Scalable and Explainable 1-Bit Matrix Completion via Graph Signal Learning

Chao Chen[1], **Dongsheng Li**[3,4], **Junchi Yan**[1,2*], **Hanchi Huang**[1], **Xiaokang Yang**[1]

[1] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[3] School of Computer Science, Fudan University, Shanghai, China
[4] Microsoft Research Asia, Shanghai, China
{chao.chen, yanjunchi, qq20001224m, xkyang}@sjtu.edu.cn, dongshengli@fudan.edu.cn

Abstract

One-bit matrix completion is an important class of positive-unlabeled (PU) learning problems where the observations consist of only positive examples, e.g., in top-N recommender systems. For the first time, we show that 1-bit matrix completion can be formulated as the problem of recovering clean graph signals from noise-corrupted signals in hypergraphs. This makes it possible to enjoy recent advances in graph signal learning. Then, we propose the spectral graph matrix completion (SGMC) method, which can recover the underlying matrix in distributed systems by filtering the noisy data in the graph frequency domain. Meanwhile, it can provide micro- and macro-level explanations by following vertex-frequency analysis. To tackle the computational and memory issue of performing graph signal operations on large graphs, we construct a scalable Nyström algorithm which can efficiently compute orthonormal eigenvectors. Furthermore, we also develop *polynomial and sparse* frequency filters to remedy the accuracy loss caused by the approximations. We demonstrate the effectiveness of our algorithms on top-N recommendation tasks, and the results on three large-scale real-world datasets show that SGMC can outperform state-of-the-art top-N recommendation algorithms in accuracy while only requiring a small fraction of training time compared to the baselines.

Introduction

This paper considers the problem of recovering a 0-1 matrix $\mathbf{M} \in \{0,1\}^{N \times M}$ only from positive and unlabeled data, which is also referred to as 1-bit matrix completion (Cai and Zhou 2013; Davenport et al. 2014; Hsieh, Natarajan, and Dhillon 2015). We assume that the positive samples are randomly drawn from $\{(i,u)|\mathbf{M}_{i,u}=1\}$ with probability $p(\mathbf{M}_{i,u}=1)$, and more precisely, we observe a subset Ω used for training in the presence of class-conditional random label noise which flips a 1 to 0 with probability ρ. Therefore, the unlabeled data is a mixture of unobserved positive examples and true negative examples, which raises challenges for formulating the underlying optimization problems.

Recent works (Jahrer and Töscher 2012; Park et al. 2015; He et al. 2016; Li et al. 2016; Wu, Hsieh, and Sharpnack 2017, 2018) have showed that treating all unlabeled examples as negative examples in supervised learning can obtain decent performances in practice although the learned models can be biased (Kiryo et al. 2017). However, treating all unlabeled examples as negative examples will exhibit high computational overhead because all $N \times M$ examples in $\mathbf{M}$ should be considered in training (Hu, Koren, and Volinsky 2008; Mackey, Jordan, and Talwalkar 2011; Lee et al. 2013; Chen et al. 2015), which is prohibitive for many applications with large-scale matrices, e.g., recommendation on millions of songs (Bertin-Mahieux et al. 2011). The other issue of existing 1-bit matrix completion methods is the lack of explainability (Abdollahi and Nasraoui 2016; Zhang and Chen 2018), either due to the blackbox nature of neural networks (Wang, Wang, and Yeung 2015; Lian et al. 2018; Liang et al. 2018; Zhou et al. 2018) or to the broken connection between the past actions and the future prediction due to the introduction of latent variables or other transformations (Cao et al. 2007; Rendle et al. 2009; Rendle 2010; Zheng et al. 2018).

To this end, in this paper we propose a scalable and explainable 1-bit matrix completion algorithm, namely spectral graph matrix completion (SGMC), in which the underlying matrix $\mathbf{M}$ is recovered by performing parallel signal processing on hypergraphs to improve the system scalability. The signals learned from the hypergraphs can be used to explain the models. To better illustrate the idea, let us consider the problem of top-N recommendation, in which we model a user's historical records as a signal $\mathbf{r}_u$ in a pre-specified item-item hypergraph $\mathcal{G}$ where the value $\mathbf{r}_u(i)$ at the i^{th} vertex represents whether the u^{th} user likes the i^{th} item. Then, we can pose the problem as recovering the underlying *clean* graph signal $\mathbf{m}_u$ from a noisy signal $\mathbf{r}_u$ that facilitates in flexible signal processing techniques such as graph Fourier transform and vertex-frequency analysis (Shuman et al. 2013; Shuman, Ricaud, and Vandergheynst 2016).

Motivation. The graph signal processing perspective to 1-bit matrix completion has been rarely studied yet appealing for several reasons. First, we are no longer limited to the geometrical proximity on the graph vertex domain, but are able to identify and exploit the structure in the graph frequency domain for potential performance improvement. Second, it enables us to take advantage of the well-developed vertex-frequency analysis to provide the explanations behind the predictions at both micro-level and macro-level.

The main contributions of this paper are as follows:

*Corresponding author is Junchi Yan.

• To our best knowledge, this is the first work for developing a graph signal processing formulation to the 1-bit matrix completion problem, and quantitatively as well as qualitatively justify why a graph signal processing perspective is effective in this problem.

• We propose a scalable and explainable algorithm, named spectral graph matrix completion, which minimizes unbiased risk estimators. With the benefit of spectral signals on graphs, our approach is enabled to provide micro- and macro-level explanations. This is one of the key differences to conventional solutions.

• We construct a scalable Nyström algorithm to compute orthonormal eigenvectors. In general, our SGMC model consumes $\mathcal{O}(N(K^2{+}L\eta){+}L^2K)$ time and $\mathcal{O}(\eta N(L{+}M))$ memory where $N{\gg}L{>}K$ and $N{\gg}\,\eta$.

• Top-N recommendation results on large datasets show that SGMC achieves state-of-the-art ranking accuracy, provides reasonable explanations, and requires a small fraction of training time compared to the best-performing baseline.

Related Work

One-bit matrix completion approaches either regard unlabeled data as negative data with smaller weights (Pan et al. 2008; Hsieh, Natarajan, and Dhillon 2015; Li et al. 2016; He et al. 2016), or treat unlabeled data as weighted positive and negative simultaneously (Natarajan et al. 2013; Du Plessis, Niu, and Sugiyama 2014, 2015; Kiryo et al. 2017).

The former (Hu, Koren, and Volinsky 2008; Jahrer and Töscher 2012; He et al. 2016) heavily relies on good choices of weights of unlabeled data, which is computationally expensive to tune. In practice, because the unlabeled dataset consists of both positive and negative data, this family of algorithms (Cao et al. 2007; Rendle et al. 2009; Park et al. 2015; Wu, Hsieh, and Sharpnack 2017, 2018) has a systematic estimation bias (Du Plessis, Niu, and Sugiyama 2014, 2015); By contrast, the latter focuses on unbiased risk estimators to avoid tuning the weights. However, most of existing works exhibit poor scalability due to high computation complexity on the very large matrix (Mackey, Jordan, and Talwalkar 2011; Lee et al. 2013; Chen et al. 2015).

