'consensus space':

$$J := (\mathbf{1}\mathbf{1}^T/N) \otimes I_d,$$

whence $J\boldsymbol{x} = \mathbf{1} \otimes \langle \boldsymbol{x} \rangle$. Let us denote by $J_\perp := I_{dN} - J$ the matrix of the orthogonal projection to the consensus space (here $I_{d'}$ denotes the identity matrix of dimension d'). Finally $\boldsymbol{\theta}_{\perp,n} := J_\perp \boldsymbol{\theta}_n$.

Here, we make three assumptions: (1) the communication matrix is such that ensures an averaging process for $\boldsymbol{\theta}_n$, (2) standard conditions on γ_n that enable the stochastic approximation to converge, and (3) the objective function is 'sufficiently nice' for the stochastic approximation.

ASSUMPTION 1. *Let W_n be a sequence of i.i.d. random matrices of size $Nd_1 \times Nd_1$ with non-negative elements. The following conditions hold true:*

- *W_n is row-stochastic for every n: $W_n\mathbf{1} = \mathbf{1}$,*
- *$\mathbb{E}(W_n)$ is column-stochastic for every n: $\mathbf{1}^T\mathbb{E}(W_n) = \mathbf{1}^T$,*
- *the spectral norm ρ of $\mathbb{E}((W_n)^T(I-J)W_n)$ is less than one.*

ASSUMPTION 2. *Let γ_n be a monotone non-increasing sequence of positive numbers that satisfies the following conditions:*

- *$\sum_n \gamma_n^2 < \infty$,*
- *$\sum_n \gamma_n = \infty$, and*
- *$\gamma_n/\gamma_{n+1} \to 1$.*

ASSUMPTION 3. *For any $u = 1, \dots, N$, $f^u(\theta, \varphi, X)$ satisfies the next assumptions:*

- *$f^u(\theta, \varphi, X)$ is continuously differentiable with respect to θ and φ,*
- *$f^u(\theta, \varphi, X)$ is bounded from below and*
- *for all realizations of θ, φ and X the gradient $\nabla_\theta f^u(\theta, \varphi, X)$ satisfies*

$$\sup_\theta \mathbb{E}[|\nabla_\theta f^u(\theta, \varphi, X)|^2] < \infty.$$

THEOREM 1. *If the matrices W_n satisfy Assumption 1, the sequence γ_n satisfies Assumption 2 and the function $f^u(\theta, \varphi, X)$ satisfies Assumption 3, then the variables $\boldsymbol{\theta}_n$ in (2) achieve consensus, that is $\sum_n \mathbb{E}[|\boldsymbol{\theta}_{\perp,n}|^2] < \infty$ and $\boldsymbol{\theta}_{\perp,n} \to 0$ almost surely as $n \to \infty$.*

PROOF. The proof is based on the proof of Lemma 1 in [8]. To shorten the notation we use $\mathbf{Y}_n = -\nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_n, \boldsymbol{\varphi}_n, X_n)$, so we have $\boldsymbol{\theta}_n = W_n(\boldsymbol{\theta}_{n-1} + \gamma_n \mathbf{Y}_n)$. Using the row-stochasticity of W_n it is easy to check that $J_\perp W_n J_\perp = J_\perp W_n$. Thus (2) can be written as $\boldsymbol{\theta}_{\perp,n} = J_\perp W_n(\boldsymbol{\theta}_{\perp,n-1} + \gamma_n \mathbf{Y}_n)$. Whence we conclude the estimation

$$\begin{aligned}|\boldsymbol{\theta}_{\perp,n}|^2 &= (\boldsymbol{\theta}_{\perp,n-1} + \gamma_n \mathbf{Y}_n)^T W_n^T J_\perp W_n(\boldsymbol{\theta}_{\perp,n-1} + \gamma_n \mathbf{Y}_n) \\ &\le \lambda_1(W_n^T J_\perp W_n)|\boldsymbol{\theta}_{\perp,n-1} + \gamma_n \mathbf{Y}_n|^2, \end{aligned} \quad (4)$$

where $\lambda_1(M)$ is the largest eigenvalue of the matrix M. Taking the expectation of (4) and using the Cauchy-Schwarz inequality we obtain

$$\begin{aligned}\mathbb{E}[|\boldsymbol{\theta}_{\perp,n}|^2] &\le \rho(W_n^T J_\perp W_n)\mathbb{E}[|\boldsymbol{\theta}_{\perp,n-1}|^2] \\ &+ 2\gamma_n \mathbb{E}[|\mathbf{Y}_n|]\sqrt{\mathbb{E}[|\boldsymbol{\theta}_{\perp,n-1}|^2]} + \gamma_n^2 \mathbb{E}[\mathbf{Y}_n^2].\end{aligned}$$

In order to conclude $\sum_n \mathbb{E}[|\boldsymbol{\theta}_{\perp,n}|^2] < \infty$, we need an additional lemma.

LEMMA 1. *Assume that for the sequence $\gamma_n > 0$, $\gamma_n/\gamma_{n+1} \to 1$, $\sum_n \gamma_n = \infty$, $\sum_n \gamma_n^2 < \infty$ and $0 < \rho < 1$. Let $v_n > 0$ be such that $v_n \le \rho v_{n-1} + \sqrt{v_{n-1}}\gamma_n + \gamma_n^2$. Then $\sum_{n=0}^\infty v_n < \infty$.*

A statement similar to Lemma 1 (with slightly different assumptions) is proved in [8], and we omit the proof for space limitations. The assertion of Theorem 1 then follows from the Borel-Cantelli lemma. □

In Theorem 2 we prove that the algorithm converges to a local minimum. The proof of the convergence is based on the proof of Theorem 1 in [7]. We will need some additional assumptions and a lemma.

