Understanding and Improving Relational Matrix Factorization in Recommender Systems

Li Pu
Artificial Intelligence Laboratory, EPFL
INR240, EPFL-IC-LIA, Station 14
CH-1015, Lausanne, Switzerland
li.pu@epfl.ch

Boi Faltings
Artificial Intelligence Laboratory, EPFL
INR230, EPFL-IC-LIA, Station 14
CH-1015, Lausanne, Switzerland
boi.faltings@epfl.ch

ABSTRACT

Matrix factorization techniques such as the singular value decomposition (SVD) have had great success in recommender systems. We present a new perspective of SVD for constructing a latent space from the training data, which is justified by the theory of hypergraph model. We show that the vectors representing the items in the latent space can be grouped into (approximately) orthogonal clusters which correspond to the vertex clusters in the co-rating hypergraph, and the lengths of the vectors are indicators of the representativeness of the items. These properties are used for making top-N recommendations in a two-phase algorithm. In this work, we provide a new explanation for the significantly better performance of the asymmetric SVD approaches and a novel algorithm for better diversity in top-N recommendations.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*

Keywords

recommender system, matrix factorization, hypergraph

1. INTRODUCTION

In a recommender system, there are two types of entities, i.e. users and items. The users give weighted connections, or ratings, to the items. These connections can be modeled as a rating matrix $\boldsymbol{R}$ where the rows represent the users and the columns represent the items. The entry $\boldsymbol{R}(i,k)$ is the rating given by user i to item k (if the rating exists). The goal of the recommender system is to predict unseen items that might be rated with high scores. A common assumption for making predictions is that certain latent factors are associated with each user and each item, and a user gives ratings according to her latent factors and the latent factors of the items.

RecSys'13, October 12–16, 2013, Hong Kong, China.

http://dx.doi.org/10.1145/2507157.2507178.

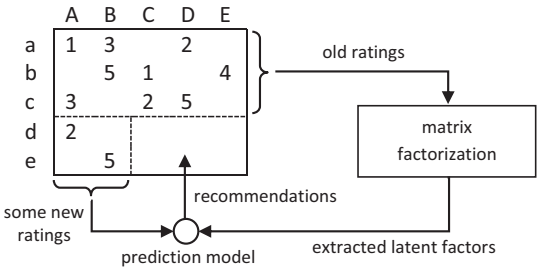

Figure 1: Recommender system using matrix factorizations.

For example, in the `Lastfm` dataset, the items are the artists in a music website. The latent factors could be the music genres. The recommender system first obtains the genres of the artists, and the users' interests on different genres. Then the system makes recommendations by aligning the user's interests to the genres of the artists.

For ratings modeled as the matrix $\boldsymbol{R}$, various matrix factorization techniques have been applied to extract the latent factors and make recommendations. Figure 1 shows a typical recommender system using matrix factorization. The old ratings given by existing users are first decomposed into multiple matrices to extract the latent factors. When new users come into the system, their ratings are combined with the extracted latent factors to make recommendations.

Sarwar et al. [17] proposed one of the early works that use the low-rank Singular Value Decomposition (SVD) approximation to replace the missing values in the original rating matrix, i.e. $\boldsymbol{R} \approx \boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{V}_l^\top$ where the sizes of the matrices are $\boldsymbol{R}$: $m \times n$, $\boldsymbol{U}_l$: $m \times l$, $\boldsymbol{S}_l$: $l \times l$, and $\boldsymbol{V}_l$: $n \times l$. The diagonal matrix $\boldsymbol{S}_l$ contains the l biggest singular values of $\boldsymbol{R}$, and this low-rank approximation is also called the *truncated SVD*. This approach relies on the fact that the rank-l truncated SVD approximation $\hat{\boldsymbol{R}} = \boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{V}_l^\top$ minimizes the Frobenius norm $\|\hat{\boldsymbol{R}} - \boldsymbol{R}\|_F$ among all the rank-l matrices, and we hope that the matrix $\hat{\boldsymbol{R}}$ would generalize the correct ratings to the missing entries in $\boldsymbol{R}$. Note that imputation of missing values is implicitly required when computing the truncated SVD. We can represent user i with the vector $\bar{\boldsymbol{u}}_i = \boldsymbol{S}_l^{-1/2} \boldsymbol{U}_l(i,\cdot)^\top$ and item k with the vector $\bar{\boldsymbol{v}}_k = \boldsymbol{S}_l^{-1/2} \boldsymbol{V}_l(i,\cdot)^\top$. The predicted rating from user i to item k is then $\hat{\boldsymbol{R}}(i,k) = \bar{\boldsymbol{u}}_i^\top \bar{\boldsymbol{v}}_k$, where $\bar{\boldsymbol{u}}_i$ and $\bar{\boldsymbol{v}}_k$ can be considered as latent vectors.