We propose a composite loss (Eq. (5)) that can cancel the bias, and by applying *sparsity and orthonormality constraints* (Eq. (14)) our approach can offer different levels of explanations behind the predictions. To scale to very large datasets, we also devise a parallel matrix approximation method which guarantees the orthogonality of the outputs. This makes a difference from prior research. More related works are provided in **supplementary materials**.

The Model

Throughout this paper, we denote scalars by either lowercase or uppercase letters, vectors by boldface lowercase letters, and matrices by boldface uppercase letters. Unless otherwise specified, all vectors are considered as columns vectors. In addition, we define the following definitions in this paper as:

Definition 1. (**Hypergraph**). A undirected and connected hypergraph which consists of a finite set of vertices (*items*) $\mathcal{I}$ with $|\mathcal{I}| = N$ and a set of hyperedges (*users*) $\mathcal{U}$ with $|\mathcal{U}| = M$, is defined as $\mathcal{G} = \{\mathcal{I}, \mathcal{U}\}$. Each hyperedge is defined as a subset of $\mathcal{I}$ such that $\cup_{u\in\mathcal{U}} = \mathcal{I}$, and a hyperedge $u \in \mathcal{U}$ containing only two vertices is a simple graph edge.

Definition 2. (**Incidence Matrix**). Given any hypergraph $\mathcal{G} = \{\mathcal{I}, \mathcal{U}\}$, we say a hyperedge u is incident with a vertex i when $i \in u$. Then, this hypergraph $\mathcal{G}$ can be represented by an N-by-M incidence matrix (*item-user implicit feedback matrix*) $\mathbf{R}$ defined as following:

$$\mathbf{R}_{i,u} = \begin{cases} 1 & \text{if } i \in u \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Definition 3. (**Hypergraph Laplacian Matrix**). Given any hypergraph $\mathcal{G} = \{\mathcal{I}, \mathcal{U}\}$, we denote the degree of a vertex i by $d(i) = \sum_{u\in\mathcal{U}} \mathbf{R}_{i,u}$ and the degree of a hyperedge u by $\delta(u) = \sum_{i\in\mathcal{I}} \mathbf{R}_{i,u}$. $\mathbf{D}_v$ and $\mathbf{D}_e$ denote the diagonal matrices containing the vertex and hyperedge degrees, respectively. Then the hypergraph Laplacian matrix $\mathbf{Ł}$ can be defined as follows:

$$\mathbf{Ł} = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{R}\mathbf{D}_e^{-1}\mathbf{R}^\top\mathbf{D}_v^{-1/2}. \tag{2}$$

Definition 4. (**Graph Signal**). For a hypergraph $\mathcal{G} = \{\mathcal{I}, \mathcal{U}\}$, the data at each vertex in the graph is referred to as a graph signal that can be represented as a vector $\mathbf{r} \in \mathbb{R}^N$. The implicit data of the u^{th} user can be viewed as a signal $\mathbf{r}_u$ on $\mathcal{G}$, where the i^{th} entry of the vector $\mathbf{r}_u$ is equal to $\mathbf{R}_{i,u}$, namely $\mathbf{r}_u(i) = \mathbf{R}_{i,u}$.

Graph Fourier Transform

The *classical Fourier transform* is defined as an expansion of a function f w.r.t. complex exponentials:

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t)e^{-2\pi i\xi t}\,\mathrm{d}t, \tag{3}$$

and analogously the *graph Fourier transform* is defined as an expansion of a graph signal $\mathbf{r}_u$[1] in terms of the eigenvectors of the (hyper)graph Laplacian matrix $\mathbf{Ł}$ (Shuman et al. 2013). To be specific, let $\{\mathbf{v}_l\}_{l=0}^{N-1}$ and $\{\lambda_l\}_{l=0}^{N-1}$ denote the eigenvectors and eigenvalues of the hypergraph Laplacian $\mathbf{Ł}$ respectively and $\mathbf{r}_u \in \mathbb{R}^N$ be a graph signal on $\mathcal{G}$, then the graph Fourier transform and its inverse can be defined as:

$$\hat{\mathbf{r}}_u(\lambda_l) = \sum_{i=0}^{N-1} \mathbf{r}_u(i)\boldsymbol{\Phi}_{l,i}^\top \quad \text{and} \quad \mathbf{r}_u(i) = \sum_{l=0}^{N-1} \hat{\mathbf{r}}_u(\lambda_l)\boldsymbol{\Phi}_{i,l} \tag{4}$$

where $\hat{\mathbf{r}}_u(\lambda_l)$ represents the signal in the graph frequency domain, the eigenvalue λ_l carries a notion of frequency, and the complete set of orthonormal eigenvectors $\boldsymbol{\Phi} = [\mathbf{v}_0, \ldots, \mathbf{v}_{N-1}]$ serves as a basis in the forward and inverse graph Fourier transforms.

The Proposed SGMC Method

In practice, it is beneficial to assign different weights to different data examples. For instance, in recommender systems, we can punish popular items/users to achieve more accurate long-tail recommendations (Steck 2011, 2019). Similarly, we also set lower weights to the graph signals at

[1]In general, the graph signal can also be viewed as a function $\mathbf{r}_u : \mathcal{I} \to \mathbb{R}$ where $\mathcal{I}$ is the vertex set.

the vertices with higher degrees, namely $\mathbf{Y}_{i,u} = (1-\rho)\mathbf{M}_{i,u}/(d(i)\delta(u))^{\beta}$ where ρ is the rate of flipping a 1 to 0. Empirically, $\beta = 1/2$ can achieve decent performance. Then we recover the underlying matrix by minimizing the unbiased estimator defined on each entry (Natarajan et al. 2013), which leads to the following composite problem:

$$\min_{\text{rank}(\mathbf{X})=K} \frac{1}{MN}\sum_{i,u} \hat{\ell}\left((1-\rho)\mathbf{R}_{i,u}/\sqrt{d(i)\delta(u)}, \mathbf{X}_{i,u}\right) \quad (5)$$

where $\hat{\ell}(t,x) = [(t-x)^2 - \rho x^2]/(1-\rho)$ if $t > 0$ and $\hat{\ell}(t,x) = x^2$ otherwise.

Theorem 1. *(Unbiased Risk Estimator). Let ℓ be the squared loss. For any $\mathbf{X} \in \mathbb{R}^{N\times M}$ and $\mathbf{Y}_{i,u} = (1-\rho)\mathbf{M}_{i,u}/\sqrt{d(i)\delta(u)}$, $E_{i,u}\ell(\mathbf{Y}_{i,u}, \mathbf{X}_{i,u}) = E_{i,u}\hat{\ell}((1-\rho)\mathbf{R}_{i,u}/\sqrt{d(i)\delta(u)}, \mathbf{X}_{i,u})$.*

Minimizing the above optimization problem is equivalent to minimizing the following problem:

$$\min_{\mathbf{V},\mathbf{U}} \| \mathbf{D}_v^{-1/2}\mathbf{R}\mathbf{D}_e^{-1/2} - \mathbf{V}\mathbf{U}^\top \|_2^2 \quad (6)$$

where $\mathbf{V} \in \mathbb{R}^{N\times K}, \mathbf{U} \in \mathbb{R}^{M\times K}$.