ASSUMPTION 4. *There are two finite constants C_1 and C_2 such that for all realizations of $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$, $\boldsymbol{\varphi}$ and X the next assumptions are satisfied:*

- *$|\langle \nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}, X) \rangle - \langle \nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_2, \boldsymbol{\varphi}, X) \rangle| < C_1|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2|$,*
- *$|\nabla_\varphi \boldsymbol{f}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}, X) - \nabla_\varphi \boldsymbol{f}(\boldsymbol{\theta}_2, \boldsymbol{\varphi}, X)| < C_2|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2|$.*

LEMMA 2. *If the Assumptions 1, 2, 3 and 4 are satisfied then there exist some random vectors $\zeta_{n,\theta} \in \mathbb{R}^{d_1}$ and $\zeta_{n,\varphi} \in \mathbb{R}^{Nd_2}$ which satisfy*

$$\limsup_{k\to\infty} \sup_{l \ge k} |\sum_{i=k}^{l} \gamma_i \zeta_{i,\theta}| = 0 \text{ and} \quad (5)$$

$$\limsup_{k\to\infty} \sup_{l \ge k} |\sum_{i=k}^{l} \gamma_i \zeta_{i,\varphi}| = 0, \quad (6)$$

almost surely, such that

$$\langle \boldsymbol{\theta}_n \rangle = \langle \boldsymbol{\theta}_{n-1} \rangle - \gamma_n \left\langle \nabla_\theta \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_{n-1} \rangle, \boldsymbol{\varphi}_{n-1}, X_n) \right\rangle + \gamma_n \zeta_{n,\theta}, \quad (7)$$

$$\boldsymbol{\varphi}_n = \boldsymbol{\varphi}_{n-1} - \gamma_n \nabla_\varphi \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_n \rangle, \boldsymbol{\varphi}_{n-1}, X_n) + \gamma_n \zeta_{n,\varphi}. \quad (8)$$

PROOF. Rearranging terms, we obtain $\zeta_{n,\varphi} = \nabla_\varphi \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_n \rangle, \boldsymbol{\varphi}_{n-1}, X_n) - \nabla_\varphi \boldsymbol{f}(\boldsymbol{\theta}_n, \boldsymbol{\varphi}_{n-1}, X_n)$. Using Assumption 4 and the inequality $2ab \leq a^2 + b^2$, we get

$$\begin{aligned} |\sum_{i=k}^{l} \gamma_i \zeta_{i,\varphi}| &\leq \frac{1}{2}\sum_{i=k}^{l} |\gamma_i|^2 + \frac{1}{2}\sum_{i=k}^{l} |\zeta_{i,\varphi}|^2 \\ &\leq \frac{1}{2}\sum_{i=k}^{l} |\gamma_i|^2 + \frac{C_2^2}{2}\sum_{i=k}^{l} |\boldsymbol{\theta}_{\perp,i}|^2. \end{aligned}$$

Now (6) follows from Assumption 2 and Theorem 1.
The argument proving (5) is similar to the one in Theorem 1 of [7]. The row-stochastic property of W_n implies $W_n J = J$. Using (1), (2) and (3) we obtain

$$\langle \boldsymbol{\theta}_n \rangle = \langle \boldsymbol{\theta}_{n-1} \rangle - \gamma_n \langle Z_n \rangle$$

where $Z_n = W_n \left(\nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\varphi}_{n-1}, X_n) + \gamma_n^{-1} \boldsymbol{\theta}_{\perp,n-1} \right)$. Rearranging terms, we get

$$\begin{aligned} \langle \boldsymbol{\theta}_n \rangle = \langle \boldsymbol{\theta}_{n-1} \rangle &- \gamma_n \left\langle \nabla_\theta \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_{n-1} \rangle, \boldsymbol{\varphi}_{n-1}, X_n) \right\rangle \\ &+ \gamma_n e_{n,\theta} + \gamma_n \xi_{n,\theta} \end{aligned}$$

where $e_{n,\theta}$ and $\xi_{n,\theta}$ are defined as

$$\begin{aligned} e_{n,\theta} &= \langle \nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\varphi}_{n-1}, X_n) \rangle \\ &\quad - \langle W_n(-\nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\varphi}_{n-1}, X_n) + \gamma_n^{-1} \boldsymbol{\theta}_{\perp,n-1}) \rangle \\ \xi_{n,\theta} &= \langle \nabla_\theta \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_{n-1} \rangle, \boldsymbol{\varphi}_{n-1}, X_n) \rangle \\ &\quad - \langle \nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\varphi}_{n-1}, X_n) \rangle. \end{aligned}$$

We can prove $\limsup_{k\to\infty} \sup_{l \geq k} |\sum_{i=k}^{l} \gamma_i \xi_{i,\theta}| = 0$ in the same way as we proved the corresponding statement for $\zeta_{i,\varphi}$. On the other hand, a direct computation shows that $\gamma_i e_{i,\theta}$ is a martingale difference sequence. Furthermore, the corresponding martingale is bounded in L^2, hence converges with probability probability 1 (see e.g. Corollary 2.2 in [14]). That is, $\sum_{k=1}^{\infty} \gamma_k e_{k,\theta}$ exists and is finite almost surely. The lemma follows. □

Let us define

$$\mathcal{L} = \{(\theta, \boldsymbol{\varphi}) : \nabla f^*(\theta, \boldsymbol{\varphi}) = 0\}.$$

In order to guarantee the convergence (i.e. Theorem 2), we need some further criteria:

ASSUMPTION 5.
- *There exists $M_0 > 0$ such that $\mathcal{L} \subset \{(\theta, \boldsymbol{\varphi}) : f^*(\theta, \boldsymbol{\varphi}) < M_0\}$.*
- *There exists $M_1 \in (M_0, \infty)$ such that $\{(\theta, \boldsymbol{\varphi}) : f^*(\theta, \boldsymbol{\varphi}) < M_1\}$ is a compact set.*
- *The interior of the set $f^*(\mathcal{L})$ is empty.*

THEOREM 2. *If the Assumptions 1, 2, 3, 4, 5 are satisfied and θ_0 and $\boldsymbol{\varphi}_0$ are such that $f^*(\theta_0, \boldsymbol{\varphi}_0) < M_0$ then $\boldsymbol{\theta}_n$ and $\boldsymbol{\varphi}_n$ converge with probability 1, moreover*

$$\lim_{n\to\infty} \inf\{|(\langle \boldsymbol{\theta}_n \rangle, \boldsymbol{\varphi}_n) - (\theta, \boldsymbol{\varphi})|, (\theta, \boldsymbol{\varphi}) \in \mathcal{L}\} = 0.$$