During the Netflix prize competition, Funk [5] proposed an iterative way to compute the low-rank SVD approxima-

tion on a very big matrix that minimizes only the Frobenius norm on the non-empty entries in the original rating matrix. Based on Funk's work, Paterek [14] suggested a new model to reduce the number of parameters that need to be stored in the recommender system. In Paterek's approach, a truncated SVD is computed as before, but only the right singular vectors in $\boldsymbol{V}_l$ are taken into the second step. In the second step, item k is represented by $\boldsymbol{v}_k$ which is the k-th column of $\boldsymbol{V}_l^\top$, while user i's latent vector is computed by a linear combination of the $\boldsymbol{v}_k$'s whose corresponding items are rated by user i. Finally the rating prediction is the inner product of a user's latent vector and an item's latent vector. Koren [9] pointed out that this "asymmetric SVD" which takes only the right singular vectors actually works better in prediction accuracy. However, besides the empirical studies, the reason of such improvement for asymmetric SVD is not clear.

The **main contributions** of this paper include **(1)** a new explanation of the (approximate) orthogonal structure of the latent space constructed from the asymmetric SVD, which is used as an intermediate step in a top-N recommender system, and **(2)** a novel algorithm based on a hypergraph formulation that brings more diversity in the recommendations by applying normalizations in the asymmetric SVD.

In the reminder of the paper, we start with the hypergraph representation and the normalized hypergraph cut. We show that the asymmetric SVD approach follows the hypergraph learning scheme, and there is an (approximate) orthogonal structure associated with the right singular vectors. Based on the lengths of the latent vectors obtained from the right singular vectors, the items can be categorized into anchor items and non-anchor items. Finally we present a top-N recommendation algorithm that utilizes the above structures, and conduct experiments to compare our algorithm with the state-of-the-art approaches.

2. PRELIMINARIES

Suppose we have two sets of entities: the user set $Y = \{y_1, y_2, ..., y_m\}$ of size m, and the item set $Z = \{z_1, z_2, ..., z_n\}$ of size n. The items in set Z belong to s clusters $\{C_1, C_2, ..., C_s\}$. Each cluster C_j is a subset of Z and $C_j \cap C_{j'} = \emptyset$ for $j \neq j'$.

Taking the `Lastfm` dataset as an example again, let the set Y be the set of users and the set Z be the set of artists. Clusters or partitions over Z can be established according to the genres of the artists. For example, C_1 includes the rock music artists, while all the country music artists are in C_2. Each user in the set Y, on the other hand, is assumed to have a preference over music genres. If the user likes rock music, she would listen to the artists from C_1 with high probability. If the user does not like country music, she would rarely listen to the artists from C_2.

The mutually exclusion between clusters of items is the basic assumption in our analysis. Each cluster can be considered as a latent factor by which the ratings are generated. In some cases, however, an artist could play in multiple genres. This does not pose a problem to our basic assumption, because we can create a cluster (latent factor) that corresponds to a combination of genres. In fact, if the intuition behind the latent factors follows some categorical attributes, e.g. genres, languages, professions, etc., or a combination of categorical attributes, our basic assumption always holds.