It is expensive to solve Eq. (6) due to its non-convexity. However, Theorem 2 provides a clue to obtain a closed-form solution: when the hypergraph $\mathcal{G}$ is fully observed without noise, the optimal *orthonormal* $\mathbf{V}$ can be learned by solving the spectral clustering problem (Shi and Malik 2000).

Theorem 2. *Suppose that a hypergraph $\mathcal{G} = \{\mathcal{I}, \mathcal{U}\}$ is fully observed without noise and its incidence matrix $\mathbf{R} \in \mathbb{R}^{N\times M}$ is of rank K, then the matrix whose columns are the eigenvectors with the K lowest eigenvalues of the (hyper)graph Laplacian Ł of $\mathcal{G}$ is the optimal solution of*

$$\min_{\text{rank}(\mathbf{V})=K} \| \mathbf{D}_v^{-1/2}\mathbf{R}\mathbf{D}_e^{-1/2} - \mathbf{V}\mathbf{U}^\top \|_2^2 \quad \text{s.t.,} \quad \mathbf{V}^\top\mathbf{V} = \mathbf{I}_K$$

where $\mathbf{D}_v$ and $\mathbf{D}_e$ denote the diagonal degree matrices of the vertex and hyperedge respectively, and $\mathbf{U}$ is an optimal M-by-K matrix.

However, the actual hypergraph in the targeted problem is *partially observed* in the presence of noise. To relax this challenge, we pose an assumption that the partially observed hypergraph retains a few *principal components* in its clean version. This allows us to construct an accurate low-rank approximation. Then we define SGMC which consists of the following three steps:

Step 1: We first follow Theorem 2 to learn $\mathbf{V}$, where the eigenvectors of Ł correspond to the optimal result. Notably, $\mathbf{V}$ preserves many properties of the graph, different from (Hu, Koren, and Volinsky 2008; Jahrer and Töscher 2012).

Step 2: We then optimize $\mathbf{U}$ by minimizing the following least-squared empirical risk given $\mathbf{V}$:

$$\min_{\text{rank}(\mathbf{U})=K} \| \mathbf{D}_v^{-1/2}\mathbf{R}\mathbf{D}_e^{-1/2} - \mathbf{V}\mathbf{U}^\top \|_2^2, \quad (7)$$

where $\mathbf{D}_v$ and $\mathbf{D}_e$ denote the diagonal degree matrices of the vertex and hyperedge, respectively.

Step 3: We finally calculate the estimated matrix $\mathbf{S}$ of the underlying matrix $\mathbf{M}$ as follows:

$$\mathbf{S} = \mathbf{D}_v^{1/2}\mathbf{V}\mathbf{U}^\top\mathbf{D}_e^{1/2} = \mathbf{D}_v^{1/2}\mathbf{V}\mathbf{V}^\top\mathbf{D}_v^{-1/2}\mathbf{R}, \quad (8)$$

where the observation $\mathbf{R}$ is scaled prior to model training and the predicted value is rescaled back to the original space. This makes a lot of sense in practice, as it helps the predictions reflect the full popularity bias in the training data.

Remark. We next view our method from a perspective of graph signal processing. Denote by $\bar{\mathbf{r}}_u = \mathbf{D}_v^{-1/2}\mathbf{r}_u$ a normalized graph signal where $\mathbf{r}_u$ is the u^{th} column of $\mathbf{R}$, and by $\bar{\mathbf{s}}_u = \mathbf{D}_v^{-1/2}\mathbf{s}_u$ the output signal as defined:

$$\begin{aligned}\bar{\mathbf{s}}_u = \mathbf{\Phi}\hat{h}(\text{Ł})\mathbf{\Phi}^\top\bar{\mathbf{r}}_u &= \sum_{l=0}^{N-1}\mathbf{v}_l\hat{h}(\lambda_l)\mathbf{v}_l^\top\bar{\mathbf{r}}_u \\ &= \sum_{l=0}^{K-1}\mathbf{v}_l\mathbf{v}_l^\top\bar{\mathbf{r}}_u = \mathbf{V}\mathbf{V}^\top\bar{\mathbf{r}}_u,\end{aligned} \quad (9)$$

where $\hat{h}(\text{Ł})\mathbf{\Phi}^\top\bar{\mathbf{r}}_u$ acts as graph frequency filtering with signal $\mathbf{\Phi}^\top\bar{\mathbf{r}}_u$ in the graph frequency domain and transfer function $\hat{h}(\text{Ł}) = \text{diag}(\hat{h}(\lambda_0), \ldots, \hat{h}(\lambda_{N-1}))$ with a low-pass kernel $\hat{h}(\lambda_l) = \mathbf{1}_{l<K}$. This filter only passes the graph signal $\bar{\mathbf{r}}_u$ within K lowest frequencies, namely low-pass filter. In effect, we can rewrite Eq. (9) as $\mathbf{s}_u = \mathbf{D}_v^{1/2}\mathbf{V}\mathbf{V}^\top\mathbf{D}_v^{-1/2}\mathbf{r}_u$ which resembles Eq. (8), indicating that the learning of SGMC is equivalent to filtering the graph signals with a low-pass kernel. Perhaps more importantly, this filter paraphrases our assumption in the graph frequency domain — *the partially observed hypergraph preserves the signal within low frequencies.*

Scalable Approximations for SGMC

The Laplacian Ł of a hypergraph $\mathcal{G}$ with N vertices and M hyperedges requires $\mathcal{O}(N^2M)$ computational time and $\mathcal{O}(N^2)$ memory use. The time complexity of standard eigendecomposition is $\mathcal{O}(N^3)$. Thus, SGMC with a standard eigendecomposition on Ł is prohibitive on large datasets.

One of the solutions is to zero out some elements in the hypergraph Laplacian Ł or to sparsify Ł (Chen et al. 2010; Lehoucq, Sorensen, and Yang 1998). Nonetheless, the time reduction is significant only when Ł is sparse or very few eigenvectors are extracted (Williams and Seeger 2001). The other method is based on the Nyström method (Kumar, Mohri, and Talwalkar 2009; Williams and Seeger 2001), which solves the eigendecomposition problem for a small random subset of the vertices, then extrapolates this solution to the full set of vertices in the hypergraph (Fowlkes et al. 2004). However, a sufficiently large amount of columns of Ł have to be sampled to yield an accurate approximation. This makes this class of algorithms impractical on very large-scale datasets, because the eigendecomposition on the resultant column-based submatrix will soon dominate the computation and become prohibitive (Li, Kwok, and Lü 2010).