PROOF. Note that $\boldsymbol{\theta}_n = \mathbf{1} \otimes \langle \boldsymbol{\theta}_n \rangle + \boldsymbol{\theta}_{\perp,n}$. Since in Theorem 1 we proved that $\boldsymbol{\theta}_{\perp,n} \to 0$ almost surely as $n \to \infty$, we only need to examine the convergence of $\langle \boldsymbol{\theta}_n \rangle$. Lemma 2 says

$$\langle \boldsymbol{\theta}_n \rangle = \langle \boldsymbol{\theta}_{n-1} \rangle - \gamma_n \left\langle \nabla_\theta \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_{n-1} \rangle, \boldsymbol{\varphi}_{n-1}, X_n) \right\rangle + \gamma_n \zeta_{n,\theta}$$

$$\boldsymbol{\varphi}_n = \boldsymbol{\varphi}_{n-1} - \gamma_n \nabla_\varphi \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_n \rangle, \boldsymbol{\varphi}_{n-1}, X_n) + \gamma_n \zeta_{n,\varphi}.$$

So we can use Theorem 2.2 and 2.3 from [4] for both $\langle \boldsymbol{\theta}_n \rangle$ and $\boldsymbol{\varphi}_n$ (note that we need Lemma 2 to verify certain assumptions of the cited theorems). The statement of the Theorem 2 follows. □

In Section 5, we will instantiate the peer-to-peer algorithm discussed above to matrix factorization. In the following, we will consider whether the assumption hold for this instance. An example of a communication protocol when the complete model (θ) is sent and satisfies Assumption 1 is provided in [5]. For a matrix factorization model it may be preferable not to send the latent vectors corresponding to all items but just a small subset of the items. The strategies to select the items is discussed in Section 5. The conditions on the learning rate (Assumption 2) are fairly standard for a stochastic optimization algorithm in stationary setting. If the environment is non-stationary (as it is in the experiments), it is standard to switch to a constant learning rate. It is clear that if the loss function is the mean square error, then the function is continuously differentiable with respect to the parameters of the matrix factorization model and is bounded from below. The boundedness of the gradient in Assumption 3 and Assumption 4 hold if the parameters of the model stay bounded in the trajectories starting from an appropriately chosen starting point. This is not necessarily the case for any dataset, but in our experience it seems that the parameters stay bounded unless the initial learning rate is too high. It is clear that if the parameters diverge, they are likely to do so for the centralized algorithm as well, and the parameters needs to be projected into a compact set after the gradient step. In this case, proving Theorem 2 takes a different course, one that mirrors the proof of Theorem 1 of [8]. Assumption 5 also holds, for instance, the condition on the empty interior is satisfied by Sard's theorem if the loss function is Nd_1d_2-times differentiable, which is the case for a matrix factorization model.

4. ONLINE MATRIX FACTORIZATION

In this section we consider the online algorithms for rating prediction and top-K recommendation that form the base of the peer-to-peer algorithms presented in Section 5.

Matrix factorization algorithms constitute the most successful approaches to collaborative filtering [18, 19]. In these algorithms, users and items are mapped into a joint latent factor space of dimensionality d. Accordingly, each user u is associated with a d-dimensional vector U_u, and each item i with a d-dimensional vector V_i. The preference of the user for the specific item is given by the inner product of the two vectors. Additionally, it is usual to have biases specific to users b_u and items b_i. The predicted preference of a user u for item i, $\hat{r}_{ui}$ is given by the following formula,

$$\hat{r}_{ui} = b_u + b_i + U_u^T V_i. \tag{9}$$

There are several ways to tune factor models. In an online scenario, the most natural is the stochastic gradient descent (SGD) [1]. SGD have been a popular choice for matrix factorization algorithms and in online prediction. The online rating prediction algorithm for matrix factorization is presented in Algorithm 1.

At each time step a user connects to the recommender system and selects an item (line 2). The system predicts the rating of the item (line 3), after which the user reveals the 'true' rating and the recommender system suffers a loss

Algorithm 1 Online rating prediction

1: **for** $n = 1$ to T **do**
2: user u connects to RS and selects item i
3: RS predicts rating $\hat{r}_{ui}$
4: user reveals r_{ui} and RS suffers loss $f(r_{ui}, \hat{r}_u i)$
5: $U_u \leftarrow U_u + \gamma(r_{ui} - \hat{r}_{ui})V_i$
6: $b_u \leftarrow b_u + \gamma(r_{ui} - \hat{r}_{ui})$
7: $V_i \leftarrow V_i + \gamma(r_{ui} - \hat{r}_{ui})U_u$
8: $b_i \leftarrow b_i + \gamma(r_{ui} - \hat{r}_{ui})$

Algorithm 2 Online ranking prediction

1: **for** $n = 1$ to T **do**
2: user u connects to RS
3: RS recommends top items $\mathcal{R}$
4: user selects item i (and additionally rating r_{ui})
5: RS suffers loss $f'(i, \mathcal{R})$
6: select negative sample set $\mathcal{N}$
7: $U_u \leftarrow U_u + \gamma(r_{ui} - \hat{r}_{ui})V_i - \gamma \sum_{j \in \mathcal{N}} \hat{r}_{uj} V_j$
8: $b_u \leftarrow b_u + \gamma_t(r_{ui} - \hat{r}_{ui}) - \gamma \sum_{j \in \mathcal{N}} \hat{r}_{uj}$
9: $V_i \leftarrow V_i + \gamma(r_{ui} - \hat{r}_{ui})U_u$
10: $b_i \leftarrow b_i + \gamma(r_{ui} - \hat{r}_{ui})$
11: **for** j in $\mathcal{N}$ **do**
12: $V_j \leftarrow V_j - \gamma \hat{r}_{uj} U_u$
13: $b_j \leftarrow b_j - \gamma \hat{r}_{uj}$

(line 4). The loss function, in our case, is the mean square error between the predicted and true rating. The model parameters are then updated in the direction of the negative gradient with a constant learning rate (line 5–8). In Section 3 we assumed that the learning rate follows a decaying schedule, however, for non-stationary environments, a constant rate is more appropriate.