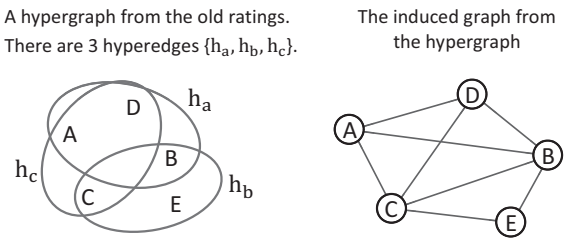

Figure 2: The hypergraph representation of the ratings in Figure 1 and the induced graph.

2.1 From Ratings to Relations

The matrix $\boldsymbol{R}$ contains the ratings between the users and the items, which can be seen as relations between two sets. For simplicity, we assume that all the ratings are bigger than zero, thus the empty entries in the $m \times n$ rating matrix $\boldsymbol{R}$ can be represented by zeros. If we ignore the actual ratings in $\boldsymbol{R}$, we can obtain a binary matrix $\boldsymbol{X} = (\boldsymbol{R} > 0)$ that represents only the relations, i.e. $\boldsymbol{X}(i,k) = 1$ if $\boldsymbol{R}(i,k) > 0$, otherwise $\boldsymbol{X}(i,k) = 0$, and the resulting matrix $\boldsymbol{X}$ can be precisely represented by a hypergraph.

A *hypergraph* is an extension of a graph for modeling relations. In a graph, an edge connects exactly two vertices, while in a hypergraph a *hyperedge* connects any number of vertices. For a binary relation matrix $\boldsymbol{X}$, the items are considered as the vertices of a hypergraph, and a user is represented by a hyperedge which contains all the items (vertices) that have been rated by the user. Following the tradition of visualizing a hypergraph in literatures, Figure 2 shows an example of the hypergraph representation. The matrix $\boldsymbol{X}$ is also called the *incident matrix* of the hypergraph. The *vertex degree* (of an item) is the sum of the k-th column of $\boldsymbol{X}$, i.e. $\deg(z_k) = \sum_i \boldsymbol{X}(i,k)$. The *hyperedge degree* (of a user) is the sum of the i-th row of $\boldsymbol{X}$, i.e. $\deg(y_i) = \sum_k \boldsymbol{X}(i,k)$.

Recall that we assume there are some clusters of items, and these clusters correspond to the latent factors. Thus extracting the latent factors is equivalent to finding clusters of items (vertices) in the hypergraph. There is a wide range of studies for clustering the vertices in a hypergraph [8, 1, 22]. In this paper, we focus on one particular line of studies that convert a hypergraph into an approximated graph so that the graph clustering algorithms can be applied to the hypergraph. Zhou et al. proposed a transformation to convert a hypergraph into a graph based on the *Normalized Hypergraph Cut* (NHC) [22] (others include the clique-expansion, the start-expansion [1], and the hyperedge-expansion [15]). We denote the transformation proposed in [22] as "NHC", and use it to find clusters in the hypergraph.

By applying the NHC transformation, the vertices of the hypergraph are copied into a new graph. Then each hyperedge is converted into a clique of weighted edges in the new graph that connect all the vertices in the original hyperedge. The weights of the edges in the clique are normalized by the hyperedge degree so that the contribution of each hyperedge to the new graph is the same. With a linear combination of all the cliques from all the hyperedges, the new graph can be seen as an approximation of the hypergraph in the sense that any two vertices connected by some hyperedges are also connected in the new graph. We call this new graph the *induced graph* (see Figure 2).

One can show that the adjacency matrix of the induced graph is $\boldsymbol{X}^\top \boldsymbol{D}_e^{-1} \boldsymbol{X}$, where $\boldsymbol{X}^\top$ is the transpose of $\boldsymbol{X}$ and $\boldsymbol{D}_e = \text{diag}(\boldsymbol{X}\boldsymbol{1})$ is the hyperedge degree matrix. $\boldsymbol{D}_e$ is actually the diagonal matrix of row sums of $\boldsymbol{X}$, and $\boldsymbol{1}$ is an all-ones vector of proper length. With the NHC transformation, it is proposed to use the normalized Laplacian of the induced graph to find clusters of vertices [22], where the normalized Laplacian matrix $\boldsymbol{L}$ is defined as

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}_v^{-1/2} \boldsymbol{X}^\top \boldsymbol{D}_e^{-1} \boldsymbol{X} \boldsymbol{D}_v^{-1/2}. \quad (1)$$

Matrix $\boldsymbol{I}$ is an identity matrix, and $\boldsymbol{D}_v = \text{diag}\left(\boldsymbol{1}^\top \boldsymbol{X}\right)$ is the vertex degree matrix.