To address above issues, we propose efficient and scalable approximations for SGMC, which combines merits of the Nyström method (Williams and Seeger 2001) and the randomized algorithm (Halko, Martinsson, and Tropp 2011).

Randomized Algorithm for Partial Decompositions

Recall that we assume the partially observed hypergraph retains a few principal components in its clean version. This means, we are only interested in the first K components of the eigendecomposition of the N-by-N hypergraph Laplacian $\mathbf{Ł}$ where $K \ll N$. Among the solutions for partial matrix decompositions, randomized algorithms (Halko, Martinsson, and Tropp 2011) are simple and can deliver an accurate approximation with very high probability.

There are two main stages in this class of algorithms (Halko, Martinsson, and Tropp 2011): 1) we first draw a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{N \times (K+p)}$ with an over-sampling parameter p (e.g., 5) and form $\mathbf{Y} = \mathbf{Ł}^q \mathbf{\Omega}$ where q (e.g., 1) is used to accelerate the decay of eigenvalues of $\mathbf{Ł}$, then find an orthonormal matrix $\mathbf{Q}$ (e.g., by QR decomposition) such that $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^\top \mathbf{Y}$; and 2) we reduce $\mathbf{Ł}$ to a small matrix $\mathbf{B} = \mathbf{Q}^\top \mathbf{Ł} \mathbf{Q}$, then by performing standard eigendecomposition on the reduced matrix we can obtain $\mathbf{B} = \mathbf{\Phi}_B \mathbf{\Sigma}_B \mathbf{\Phi}_B^\top$. Finally, the K principal eigenvectors and eigenvalues of the hypergraph Laplacian $\mathbf{Ł}$ can be approximated by $\mathbf{V}_B = \mathbf{Q}(\mathbf{\Phi}_B)_{:,0:K-1}$ and $\mathbf{\Lambda}_B = (\mathbf{\Sigma}_B)_{0:K-1,0:K-1}$, respectively.

Remark. It takes $\mathcal{O}(N^2K)$ time for $\mathbf{Y}$, $\mathcal{O}(NK)$ time for QR decomposition, $\mathcal{O}(NK^2)$ time for $\mathbf{B}$ and $\mathcal{O}(K^3)$ for the eigendecomposition (Halko, Martinsson, and Tropp 2011; Li, Kwok, and Lü 2010). Since the total complexity is quadratic to N and one pass over $\mathbf{Ł}$ is needed, a direct use of randomized algorithms on $\mathbf{Ł}$ is still highly expensive.

Approximate SGMC

Inspired by (Li, Kwok, and Lü 2010), we speed up the inner eigendecomposition in the Nyström method by using the randomized algorithm, such that we are allowed to sample a large subset of columns of the hypergraph Laplacian $\mathbf{Ł}$ for accurate approximation. To be more specific, we present the three key steps of the proposed efficient and scalable approximations for SGMC as follows:

Step 1: We randomly sample a subset of high-degree vertices $\mathcal{I}_A$ with $|\mathcal{I}_A| = L$ and let $\mathcal{I}_B = \mathcal{I} - \mathcal{I}_A$ denote the set of the remaining vertices. We then rearrange the rows and columns of the hypergraph Laplacian $\mathbf{Ł}$ such that

$$\mathbf{Ł} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \quad \text{and} \quad \mathbf{W} = \mathbf{A} + \mathbf{A}^{-1/2}\mathbf{B}\mathbf{B}^\top\mathbf{A}^{-1/2},$$

where $\mathbf{A}$ represents the $L \times L$ matrix of affinities between the sample vertices in $\mathcal{I}_A$, and $\mathbf{B}$ represents the $L \times (N-L)$ matrix of affinities between the sample vertices and the remaining vertices. Notably, we construct a rank-K approximation of $\mathbf{A}^{-1/2}$ by first approximating $\mathbf{A}$ with a rank-K matrix $\mathbf{V}_A \Lambda_A \mathbf{V}_A^\top$ using the above randomized techniques (Halko, Martinsson, and Tropp 2011), then computing $\mathbf{A}^{-1/2} \simeq \mathbf{V}_A \Lambda_A^{-1/2} \mathbf{V}_A^\top$.

Step 2: We likewise use the randomized method (Halko, Martinsson, and Tropp 2011) to obtain the rank-K approximation of the matrix $\mathbf{W} \simeq \mathbf{V}_W \Lambda_W \mathbf{V}_W^\top$, then by using Nyström method (Fowlkes et al. 2004) we have the approximate K **orthonormal** eigenvectors and eigenvalues of the hypergraph Laplacian $\mathbf{Ł}$ as follows:

$$\widetilde{\mathbf{V}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B}^\top \end{bmatrix} \mathbf{A}^{-1/2}\mathbf{V}_W \mathbf{\Lambda}_W^{-1/2} \quad \text{and} \quad \widetilde{\mathbf{\Lambda}} = \mathbf{\Lambda}_W \tag{10}$$

Step 3: We finally derive the approximate SGMC as $\mathbf{S} = \mathbf{D}_v^{1/2}\widetilde{\mathbf{V}}\widetilde{\mathbf{V}}^\top \mathbf{D}_v^{-1/2}\mathbf{R}$.

Remark. Recall that $\mathbf{R}$ has M columns and N rows with Ω observations and $N \gg \eta = \Omega/M$, and we sample L columns to approximate K eigenvectors of the Laplacian $\mathbf{Ł}$. Then, it takes totally $\mathcal{O}(\eta N L)$ time for $\mathbf{A}$ and $\mathbf{B}$ which is significantly lower than $\mathcal{O}(\eta N^2)$ cost for $\mathbf{Ł}$, and $\mathcal{O}(L^2K + K^2(N-L))$ time for $\mathbf{W}$, $\mathcal{O}(L^2K + K^3)$ for the randomized matrix decomposition on $\mathbf{A}$ and $\mathbf{W}$. Since $N \gg L \geq K$, the approximate SGMC scales linearly with N. Note that our method for partial matrix decompositions has the same time complexity with (Li, Kwok, and Lü 2010), but the approximate eigenvectors in (Li, Kwok, and Lü 2010) are not orthogonal which has been proved inferior regarding performance (Chen et al. 2010).