While rating prediction has a larger literature due to some of the competitions, recommending a list of top items is a more natural task in real applications. Continuing with the SGD set-up, we use the algorithm suggested in [25] that uses the visited item as positive training instance and some randomly sampled unvisited items as negative instances. As shown in Algorithm 2, at each time step, when a user connects to the recommender system, the system ranks the items, and presents the user the top of this list. The items are ranked according to the predicted rating, using equation (9). The system then suffers a loss. There are several choices of ranking measures, a popular one that is used in the experiments is NDCG@K [17]. In our case, there is only one item with non-zero label, and the NDCG@K of a permutation π of the items reduces to

$$\text{NDCG@K}(\pi) = \begin{cases} 1/\log_2(rank_\pi(i) + 1) & \text{if } rank_\pi(i) \leq K \\ 0 & \text{otherwise} \end{cases},$$

where i denotes the visited item, and $rank_\pi(i)$ is the position of the item in the permutation π. In the experiments, for the selection of the negative training instances (line 6 of Algorithm 2), we also consider a mechanism suggested in [32], namely sample randomly a number of unvisited items, and then select only a few items from the top of the sampled list (ranked by the model). The update of the model parameters in Algorithm 2 is shown in lines 7–13.

5. ONLINE PEER-TO-PEER PREDICTION

In Section 3, we presented the framework for a peer-to-peer online prediction algorithm. For the special case of a recommender system using matrix factorization, each user has at hand a local copy of the system, the local model including the latent vector of the user, U_u, the user bias, b_u, a local instance of the item matrix, V^u, and a local instance of the item biases, $b_\cdot^u$. Thus, the user latent vectors and the user biases from Section 4 correspond to φ from Section 3, and the item vectors and biases correspond to θ. The peer-to-peer online recommender is shown in Algorithm 3. At each time step, when a user connects to its local system (line 7), the system makes a prediction. The prediction in the case of rating prediction is estimating the rating of an item, while in the case of recommendation, a list of items. After the prediction, the user reveals its preference, and the system suffers a loss. The preference of the user is the true rating in the case of rating prediction, and a selected item for top-K recommendation. For the latter, the item may or may not be in the recommended list. These steps, shown succinctly in line 8, are identical to the corresponding steps of Algorithm 1 or Algorithm 2. The gradient descent update of the local model (line 9) is also identical to the corresponding centralized variants (including the sampling of negative instances for top-K recommendation). In line 10 the algorithm selects a set of target users and a set of items. Subsequently, the local instance of the latent vectors and biases corresponding to the selected items are sent to the selected set of target users.[1] In a simple variant, a fixed number of users are selected randomly, but more intricate strategies are also possible. A more elaborated method for setting the number of target users is described in Section 7, while users with similar taste could also be given preference. When a social network of users is available, selecting friends as target users seems reasonable for privacy reasons. It is also an easy way to identify users with similar taste. In the case of rating prediction, the natural choice for the item set to be sent is the rated item. While for the top-K recommendation task, the item set should include the item visited by the user and some subsample of the negative set of instances. When the target users receive some vectors, they combine the vectors with their local copies, using a mixing coefficient β (line 11–13).

The conditions for the peer-to-peer algorithm to mirror the centralized algorithm are as follows: (1) the set of target users should include all users, (2) the set of items sent consists of the rated one and, in the case of top-K recommendation, all negative samples, (3) $\beta = 0$, and (4) appropriately chosen initialization for the latent vectors. The last condition requires that the same latent user or item vectors are generated when first encountered in the centralized or peer-to-peer algorithms. If the first three conditions are met, then the item vectors are the same at each user, disregarding items that have not been updated by any user.

So far, we assumed that all users are present in the system from the start of the protocol until the end of it. A more realistic setting is when new users are coming continually in the system. For this setting, Algorithm 3 includes a set-up phase (line 2–6): the new user requests a copy of item vectors

[1]When it is clear from the context, for brevity, we will say sending items instead of sending the corresponding item vectors and biases.

Algorithm 3 Online peer-to-peer prediction

for $n = 1$ to T **do**
  for each new user u' **do**
    select sources $\mathcal{U}'$
    for v in $\mathcal{U}'$ **do**
      $V^{(v)} \leftarrow \frac{1}{|\mathcal{U}'|} \sum_{v \in \mathcal{U}'} V^{(v)}$
      $b^{(v)} \leftarrow \frac{1}{|\mathcal{U}'|} \sum_{v \in \mathcal{U}'} b^{(v)}$
  user u connects to local RS
  local RS predicts rating and suffers loss
  update U, b_u, $V^{(u)}$, $b_i^{(u)}$
  select target user set $\mathcal{U}$ and item set $\mathcal{I}$
  for v in $\mathcal{U}$ and j in $\mathcal{I}$ **do**
    $V_j^{(v)} \leftarrow \beta V_j^{(v)} + (1-\beta) V_j^{(u)}$
    $b_j^{(v)} \leftarrow \beta b_j^{(v)} + (1-\beta) b_j^{(u)}$

and biases from a random set of users already present in the system, and then, averages these copies. When the local copies of item vectors and biases are identical, it is enough to request from only one user. When, due to a limit on the amount of communication, this is not the case, it may be beneficial to request from more users and average their instances. This issue will be revisited in the experimental section.

6. PRIVACY

One of the advantages of the peer-to-peer architecture for recommender systems is that user data is kept locally, preventing an unsolicited party to access it. While the user data cannot be accessed externally, the communication may still reveal some private information.

A similar peer-to-peer algorithm was considered by [30] for online convex optimization, and the authors have shown that such an algorithm has intrinsic privacy-preserving properties, i.e. the gradients of the local function cannot be reconstructed for various topologies of user communication graphs. While in their case, the full model was transmitted, for matrix factorization the user vector are kept locally, and only part of the item vectors are sent at any time. It is easy to see that the inaccessibility of the user vectors and the non-convexity of the function to be optimized makes it even more difficult to reconstruct the gradients.