As in standard spectral graph theory [4], the clustering algorithm involves computing the leading (smallest) l eigenvalues $\{\lambda_1, \lambda_2, ..., \lambda_l\}$ of $\boldsymbol{L}$ and the corresponding eigenvectors $[\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_l] = \boldsymbol{F}$. Then a vertex (or an item z_k) can be represented by a vector $\boldsymbol{\alpha}_k$ that is the k-th column of $\boldsymbol{F}^\top$ (note that $\boldsymbol{F}$ is of size $n \times l$). We call the l-dimensional vector $\boldsymbol{\alpha}_k$ the *latent vector*, and the l-dimensional space the *latent space*. Once the latent vectors are computed, one can apply any clustering algorithm (such as k-means) in the latent space to find clusters of vertices.

This scheme of spectral technique (the list of eigenvalues are often called the *spectrum* of the matrix) has been widely used in many areas. It has close connections to the Principal Component Analysis (PCA), SVD, and many other matrix factorization techniques. In the next section, we show that the NHC transformation of a hypergraph is equivalent to a SVD, and under certain conditions there is a structure with the resulting latent vectors. Note that although the actual ratings are ignored in the hypergraph representation, they will be used again in the top-N recommendation algorithm presented in Section 4.

3. HYPERGRAPH LEARNING USING SVD

The normalized Laplacian $\boldsymbol{L}$ is a symmetric matrix. Let

$$\bar{\boldsymbol{X}} = \boldsymbol{D}_e^{-1/2} \boldsymbol{X} \boldsymbol{D}_v^{-1/2}. \quad (2)$$

We can rewrite $\boldsymbol{L} = \boldsymbol{I} - \bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}}$. The biggest l eigenvalues of $\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}}$ are exactly $\{1-\lambda_1, 1-\lambda_2, ..., 1-\lambda_l\}$, while the corresponding eigenvectors are still $[\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_l] = \boldsymbol{F}$. Then we decompose the matrix $\bar{\boldsymbol{X}}$ by the full SVD $\bar{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices, and $\boldsymbol{S}$ is a rectangular diagonal matrix. The biggest l singular values in $\boldsymbol{S}$ are exactly $\{\sqrt{1-\lambda_1}, \sqrt{1-\lambda_2}, ..., \sqrt{1-\lambda_l}\}$, and the columns of $\boldsymbol{V}$ (right-singular vectors) are the eigenvectors of $\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}}$. Therefore, instead of computing the eigen-decomposition of $\boldsymbol{L}$, we can obtain $\boldsymbol{F}$ from the SVD of $\bar{\boldsymbol{X}}$, i.e. $\boldsymbol{F}$ is the submatrix of the first l columns of $\boldsymbol{V}$.

Since we are looking for the right singular vectors of $\bar{\boldsymbol{X}}$ associated with the largest l singular values, computing $\boldsymbol{F}$ can be done by the truncated SVD $\bar{\boldsymbol{X}} \approx \boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{V}_l^\top = \boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{F}^\top$. Thus the NHC transformation is equivalent to the asymmetric truncated SVD of $\bar{\boldsymbol{X}}$. We call the column vectors in $\bar{\boldsymbol{X}}$ the "profiles" of the items.

There are many advantages of using SVD to compute the $\boldsymbol{\alpha}_k$'s (latent vectors) rather than the eigen-decomposition of $\boldsymbol{L}$. Firstly, the matrix $\bar{\boldsymbol{X}}$ is usually sparse, but $\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}}$ might be non-sparse. When n is large, the computational cost and storage cost of the eigen-decomposition might be impractical. Secondly, there are existing approaches to implement SVD incrementally, which allows us to just compute the minor changes when modifying $\bar{\boldsymbol{X}}$.