Polynomial and Sparse Extensions for SGMC

The approximate SGMC trades accuracy for scalability. This section considers how to improve the model performance by means of extending the SGMC to the polynomial and sparse frequency filters. Let us define a P-order polynomial filter kernel as $\hat{h}_\theta(\lambda_l) = \sum_{k=0}^{P} \theta_k \lambda_l^k$ parameterized by θ, then the estimate in Eq. (9) can be extended as follows:

$$\mathbf{s}_u^{(\theta)} = \mathbf{D}_v^{1/2}\mathbf{V}\hat{h}_\theta(\mathbf{Ł})\mathbf{V}^\top\mathbf{D}_v^{-1/2}\mathbf{r}_u = \sum_{k=0}^{P} \theta_k \mathbf{D}_v^{1/2}\widetilde{\mathbf{Ł}}^k \mathbf{D}_v^{-1/2}\mathbf{r}_u$$
$$\widetilde{\mathbf{Ł}}^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^\top, \tag{11}$$

where θ_k controls the contribution of the k^{th} order estimate $\mathbf{s}_u^{(k)} = \mathbf{D}_v^{1/2}\widetilde{\mathbf{Ł}}^k\mathbf{D}_v^{-1/2}\mathbf{r}_u$. The parameter θ can be optimized by minimizing the empirical risk defined on set Ω_θ:

$$\min_\theta \frac{1}{|\Omega_\theta|} \sum_{(i,u)\in\Omega_\theta} \hat{\ell}((1-\rho)\mathbf{R}_{i,u}/\sqrt{d(i)\delta(u)}, \mathbf{s}_u^{(\theta)}(i))$$
$$+ \frac{1}{2}\alpha\gamma \parallel \theta \parallel_2^2 + \alpha(1-\gamma) \parallel \theta \parallel_1, \tag{12}$$

where $\mathbf{s}_u^{(\theta)}$ is defined in Eq.(11), and $\rho, \hat{\ell}$ are defined in Eq. (5), and α, γ are regularization parameters. Empirically, the cost of learning θ is very low since its size is typically up to 3, such that a small Ω_θ is good enough to derive accurate results. We highlight the empirical results in Table 1.

Micro-level Explanation. We next show how Eq. (11) provides a way to explain the estimate $\mathbf{s}_u(i)$ in terms of the positive data examples. In effect, $\mathbf{s}_u(i)$ can be reformulated as a linear combination of the data at vertices within the P-hop local neighborhood $\mathcal{N}(i, P)$:

$$\mathbf{s}_u(i) = \sum_{j=0}^{N-1} \mathbf{r}_u(j)\left(\sqrt{\frac{d(i)}{d(j)}}\sum_{k=0}^{P}\theta_k\widetilde{\mathbf{Ł}}_{i,j}^k\right) = \sum_{j\in\mathcal{N}(i,P)} \mathbf{r}_u(j)\mathbf{B}_{i,j}$$
$$\mathbf{B}_{i,j} = \sqrt{\frac{d(i)}{d(j)}}\sum_{k=0}^{P}\theta_k\widetilde{\mathbf{Ł}}_{i,j}^k, \tag{13}$$

where $\mathbf{r}_u(j)$ is the (j,u)-th entry of the observation matrix R, $d(i)$ is the degree of the i^{th} vertex, and $\mathbf{B}_{i,j}$ is nonzero i.i.f the j^{th} vertex is connected to the i^{th} vertex in P-steps on $\mathcal{G}$. Hence, the positive data examples $\{(j,u)|\mathbf{R}_{j,u} = 1\}$

with the highest coefficients $\mathbf{B}_{i,j}$ can be identified as the major explanations behind the estimate $\mathbf{s}_u(i)$.

Macro-level Explanation. The Fourier bases $\mathbf{\Phi}$ are the eigenvectors of the Laplacian matrix $\mathbf{Ł}$, which account for the intrinsic geometric structure of the underlying graph $\mathcal{G}$. As any graph signal $\mathbf{r}_u$ can be represented as linear combinations of the Fourier bases, the signal $\hat{\mathbf{r}}_u$ in the frequency domain measures the contribution of each component.

For example, in the context of recommender systems, the graph signal $\mathbf{r}_u(i)$ represent whether the u^{th} user likes the i^{th} item. In the sense that the eigenvectors $\mathbf{\Phi}$ can partition the items into the clusters (Shi and Malik 2000), $\hat{\mathbf{r}}_u(k)$ measures how much the u^{th} user likes the k^{th} item cluster. In practice, due to the fact that each user mainly focuses on a few categories of items, it is desirable to assume that $\hat{\mathbf{r}}_u$ is sparse. We hence add the sparsity constraint to Eq. (7), which leads to the following optimization problem:

$$\min_{\hat{\mathbf{r}}_u} \parallel \operatorname{sign}(\bar{\mathbf{r}}_u) \circ (\bar{\mathbf{r}}_u - \mathbf{V}\hat{h}_\theta(\mathbf{Ł})\hat{\mathbf{r}}_u) \parallel_2^2 + \frac{1}{2}\alpha\gamma \parallel \hat{\mathbf{r}}_u \parallel_2^2 + \alpha(1-\gamma) \parallel \hat{\mathbf{r}}_u \parallel_1, \quad (14)$$

where $\operatorname{sign}(\bar{\mathbf{r}}_u)$ returns the sign of the normalized graph signal $\bar{\mathbf{r}}_u = \mathbf{D}_v^{-1/2}\mathbf{r}_u$ and $\circ$ denotes the element-wise product. The regularization parameters α, γ mitigate overfitting.

However, in the majority of cases, the user preferences vary along the time. To address this issue, we propose to use a sliding window over $\bar{\mathbf{r}}_u$ and study user preferences in each fragments: (1) we use the first w (e.g., 10) purchased items to form $\bar{\mathbf{r}}_{u,1:w}$ and optimize $\hat{\mathbf{r}}_{u,1:w}$ by minimizing:

$$\min_{\hat{\mathbf{r}}_{u,1:w}} \parallel \operatorname{sign}(\bar{\mathbf{r}}_{u,1:w}) \circ (\bar{\mathbf{r}}_{u,1:w} - \mathbf{V}\hat{h}_\theta(\mathbf{Ł})\hat{\mathbf{r}}_{u,1:w}) \parallel_2^2 + \frac{1}{2}\alpha\gamma \parallel \hat{\mathbf{r}}_{u,1:w} \parallel_2^2 + \alpha(1-\gamma) \parallel \hat{\mathbf{r}}_{u,1:w} \parallel_1, \quad (15)$$

then (2) the second fragment $\bar{\mathbf{r}}_{u,1+s:1+s+w}$ is formed from the first one with step size s (e.g., 5) and so on. Essentially, the outputs $\{\hat{\mathbf{r}}_{u,t:t+w}\}$ describe how the user preferences vary along a path graph of the items.

Remark. Note the problem defined in Eq. (14) has the same order of complexity as that in Eq. (7). There are two reasons: first, Eq. (14) only measures the reconstruction errors of the observations with positive labels where $N \gg \eta = \Omega/M$; Second, learning $\hat{\mathbf{r}}_u$ in Eq. (14) for different u can be easily parallelized, and for each one it takes $\mathcal{O}(l^3 + Kl^2)$ time by using LARS-EN (Zou and Hastie 2005) where l is the number of iterations of parameter updates.