While the user data is kept locally, and it is sufficiently difficult to reconstruct the gradients, the communication may still reveal information about which items are rated by the user. This vulnerability comes in because the users are sending the vectors corresponding to a restricted subset of the items that contains the rated items. A natural choice for 'attacking' the data made available to a malicious party is the ScoreBoard algorithm of [23]. The algorithm can handle well noise both in the prior information, and the information received from the communication. It assumes that the malicious party gained some auxiliary information about the preferences of a particular person (such as ratings available on IMDB, or preferences stated in a discussion), and attempts to identify the person in the network from the received communications. Subsequently, the algorithm deciphers additional information in the form of preferences for various items.

The ScoreBoard algorithm maintains a score s_u for each user, reflecting how likely is that the information about the particular user matches the auxiliary information. The auxiliary information is represented by the set of items Aux. The algorithm relies on a similarity measure at attribute (item) level. The similarities of the items in the auxiliary set, weighted with a function that depends on their respective frequencies, are combined in the scoring function:

$$s_u = \sum_{i \in Aux} \frac{1}{\log |n_i + 1|} \mathbb{I}_{(r_{ui} > \delta)},$$

where n_u if number of ratings for item i, r_{ui} is the ratio of item i received in the messages received from user u, and δ is appropriate threshold.

7. EXPERIMENTS

In this section we will test empirically the performance of the peer-to-peer online prediction algorithms using two standard benchmarks: the 1M and the 10M MovieLens datasets.[2] The datasets include movie ratings with a timestamp recorded in seconds. We use the timestamps to establish a sequential order of the ratings. Ratings with identical timestamps are considered in random order. The datasets stretch over several years thus there is a fair amount of non-stationarity. The learning rate for all experiments is kept constant 0.05, which was found to perform well for the centralized prediction variants, and kept the same for the distributed ones. The dimensionality of the latent vectors is set to 10.

Since we consider the top-K recommendation task the more interesting one, we focus most of our attention on this task. There is, however, a feature that favours rating prediction: in this task the loss function and the gradient step are closely tied. In contrast, in top-K recommendation the gradient is computed for a loss function that is only related to the true objective measure. Since the convergence results of Section 3 refer to the loss function of which gradient is taken, the first set of experiments (described in Section 7.1) can be thought as a bridge between the theoretical results and the experiments on the top-K recommendation task (Section 7.2). Finally, we describe experiments on privacy in Section 7.3.

7.1 Rating prediction

We consider a 'fixed' communication protocol where the number of target users is kept the same at each time step. Recall from Section 5 that in the case of rating prediction, only the vector and bias corresponding to the rated item are sent in each iteration. The average mean square error (MSE) for the two datasets are shown in Figure 1 and Figure 2. Each data point corresponds to a run where the number of selected targets is kept at a certain value. In both figures the data points with most messages correspond to the situation when the vectors are sent to all users and the loss obtained is the same as the one resulting from a centralized algorithm. We observe that the loss increases logarithmically with decreasing the number of messages sent (target users selected). In the figures two simple dynamic schedules that depend on the suffered loss (MSE) are included as well: namely the number of target users selected equals $(N-1)/(1+c/f_n)$, where f_n is the loss at time step n, and c is constant that is varied in the figure. One can notice that for both datasets a slight improvement can be obtained with the dynamic communication schedule.

[2]http://grouplens.org/datasets/movielens/

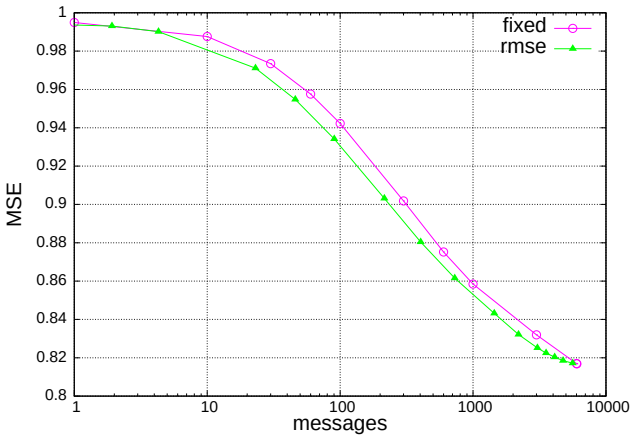

Figure 1: Peer-to-peer rating prediction on the 1M MovieLens dataset. The x-axis shows the average number of item vectors sent per time step.

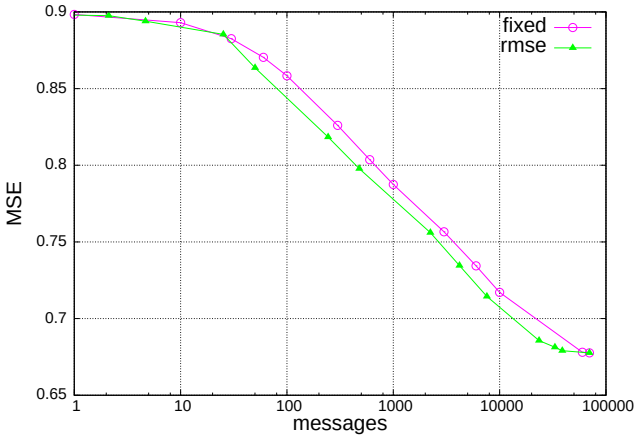

Figure 2: Peer-to-peer rating prediction on the 10M MovieLens dataset.

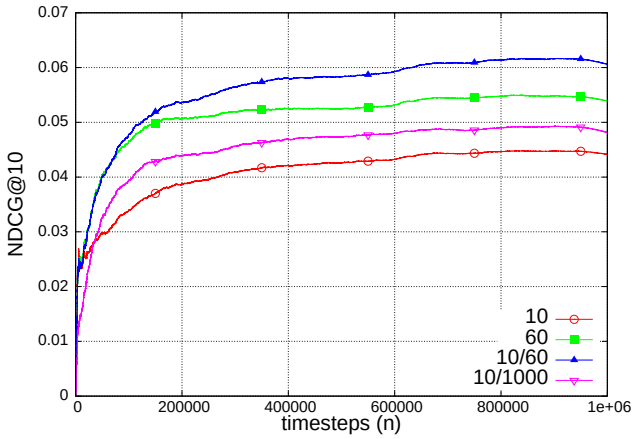

Figure 3: Centralized top-K recommendation on the 1M MovieLens dataset with different strategies to select negative training instances.