3.1 A Special Case

We first consider a special case where all the items in one cluster are rated by the same set of users. In other words, each cluster C_j is associated with a subset of users and these users have rated all the items in C_j. We call this special case the "full concentrated case" because it can be modeled with a beta distribution with a small concentration parameter. If there are s clusters, the matrix $\bar{\boldsymbol{X}}$ can be written as

$$\bar{\boldsymbol{X}} = [\underbrace{\bar{\boldsymbol{x}}_1 \cdots \bar{\boldsymbol{x}}_1}_{|C_1| \text{ vectors}} \underbrace{\bar{\boldsymbol{x}}_2 \cdots \bar{\boldsymbol{x}}_2}_{|C_2| \text{ vectors}} \cdots \underbrace{\bar{\boldsymbol{x}}_s \cdots \bar{\boldsymbol{x}}_s}_{|C_s| \text{ vectors}}], \quad (3)$$

where the column vectors in one cluster are all the same. We call the items in the s clusters the *anchor items* because they define the cluster centers and remain clustered in the latent space.

In spectral graph theory, if the graph contains s disconnected components, one can show that the latent vectors obtained from the first s eigenvectors of the graph Laplacian are aligned with the axes of the s-dimensional latent space [13]. When the disconnected components are connected by some weak links, the latent vectors are still aligned with s approximately orthogonal lines [20]. However, the induced graph of a hypergraph usually does not consist of more than one connected component. Figure 3 shows an example of a hypergraph in full concentrated case. The adjacency matrix of the induced graph is not a diagonal block matrix (not even approximately), so we need to develop a new analysis to support the clustering algorithm for hypergraphs.

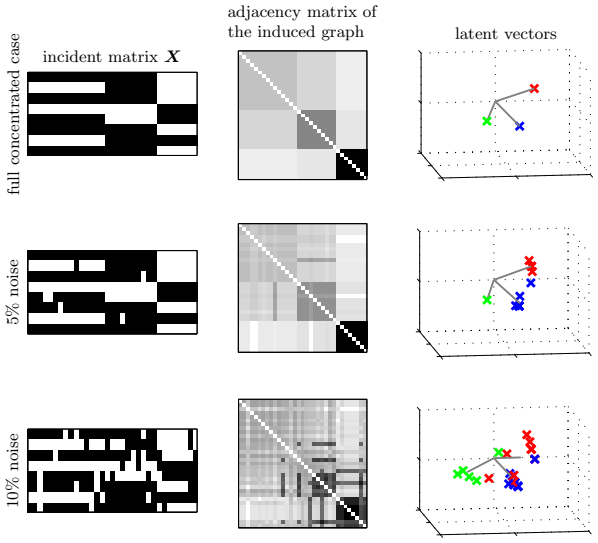

Figure 3: Examples of the incident matrix and the latent vectors with the full concentrated case and noisy cases. The noises are added by randomly flipping some entries of $\boldsymbol{X}$. Approximately orthogonal structures can be observed in the noisy cases. In the charts of latent vectors, the gray lines are from the origin to the cluster centers.

If the rank of $\bar{\boldsymbol{X}}$ is s, i.e. the anchor item profiles $\bar{\boldsymbol{x}}_j$'s are linearly independent, the truncated SVD $\bar{\boldsymbol{X}} = \boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{V}_l^\top$ is an exact decomposition when $l = s$. Recall that the latent vectors in $\boldsymbol{F}$ can be obtained from $\boldsymbol{F} = \boldsymbol{V}_l$. The following statement shows the structure of $\boldsymbol{F}$.