Experiment

This section studies the performance of the proposed SGMC algorithm in top-N recommendation tasks. We demonstrate that SGMC can not only make scalable and accurate recommendations on large datasets, but also is able to explain the recommendations in micro- and macro-level. All experiments are conducted on a server with an Intel Xeon(R) CPU E5-2678 2.50GHz CPU and 128G RAM.

Experimental Setup

Datasets We use three large-scale real-world datasets – (1) MovieLens 20M data (ML20M) (Harper and Konstan 2015) (2×10^7 ratings of 138,493 users and 26,611 items); (2) Netflix prize data (Netflix) (Bennett, Lanning et al. 2007) (10^8 ratings of 480,189 users and 17,770 items); and (3) Million Song data (MSD) (Bertin-Mahieux et al. 2011) (5×10^7 ratings of 1,019,318 users and 384,546 items). Note that the data here is binarized, and for each dataset we split it into train and test sets randomly with the ratio of 9:1. We adopt the ranking metrics of normalized discounted cumulative gain (NDCG) and F1 score, and all reported results are averaged over five different random train-test splits.

We implement our SGMC algorithm and its extensions using Apache Spark, where the matrix multiplications are parallelized. In the following experiments, we use the factor size $K = 1000$ for all SGMC models and the elastic-net parameters $\alpha = 0.007$ and $\gamma = 0.01$ in Eq. (14). Meanwhile, we sample $L = 6000$ columns of $\mathbf{Ł}$ for approximate SGMC. In addition, we compare the large-scale algorithms with publicly available codes provided by the authors:

- **CollRank** (Park et al. 2015) is the large-scale collaborative ranking algorithm which uses alternating minimization to optimize the pair-wise loss as in BPR (Rendle et al. 2009). The author parallelized their algorithm using the parallel computing techniques in OpenMP. Grid search of regularization parameter over $\lambda \in \{10, 100, 1000\}$ and factor size over $K \in \{50, 100, \ldots, 300\}$ is performed. We limit the number of ranking pairs to 3000 per user on the full Netflix and MSD datasets.

- **Primal-CR** (Wu, Hsieh, and Sharpnack 2017) reduces the total complexity of CollRank (Park et al. 2015) to near-linear by rearranging the computations of the gradients and Hessian vector product. The authors implemented the algorithm using the parallel computing techniques in Julia. Similarly, regularization parameter and factor size are selected by grid search over $\lambda \in \{1000, 2000, \ldots, 10000\}$ and $K \in \{100, 200, \ldots, 500\}$. We limit the number of ranking pairs per user to 5000 on the full MSD dataset.

- $\hat{\mathbf{B}}^{(\text{sparse})}$ (Steck 2019) resembles SLIM (Ning and Karypis 2011) with a closed-form solution and leverages the sparse inverse co-variance estimate for the reduction of training time. We adopt the source code provided by the authors and rewrite part of it in Spark to improve its efficiency. We use grid search over the optimal regularization over $\lambda \in \{1, 3, 6, 9\}$ and hyper-parameter over $r \in \{1/8, 2/8, \ldots, 5/8\}$.

We do not compare to neural models (e.g., NeuCF (He et al. 2017) and MULT-VAE (Liang et al. 2018)) and other graph models (e.g., SpectralCF (Zheng et al. 2018) and NGCF (Wang et al. 2019)), because the full MSD is too large for these models to finish in a reasonable amount of time or to store in the memory of our GTX 1080Ti GPU.

Ablation Analysis

To assess the effects of different components, we first perform an ablation study on the full ML20M data, with results in Table 1. SGMC takes the longest training time, while the approximate SGMC (i.e., approx SGMC) trades the accuracy for speed: the training time has been reduced from about 10 minutes for the exact solution to under 3 minutes

			F1 Score			NDCG	
	Models	**Training Time**	@5	@50	@100	@5	@50	@100
ML20M	**SGMC**	9 min 54 sec	0.1167	0.1023	0.0764	0.2257	0.2538	0.2767
	apprx SGMC	2 min 41 sec	0.1170	0.1016	0.0755	0.2254	0.2528	0.2749
	apprx SGMC$^{(\mathrm{sp})}$	3 min 25 sec	0.1187	0.1012	0.0762	0.2278	0.2550	0.2775
	apprx SGMC$^{(\mathrm{sp+poly})}$	3 min 27 sec	0.1188	0.1035	0.0776	0.2290	0.2568	0.2801

Table 1: Ablation study on full ML20M (138,493 users & 26,611 items). The approximate SGMC (i.e., $\mathrm{apprx\ SGMC}$) reduces training time with lower accuracy. Sparsifying spectral signals (*sp*) and 2nd-order polynomial filter (*poly*) improve accuracy.

			F1 Score			NDCG		
	Models	**Training Time**	@5	@50	@100	@5	@50	@100
ML20M	**Collrank**	2 hour 21 min	0.0740	0.0742	0.0584	0.1421	0.1771	0.2012
	Primal-CR	1 hour 24 min	0.0825	0.0796	0.0620	0.1493	0.1912	0.2156
	$\hat{\mathbf{B}}^{(\mathrm{sparse})}$	7 min 18 sec	0.1183	0.1012	0.0760	0.2219	0.2519	0.2758
	apprx SGMC$^{(\mathrm{sp+poly})}$	3 min 31 sec	0.1188	0.1035	0.0776	0.2290	0.2568	0.2801
Netflix	**Collrank**	1 hour 43 min	0.0081	0.0107	0.0101	0.0179	0.0225	0.0257
	Primal-CR	1 hour 42 min	0.0082	0.0115	0.0107	0.0197	0.0226	0.0265
	$\hat{\mathbf{B}}^{(\mathrm{sparse})}$	12 min 50 sec	0.0146	0.0171	0.0143	0.0347	0.0366	0.0407
	apprx SGMC$^{(\mathrm{sp+poly})}$	3 min 46 sec	0.0146	0.0174	0.0145	0.0359	0.0372	0.0411
MSD	**Collrank**	1 hour 26 min	0.0384	0.0246	0.0181	0.0524	0.0813	0.0935
	Primal-CR	1 hour 1 min	0.0402	0.0296	0.0205	0.0532	0.0870	0.0996
	$\hat{\mathbf{B}}^{(\mathrm{sparse})}$		— out of memory error —					
	apprx SGMC$^{(\mathrm{sp+poly})}$	4 min 40 sec	0.1006	0.0534	0.0354	0.1462	0.1869	0.2042

Table 2: Performance comparison on the full ML20M (138,493 users and 26,611 items), Netflix (480,189 users and 17,770 items) and MSD (1,019,318 users and 384,546 items) datasets. Unless otherwise specified, $\mathrm{apprx\ SGMC}^{(\mathrm{sp+poly})}$ stands for the approximate SGMC with using the sparse and 2nd-order polynomial frequency filter. $\mathrm{apprx\ SGMC}^{(\mathrm{sp+poly})}$ obtains competitive ranking accuracy while requiring only a small fraction of the training time.

with small accuracy loss. On the other hand, such a loss is remedied by $\mathrm{apprx\ SGMC}^{(\mathrm{sp})}$ which sparsified the signal in the graph frequency domain, as well as its enhanced version $\mathrm{apprx\ SGMC}^{(\mathrm{sp+poly})}$ using the 2nd-order polynomial low-pass filter. This shows the effectiveness of the sparse and polynomial frequency filters.