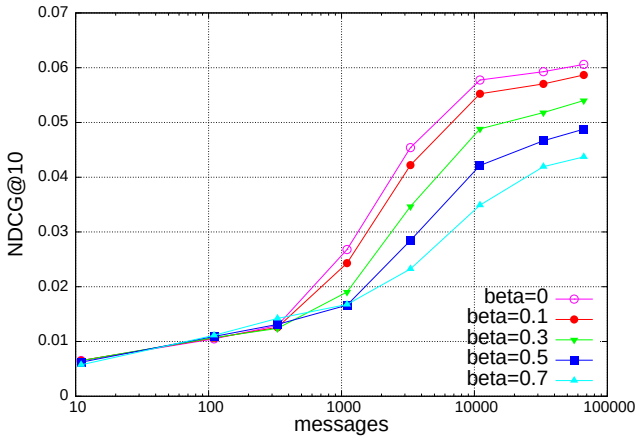

Figure 4: Peer-to-peer top-K recommendation on the 1M MovieLens dataset with varying mixing rate.

7.2 Top-K recommendation

For the top-K recommendation, first, we choose the strategy of selecting negative samples (line 6 in Algorithm 2). In Figure 3 we plot the performance of four strategies on the 1M dataset: (1) select randomly 10 unvisited items, (2) select randomly 60 items, (3) select randomly 60 items and use only the top ten of these, and (4) select to top ten of 1000 random ones. The performance is measured as average cumulative NDCG@10. More precisely, at each time step n, we sum the instantaneous NDCG@10 up to that point, and divide it with n. While in Algorithm 2 we refer to the measure as loss function, for NDCG the higher is the better. The performance seems very sensitive to the chosen negative sampling. The best performing strategy is to select randomly 60 item, and use the top 10 for the update step. We will use this strategy for the remaining set of experiments.

Turning to the peer-to-peer recommendation, the first question is how to mix the received vectors with the local ones (β in Algorithm 3). In this experiment the vectors of the positive instance and all negative instances are sent to the target users. The number of target users is fixed (in a similar way as in the fixed protocol above) to a particular value that is varied. Figure 4 shows the performance on the 1M dataset with different values for β. The average cumulative NDCG@10 is shown in this figure and the subsequent ones for the whole dataset. Thus, the data point on these figures correspond to the last measurement in Figure 5 (i.e. for the data point with largest number of time steps). The performance of the centralized algorithm is indeed identical to the performance of the peer-to-peer algorithm for the data point that satisfies the conditions enumerated in Section 5. Recall that the conditions include sending all negative items to all users and having $\beta = 0$. It is clear from Figure 4 that the best result is obtained for $\beta = 0$, in which case the received latent vector and bias replaces the local instance. In the following we will use this setting.

The next experiment investigates if all (10) negative instances should be sent, or just a random subset of these, along with the positive one. Therefore, in Figure 5 we plot the performance if 3, 5, 7, or 10 (all) negative instances are sent. Note that if we select the same number of target users and send only vectors corresponding to five negative instances the amount of communication is reduced (almost halved). The results for the four values are rather interest-

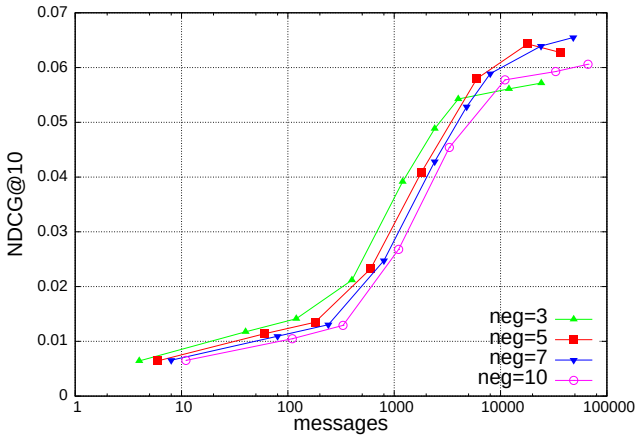

Figure 5: Peer-to-peer top-K recommendation on the 1M MovieLens dataset with varying item selection.

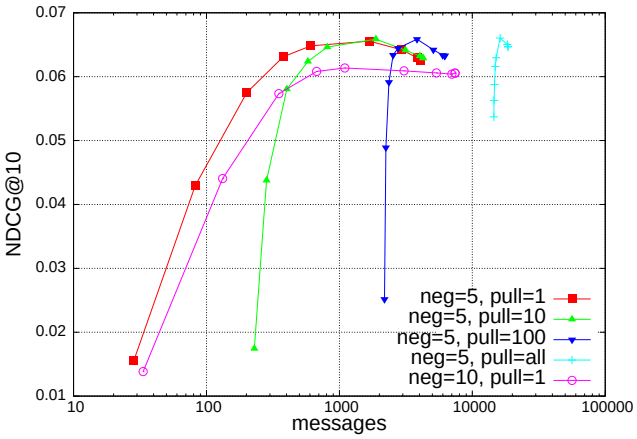

Figure 6: Peer-to-peer top-K recommendation on the 1M MovieLens dataset in the case of dynamic user availability.

ing. First, it is clear that if the number of messages are kept small, then it is better to select less items and use the reduction in the message to send to more users. Surprisingly, when all (or almost all) users are targets, performance gain can be obtained by not sending all negative items. In fact, this results in a performance gain over the centralized algorithm. We do not have a clear explanation for the improvement, it could be due to some additional diversity among the copies of the models, but this is purely a conjecture.