PROPOSITION 1. *If $\bar{\boldsymbol{X}}$ is in the form of Eq. (3) and $\bar{\boldsymbol{x}}_j$'s are linearly independent, the latent vectors $\boldsymbol{F} = [\boldsymbol{\alpha}_1 \cdots \boldsymbol{\alpha}_n]^\top$ given by the truncated SVD $\bar{\boldsymbol{X}} = \boldsymbol{U}_s \boldsymbol{S}_s \boldsymbol{F}^\top$ (columns of $\boldsymbol{F}$ are normalized to 1) can be grouped by the clusters $\boldsymbol{\alpha}_k = \boldsymbol{\beta}_j$ for $\forall z_k \in C_j$. The $\boldsymbol{\alpha}_k$'s from the same cluster are identical (denoted as $\boldsymbol{\beta}_j$). Furthermore, we have $\boldsymbol{\beta}_j^\top \boldsymbol{\beta}_{j'} = 0$ for $\forall j \neq j'$, and $\boldsymbol{\beta}_j^\top \boldsymbol{\beta}_j = |C_j|^{-1}$ for $\forall j \in \{1, 2, ..., s\}$. (See appendix for proof.)*

If all the items are anchor items, i.e. in the form of Eq. (3) and having linearly independent $\bar{\boldsymbol{x}}_j$'s, the latent vectors would be in exactly s clusters whose centers are orthogonal to each other (see Figure 3). The length of the latent vector is the reciprocal of the cluster size. Unlike the graph case, the latent vectors computed from a hypergraph are usually not aligned with the axes of the latent space, but up to a rotation that depends on the cluster sizes and the distribution of non-zero entries in $\boldsymbol{X}$. Nevertheless, the orthogonal structure of the latent vectors still provides us a similar theoretical foundation for clustering vertices as in the standard spectral graph theory.

3.2 More General Case

The assumption that all the items are anchor items is too strict. We usually have some item profiles which are combinations of some other item profiles. For example, there are pure "SciFi" books and pure "romance" books that attract distinct sets of readers (the anchor items). But one book might have the "SciFi" element and the "romance" element at the same time, so it could attract readers from both sets. The item profile of this book is apparently a combination of two profiles.

Besides the anchor items, we now consider the *non-anchor items* that are combinations of anchor items. We assume that the weights of the combinations are always non-negative, which implies that only additions of profiles are allowed, but not subtractions. If there are n' non-anchor items and s clusters of anchor items, $\bar{\boldsymbol{X}}$ can be rewritten as

$$\bar{\boldsymbol{X}} = [\underbrace{\bar{\boldsymbol{x}}'_1 \cdots\cdots \bar{\boldsymbol{x}}'_{n'}}_{n' \text{ vectors}} \underbrace{\bar{\boldsymbol{x}}_1 \cdots \bar{\boldsymbol{x}}_1}_{|C_1| \text{ vectors}} \underbrace{\bar{\boldsymbol{x}}_2 \cdots \bar{\boldsymbol{x}}_2}_{|C_2| \text{ vectors}} \cdots \underbrace{\bar{\boldsymbol{x}}_s \cdots \bar{\boldsymbol{x}}_s}_{|C_s| \text{ vectors}}], \quad (4)$$

where the profile of non-anchor item z'_k is denoted as $\bar{\boldsymbol{x}}'_k$ ($k \in \{1, 2, ..., n'\}$), and the profile of an anchor item from cluster C_j is still denoted as $\bar{\boldsymbol{x}}_j$. Note that $\bar{\boldsymbol{x}}'_k$ is not an exact linear combination of the anchor items, because when combining the profiles of the anchor items, there might be redundant entries. For example, if $\bar{\boldsymbol{x}}_1 = [1\,1\,0\,0]^\top$, $\bar{\boldsymbol{x}}_2 = [0\,1\,1\,0]^\top$ and $\bar{\boldsymbol{x}}'_k$ is a combination of $\bar{\boldsymbol{x}}_1$ and $\bar{\boldsymbol{x}}_2$, $\bar{\boldsymbol{x}}'_k$ would be $[1\,1\,1\,0]^\top$ instead of $[1\,2\,1\,0]^\top$ because the incident matrix $\boldsymbol{X}$ is always a binary matrix (normalizations are ignored in this example for simplicity). Thus we need to subtract the redundant entries when summing the anchor item profiles:

$$\bar{\boldsymbol{x}}'_k = \sum_{j \in I(z'_k)} \sqrt{\frac{\deg(z_j)}{\deg(z'_k)}} \bar{\boldsymbol{x}}_j - \boldsymbol{r}, \quad (5)$$

where $I(z'_k)$ is the set of anchor items from which z'_k is constructed, and $\boldsymbol{r}$ denotes the column vector that consists of the redundant entries in the profiles of the anchor items. Obviously all entries of $\boldsymbol{r}$ are non-negative.