Another interesting observation is that the third-order or higher-order polynomial filter does not contribute to further improvement of the accuracy. This is because most items (vertices) can be reached from each other in a small number of steps (e.g., 3 steps here), such that the high-order polynomial filters actually extend the neighborhood to the whole item set. Prior works (Deshpande and Karypis 2004; Ning and Karypis 2011) have shown that extremely large size of neighborhood is not helpful for performance.

Quantitative Analysis

Table 2 summarizes the experimental results of the baselines and our $\mathrm{apprx\ SGMC}^{(\mathrm{sp+poly})}$ model on the full ML20M, Netflix and MSD datasets. We note that *all sparse users and items* are kept in our study to better study the model scalability and recommendation accuracy. It shows that $\hat{\mathbf{B}}^{(\mathrm{sparse})}$ achieves the best results among baselines, and our model outperforms all three competitive algorithms in recommendation accuracy across all three datasets. This demonstrates that the informative feature in the graph frequency domain is important for making better recommendations. In addition, we can see that the Collrank and Primal-CR models perform poorly on the very large Netflix and MSD datasets. This is due to the insufficient ranking pairs used in the model training. However, the fact is that when we try to apply greater than 3000 and 5000 ranking pairs per user for the Collrank and Primal-CR models respectively, these algorithms run very slowly and report memory error. This memory error also happens to $\hat{\mathbf{B}}^{(\mathrm{sparse})}$ on the MSD dataset when computing the sparse inverse covariance estimation.

Besides the difference in accuracy, Table 2 shows that the training time of SGMC is at least two times less than that of $\hat{\mathbf{B}}^{(\mathrm{sparse})}$, and the margin grows with the increasing scale of the data. This can be attributed to two factors: 1) the randomized algorithm (Halko, Martinsson, and Tropp 2011) that delivers accurate approximations for the eigendecomposition on the column-based submatrix of the hypergraph Laplacian $\mathbf{Ł}$; 2) the Nyström method (Kumar, Mohri, and Talwalkar 2009; Williams and Seeger 2001) which extrapolates this approximation to the full $\mathbf{Ł}$. Also note that Primal-CR has larger factor size and more pairwise comparisons than Collrank. That is why Primal-CR seems to have the comparable training time with the Collrank on Netflix and MSD.

Braveheart (1995)	**The Sword in the Stone**	**Lord of the Rings I**
Jurassic Park (1993)	Snow White and the Seven Dwarfs	Pirates of the Caribbean I
Forrest Gump (1994)	Peter Pan	Lord of the Rings II
The Shawshank Redemption (1994)	Alice in Wonderland	Lord of the Rings III

Table 3: Three recommendations (bold-faced) with micro-level explanations for an ML20M user. Each recommended movie is recommended due to a unique set of already-watched movies, while the three movies used as explanations contribute the most.

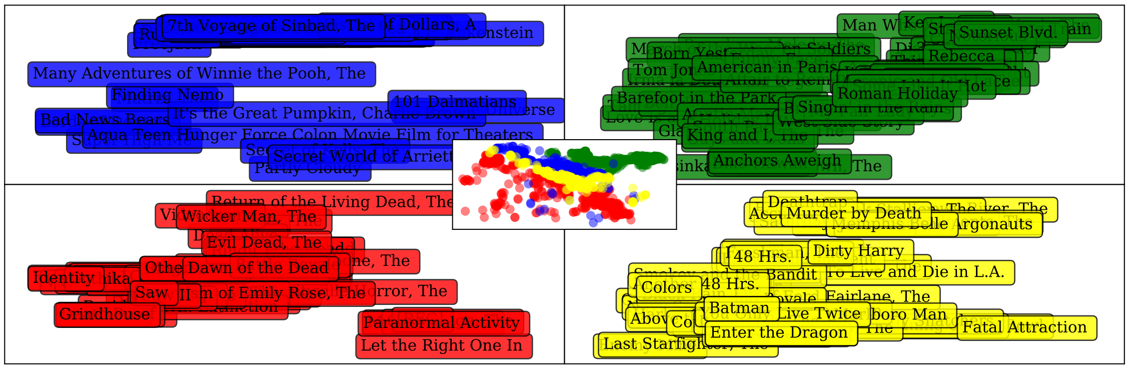

Figure 1: Static macro-level explanations for an ML20M user. We use t-SNE to visualize the top-four favorite item clusters of this user. We can discern animation (blue), comedy romance (green), thriller (red), action comedy (yellow).

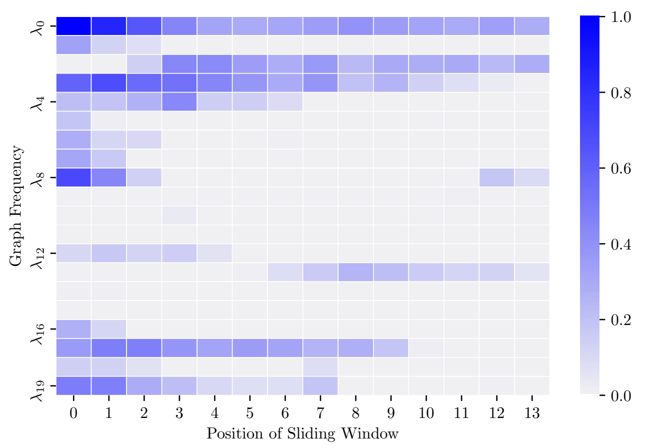

Figure 2: Dynamic macro-level explanations. We use a sliding window of size w=10 over one's behavior sequence with step size s=3, and the resulted frequency signals are normalized for better illustration. It visualizes how user preferences vary along the sequence (i.e., different t in $\{r_{u,t:t+w}\}$) and shows the dynamic nature of user preferences.

Qualitative Analysis

Table 3 provides the micro-level explanations to each of three recommendations for a user on ML20M. As shown in Eq. (13), we recall that the ranking score of each recommended movie is contributed by a unique set of already-watch movies, and the top-3 ones which contribute the most are regarded as the explanations behind the recommendation. In detail, *Braveheart* is pushed to the user because she/he previously watched the Oscar winning movies, and likewise the high ranking-scores of *The Sword in the Stone* and *Lord of the Rings* are largely impacted by the watched children animation movies (e.g., Snow White) as well as adventure fantasy movies (e.g., Pirates of the Caribbean).

We also analyze the user's macro-level preferences in both static and dynamic manners. By using t-SNE (Maaten and Hinton 2008), Figure 1 visualizes the user's top four favorite clusters derived by the eigenvectors (frequencies) with largest frequency signals (Shi and Malik 2000). We can see that the movies are well classified, where we can discern animation (blue), comedy romance (green), thriller (red), action comedy movies (yellow).