Up to this point, we assumed that users were in the system during the whole episode. In the final experiment we consider the situation when this is not the case. It is difficult to know from the data when a particular user was active, and we set this time frame as ranging from the first rating of the user until the last one (naturally the user could have registered much earlier, and could have been using the system for much later without performing any rating). In this scenario lines 2–6 of Algorithm 3 become relevant, and we have to decide from how many users to 'pull' the model. The results are shown in Figure 6 with different number of source users. The performances are shown for the case when five negative instances are sent after a gradient update (a combination of all negatives sent and one source user polled is also shown). It seems obvious from the figure that the communication efficient strategy is to request the model from only one user. If we send to the same number of target users (after gradient update) pulling models from more users increases the performance, but it comes at a cost of increased communication that can be better used after the gradient update.

For top-K recommendation, the results were presented for the 1M dataset. We observed a similar pattern of results for the 10M dataset (not shown). Finally, we would like to draw some comparison between how the performance degrades with decreasing communication for the two types of tasks, i.e. rating prediction and top-K recommendation. If we compare Figures 1 and 2 with Figures 4 and 5, the curves seems fairly similar taken into account that for the first task we minimize the measure, while for the second, we maximize. Figure 6 shows a different pattern. We observe a plateau at the end of each curve, clearest being for the curve corresponding to sending all negative samples and requesting from only one user (neg=10, pull=1). This suggests that if at each time step we are only concerned to send items that are active, then we do not lose in performance even if we the amount of communication drops by an order of magnitude.

7.3 Privacy

The privacy preserving properties of the peer-to-peer recommendation system is tested with the ScoreBoard algorithm described in Section 6. We consider the top-K recommendation task on the 1M MovieLens dataset with varying item selection and with continuous presence of all users (thus the same set-up as for Figure 5). We consider two scenarios: when the adversary knows 10 percent of the items rated by a particular user and when all ratings of the user are known. The probability of correctly identifying the user from the messages received is averaged over all users. If the adversary has access to the whole data set the detection probability is 0.48 and 0.96, respectively. The detection probability against the peer-to-peer algorithm is plotted in Figure 7 and Figure 8. In both figures the threshold δ was set to 0.5 which was found to result the highest probability of identifying correctly the users. The value plotted was chosen as the highest value at any time for a given set-up. In both scenarios (10 percent and all ratings) the detection probability is hugely decreased compared to the above mentioned baselines. Sending more negative items increases privacy (i.e. decreases detection probability) as well as sending to items to less peers.

Next, we compare the ranking performance/privacy tradeoff for the peer-to-peer algorithm and the centralized one using the k-CoRating algorithm to improve privacy. k-CoRating extends the rating matrix in a heuristic way such that for any user there are at least $k-1$ other users that rated exactly the same items. In an online scenario this is fairly difficult to achieve, but we assume that there is an oracle that knows in advance all the ratings and extends the rating matrix with ratings such that the above property will hold at the end. The artificial ratings are spread uniformly over the episode, and set to an average value.[3] The training for the centralized algorithm is performed on the extended

[3]The rating value is not critical since the ranking performance is not sensitive to the actual rating (cf. the equation of NDCG@K in Section 4).

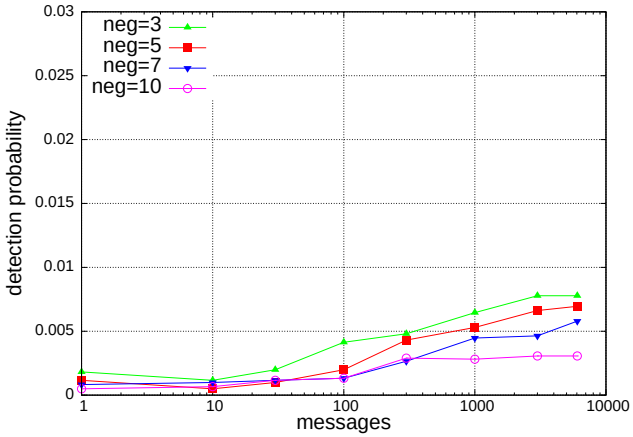

Figure 7: Detection probability on the 1M MovieLens dataset when 10% of the ratings are known for a particular user.

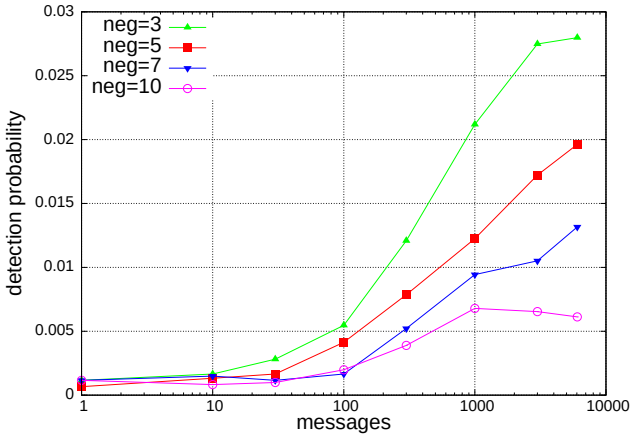

Figure 8: Detection probability on the 1M MovieLens dataset when all ratings are known.

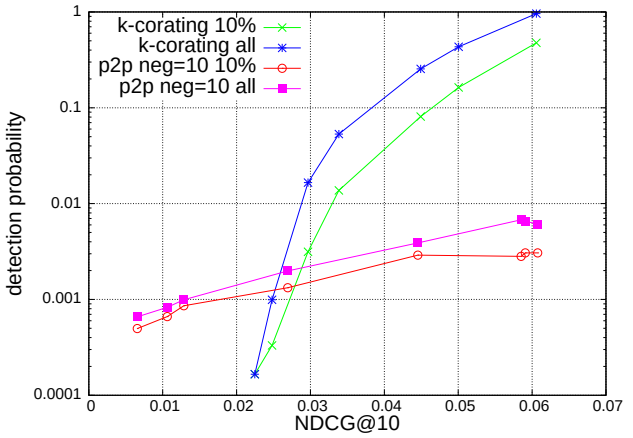

Figure 9: Detection probability vs. ranking performance.

rating sequence, the ranking performance is measured on the original sequence, and the detection probability (with the ScoreBoard algorithm) on the extended sequence.