Denote the latent vector of an anchor item from cluster C_j as $\boldsymbol{\beta}_j$, and the latent vector of a non-anchor item z'_k as $\boldsymbol{\beta}'_k$. By the SVD on $\bar{\boldsymbol{X}}$, $\boldsymbol{\beta}'_k$ can be expressed by the combination

$$\boldsymbol{\beta}'_k = \sum_{j \in I(z'_k)} \sqrt{\frac{\deg(z_j)}{\deg(z'_k)}} \boldsymbol{\beta}_j - \boldsymbol{S}_l^{-1} \boldsymbol{U}^\top \boldsymbol{r}. \quad (6)$$

If the entries in $\boldsymbol{r}$ are small enough to be considered as residuals, or in other words the anchor items are associated with disjoint sets of users, we could still expect approximately orthogonal structure between the latent vectors of the anchor items.

PROPOSITION 2. *If the anchor items are associated with disjoint sets of users, i.e. $\bar{\boldsymbol{x}}_j^\top \bar{\boldsymbol{x}}_{j'} = 0$ for $\forall j, j' \in \{1, 2, ..., s\}$, $j \neq j'$, the latent vectors of the anchor items are approximately orthogonal $\boldsymbol{\beta}_j^\top \boldsymbol{\beta}_{j'} \approx 0$ for $\forall j \neq j'$.*

The conclusion of Proposition 2 follows from the fact that $\bar{\boldsymbol{x}}_j^\top \bar{\boldsymbol{x}}_{j'} = \boldsymbol{\beta}_j^\top \boldsymbol{S}_l^2 \boldsymbol{\beta}_{j'} = 0$, and the first l singular values in $\boldsymbol{S}_l$ are approximately the same when the cluster sizes are similar.

In practice, we can consider the anchor items as the most representative items of each cluster. For example, the pure "SciFi" books are the anchor items of the "SciFi" cluster, because they have the most unique features that define what is "SciFi". The anchor items are not necessarily the most popular items or the most informative items in each cluster, but the items that are most distant from all the other clusters and attract a very unique group of users. Usually the anchor items are in the long-tail part, i.e. less popular.

Although the anchor items exhibit the approximately orthogonal structure, we do not know which items are the anchor items. The following result suggests that it is possible to distinguish the anchor items and the non-anchor items by examining the length of the latent vectors.

PROPOSITION 3. *For a non-anchor item z'_k, denote $I(z'_k)$ the set of anchor items from which z'_k is constructed. Suppose that the anchor items in $I(z'_k)$ are non-overlapping, i.e. $\bar{\boldsymbol{x}}_j^\top \bar{\boldsymbol{x}}_{j'} = 0$ for $\forall j, j' \in I(z'_k), j \neq j'$, and the cluster sizes of the anchor items are similar, i.e. the lengths of the anchor latent vectors are similar (denoted as $\|\boldsymbol{\beta}_j\|_2 \approx b_0$ for $\forall j \in I(z'_k)$, where $\|\cdot\|_2$ is the l_2 norm). We have $\|\boldsymbol{\beta}'_k\|_2 < b_0$. (See appendix for proof.)*

This result states that the length of a non-anchor latent vector is smaller than the lengths of the anchor latent vectors from which the former is constructed. Therefore, we can find the anchor items by sorting the lengths of the latent vectors in descending order.