We recall that each frequency corresponds to a specific set of items, and each frequency component reflects how much this user likes these items. Hence, Figure 2 shows how user preferences vary along one's behavior sequence, where a 10-size sliding window with step size 3 is used. In concrete, this user has long-term interests in the items with frequencies λ_0 and λ_2, and the interests related to the frequency λ_{13} (λ_4) emerged (disappeared) since 6^{th} position. This study consists with the dynamic nature of user preferences.

Conclusion

This paper introduces a graph signal processing formulation to 1-bit matrix completion and proposes the SGMC algorithm to leverage the flexible signal operations on graphs. SGMC can not only recover the underlying 0-1 matrix but also provides micro- and macro-level explanations behind the predictions. To scale to very large datasets, we also develop efficient and scalable approximations for the SGMC algorithm. Empirical results across multiple datasets validate the superior accuracy, efficiency and interpretability of the proposed method in top-N recommendation tasks.

Acknowledgment

This research was partially supported by China Major State Research Development Program (2020AAA0107600), NSFC (U19B2035, 61972250, 72061127003).

Ethics Statement

The explanations outputted by SGMC can reflect the rich information of individual human being, which can incur the privacy issue. Therefore when deploying such technologies, we shall keep in mind the potential privacy risk as well as other potential issues it may bring about.

References

Abdollahi, B.; and Nasraoui, O. 2016. Explainable matrix factorization for collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, 5–6.

Bennett, J.; Lanning, S.; et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, 35. Citeseer.

Bertin-Mahieux, T.; Ellis, D. P.; Whitman, B.; and Lamere, P. 2011. The Million Song Dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Cai, T.; and Zhou, W.-X. 2013. A max-norm constrained minimization approach to 1-bit matrix completion. *The Journal of Machine Learning Research* 14(1): 3619–3647.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, 129–136.

Chen, C.; Li, D.; Zhao, Y.; Lv, Q.; and Shang, L. 2015. WEMAREC: Accurate and scalable recommendation through weighted and ensemble matrix approximation. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 303–312.

Chen, W.-Y.; Song, Y.; Bai, H.; Lin, C.-J.; and Chang, E. Y. 2010. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence* 33(3): 568–586.

Davenport, M. A.; Plan, Y.; Van Den Berg, E.; and Wootters, M. 2014. 1-bit matrix completion. *Information and Inference: A Journal of the IMA* 3(3): 189–223.

Deshpande, M.; and Karypis, G. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22(1): 143–177.

Du Plessis, M.; Niu, G.; and Sugiyama, M. 2015. Convex formulation for learning from positive and unlabeled data. In *International Conference on Machine Learning*, 1386–1394.

Du Plessis, M. C.; Niu, G.; and Sugiyama, M. 2014. Analysis of learning from positive and unlabeled data. In *Advances in neural information processing systems*, 703–711.

Fowlkes, C.; Belongie, S.; Chung, F.; and Malik, J. 2004. Spectral grouping using the Nystrom method. *IEEE transactions on pattern analysis and machine intelligence* 26(2): 214–225.

Halko, N.; Martinsson, P.-G.; and Tropp, J. A. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53(2): 217–288.

Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5(4): 1–19.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182.

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*, 549–558.

Hsieh, C.-J.; Natarajan, N.; and Dhillon, I. S. 2015. PU Learning for Matrix Completion. In *ICML*, 2445–2453.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, 263–272. Ieee.

Jahrer, M.; and Töscher, A. 2012. Collaborative filtering ensemble for ranking. In *Proceedings of KDD Cup 2011*, 153–167.

Kiryo, R.; Niu, G.; du Plessis, M. C.; and Sugiyama, M. 2017. Positive-unlabeled learning with non-negative risk estimator. In *Advances in neural information processing systems*, 1675–1685.

Kumar, S.; Mohri, M.; and Talwalkar, A. 2009. Ensemble nystrom method. In *Advances in Neural Information Processing Systems*, 1060–1068.

Lee, J.; Kim, S.; Lebanon, G.; and Singer, Y. 2013. Local low-rank matrix approximation. In *International conference on machine learning*, 82–90.

Lehoucq, R. B.; Sorensen, D. C.; and Yang, C. 1998. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6. Siam.

Li, D.; Chen, C.; Lv, Q.; Yan, J.; Shang, L.; and Chu, S. 2016. Low-rank matrix approximation with stability. In *ICML*, 295–303.

Li, M.; Kwok, J. T.-Y.; and Lü, B. 2010. Making large-scale Nyström approximation possible. In *ICML 2010-Proceedings, 27th International Conference on Machine Learning*, 631.

Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; and Sun, G. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1754–1763.

Liang, D.; Krishnan, R. G.; Hoffman, M. D.; and Jebara, T. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, 689–698.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9(Nov): 2579–2605.

Mackey, L. W.; Jordan, M. I.; and Talwalkar, A. 2011. Divide-and-conquer matrix factorization. In *Advances in neural information processing systems*, 1134–1142.

Natarajan, N.; Dhillon, I. S.; Ravikumar, P. K.; and Tewari, A. 2013. Learning with noisy labels. In *Advances in neural information processing systems*, 1196–1204.

Ning, X.; and Karypis, G. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*, 497–506. IEEE.

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

Park, D.; Neeman, J.; Zhang, J.; Sanghavi, S.; and Dhillon, I. 2015. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *International Conference on Machine Learning*, 1907–1916.

Rendle, S. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*, 995–1000. IEEE.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, 452–461. AUAI Press.

Shi, J.; and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22(8): 888–905.

Shuman, D. I.; Narang, S. K.; Frossard, P.; Ortega, A.; and Vandergheynst, P. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine* 30(3): 83–98.

Shuman, D. I.; Ricaud, B.; and Vandergheynst, P. 2016. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis* 40(2): 260–291.

Steck, H. 2011. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*, 125–132.

Steck, H. 2019. Markov Random Fields for Collaborative Filtering. In *Advances in Neural Information Processing Systems*, 5474–5485.

Wang, H.; Wang, N.; and Yeung, D.-Y. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1235–1244.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 165–174.

Williams, C. K.; and Seeger, M. 2001. Using the Nyström method to speed up kernel machines. In *Advances in neural information processing systems*, 682–688.

Wu, L.; Hsieh, C.-J.; and Sharpnack, J. 2017. Large-scale collaborative ranking in near-linear time. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 515–524.

Wu, L.; Hsieh, C.-J.; and Sharpnack, J. 2018. Sql-rank: A listwise approach to collaborative ranking. *arXiv preprint arXiv:1803.00114* .

Zhang, Y.; and Chen, X. 2018. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192* .

Zheng, L.; Lu, C.-T.; Jiang, F.; Zhang, J.; and Yu, P. S. 2018. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 311–319.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1059–1068.

Zou, H.; and Hastie, T. 2005. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)* 67(2): 301–320.