The detection probability corresponding to various levels of ranking performance is plotted in Figure 9. For the peer-to-peer algorithm, we included the results with all negative items sent, with the number of target users varied (as in Figure 7 and 8, while for the centralized algorithm with k-CoRating the value k is varied. Again the adversary knows all ratings of a user or ten percent of it. We observe that to obtain a reasonable ranking performance k-CoRating is unable to ensure high level of privacy (values of k close to 1), while the peer-to-peer algorithm can achieve a good ranking performance and in the same time offer considerable privacy. It is interesting that k-CoRating can achieve perfect privacy and still provide a meaningful ranking, while the peer-to-peer algorithm is unable to achieve this in the considered version. It is expected that if more items are sent (other than the positive and negative samples) the privacy can be further improved without deteriorating the ranking performance. This comes however with the expense of increased communication costs.

8. CONCLUSIONS

In this paper we proposed an online peer-to-peer collaborative filtering algorithm that stores the ratings of a user and the private data local. Additionally, the users have a local copy of the common part of the factor model and communicate with other users to advance towards a consensus on it. A general form of the algorithm is proven to converge to a set of local optima in the stationary case. In a more specific form, we provided peer-to-peer matrix factorization algorithms for rating prediction and top-K recommendation.

We note that the general form is straightforward to instantiate to more intricate models such as factorization machines [26], or extend with context-aware features [2]. In the same way, it is easy to replace the negative sampling and gradient update step in the top-K recommendation algorithm with other choices employed in a centralized algorithm, e.g., [27] or [32]. Essentially, most algorithms where the private data can be separated, and use some form of stochastic gradient descent can be distributed in the same way as we did with our matrix factorization model in Section 5.

The online peer-to-peer matrix factorization algorithms were evaluated on the two larger MovieLens datasets that seem to us fairly non-stationary. We observed that the mean square error increases logarithmically with decreasing communication, and we suggested a simple way to control the amount of communication more efficiently. We expect that targeting users with similar tastes could make the communication more efficient, but we leave this issue for future research. For top-K recommendation we compared a few negative sampling strategies, and we showed how to handle new users. Crucially, we observed that sending only a subset of the negative samples could not only make the communication more efficient, but it could improve on the performance of the centralized algorithm. Understanding why the improvement is possible could help to improve centralized recommendation algorithms as well.

The peer-to-peer recommendation algorithm was shown to offer improved privacy without deteriorating the recommendation performance, which was not the case for the baseline algorithm tested.

9. REFERENCES

[1] J. Abernethy, K. Canini, J. Langford, and A. Simma. Online collaborative filtering. Technical report, UC Berkeley, Tech. Rep, 2011.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[3] M. Ali, C. C. Johnson, and A. K. Tang. Parallel collaborative filtering for streaming data. *University of Texas Austin, Tech. Rep*, 2011.

[4] C. Andrieu, É. Moulines, and P. Priouret. Stability of stochastic approximation under verifiable conditions. *SIAM Journal on control and optimization*, 44(1):283–312, 2005.

[5] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57(7):2748–2761, 2009.

[6] O. Besbes, Y. Gur, and A. Zeevi. Non-stationary stochastic optimization. *arXiv preprint arXiv:1307.5449*, 2013.

[7] P. Bianchi, G. Fort, and W. Hachem. Performance of a distributed stochastic approximation algorithm. *IEEE Trans. on Information Theory*, 59:7405–7418, 2013.

[8] P. Bianchi and J. Jakubowicz. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control*, 58(2):391–405, 2013.

[9] M. Boley, M. Kamp, D. Keren, A. Schuster, and I. Sharfman. Communication-efficient distributed online prediction using dynamic model synchronizations. In *First International Workshop on Big Dynamic Distributed Data (BD3@VLDB)*, pages 13–18, 2013.

[10] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec. Privacy-preserving distributed collaborative filtering. In *Networked Systems*, pages 169–184. Springer, 2014.

[11] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, pages 1–53, 2013.

[12] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13:165–202, 2012.

[13] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011.

[14] P. Hall and C. C. Heyde. *Martingale Limit Theory and Its Application*. Academic Press, 1980.

[15] I. Hegedus, M. Jelasity, L. Kocsis, and A. A. Benczúr. Fully distributed robust singular value decomposition. In *14-th IEEE International Conference on Peer-to-Peer Computing*, pages 1–9. IEEE, 2014.

[16] S. Isaacman, S. Ioannidis, A. Chaintreau, and M. Martonosi. Distributed rating prediction in user generated content streams. In *Proc. Fifth ACM Conf. on Rec. Sys.*, pages 69–76. ACM, 2011.

[17] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48, 2000.

[18] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[19] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[20] N. Lathia, S. Hailes, and L. Capra. Private distributed collaborative filtering using estimated concordance measures. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 1–8. ACM, 2007.

[21] N. Lathia, S. Hailes, and L. Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 796–797. ACM, 2009.

[22] O.-A. Maillard and R. Munos. Online learning in adversarial lipschitz environments. In *Machine Learning and Knowledge Discovery in Databases*, pages 305–320. Springer, 2010.

[23] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy, 2008.*, pages 111–125. IEEE, 2008.

[24] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh. Privacy-preserving matrix factorization. In *ACM SIGSAC Conf. on Comp. and Comm. Security*, pages 801–812. ACM, 2013.

[25] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Eighth IEEE International Conference on Data Mining (ICDM'08)*, pages 502–511. IEEE, 2008.

[26] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[27] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[28] C. Tekin, S. Zhang, and M. van der Schaar. Distributed online learning in social recommender systems. *arXiv preprint arXiv:1309.6707*, 2013.

[29] K. I. Tsianos and M. G. Rabbat. Consensus-based distributed online prediction and optimization. In *IEEE GlobalSIP Network Theory Symposium*, 2013.

[30] F. Yan, S. Sundaram, S. Vishwanathan, and Y. Qi. Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2483–2493, 2013.

[31] F. Zhang, V. E. Lee, and R. Jin. k-corating: Filling up data to obtain privacy and utility. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[32] W. Zhang, T. Chen, J. Wang, and Y. Yu. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 785–788, 2013.