In order to verify the conclusions in Propositions 2 and 3, we compute the latent vectors by the rank-l truncated SVD of $\bar{\boldsymbol{X}}$, and select $10 \cdot l$ items whose latent vectors have the longest lengths. Then the selected latent vectors are grouped into clusters by the k-means algorithm. By the conclusions of the Propositions, the cluster centers of the selected latent vectors should be approximately orthogonal. In Figure 4, we show the results from a real dataset with NHC and SVD. Because there is no hyperedge-degree and vertex-degree normalizations in SVD, the longest latent vectors are not necessarily the anchor items. We can observe that there exists an approximately orthogonal structure in the NHC results, while such structure is less clear with SVD.

There are existing works about the categorization of the items or users based on their impact on the recommender

system, e.g. [12]. Our classification of anchor and non-anchor items provides a new perspective in the context of latent space. In summary, **(1)** if the latent vectors are all orthogonal, each corresponds to a different anchor item (the full concentrated case). **(2)** if they are not, there is an orthogonal subset such that each latent vector in this subset corresponds to a different anchor item, and all the others (non-anchor items) are approximately in the subspace spanned by the orthogonal subset. **(3)** generally, the anchor items are rated by different groups of users.

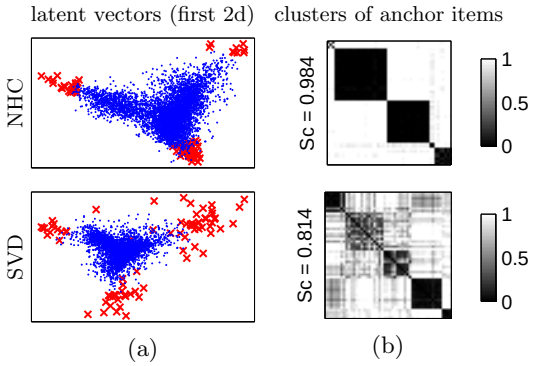

Figure 4: The selected anchor items with longer latent vector lengths in the dataset `BookCrossing` (see Section 6 for the description of the dataset). (a) The anchor items are shown in red x-markers. (b) The cosine distance matrix of the clustered anchor items. Sc is the sum of average distances between clusters minus the sum of average distances within clusters, normalized by the largest possible value with a perfect orthogonal structure. A Sc value close to 1 suggests a better orthogonal structure.

4. THE NORMALIZED RELATIONAL SVD

We now present a new algorithm based on the NHC Laplacian and the above analysis. Once the latent vectors are computed for the items, we model the user y_i's interests (or latent profile) as another vector $\boldsymbol{\theta}_i$ in the same latent space. The predicted score for user y_i on item z_k is $P(y_i, z_k) = \boldsymbol{\theta}_i^\top \boldsymbol{\alpha}_k$, and the vector $\boldsymbol{\theta}_i$ should take the values such that the predictor P coincides with the existing ratings (or as close as possible).

In order to simulate the scenario where the recommendations are made for new users and evaluate the algorithm, we split the rating matrix $\boldsymbol{R}$ into three parts

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_m \\ \boldsymbol{R}_{tr}, \boldsymbol{R}_{te} \end{bmatrix} \begin{matrix} \}\text{subset } Y_m \\ \}\text{subset } Y_t \end{matrix} . \tag{7}$$

Firstly the users are separated into two subsets. Y_m is the subset of old users, and Y_t is the subset of new users. The sub-matrix $\boldsymbol{R}_m$ contains all the ratings from the users in Y_m, and the ratings in $\boldsymbol{R}_m$ are used for constructing the hypergraph and computing the latent vectors. The ratings from the users in Y_t are further split into two parts: $\boldsymbol{R}_{tr}$ and $\boldsymbol{R}_{te}$. $\boldsymbol{R}_{tr}$ contains the "known" ratings of the new users. Based on $\boldsymbol{R}_{tr}$ and the latent vectors, we can predict or recommend more items for the users in Y_t. Then the predictions are evaluated by the ratings in $\boldsymbol{R}_{te}$. Note that only $\boldsymbol{R}_m$ and $\boldsymbol{R}_{tr}$ are taken as the inputs of the algorithm. $\boldsymbol{R}_{te}$ is used for evaluation.

Since our algorithm works with the hypergraph transformation and SVD, we call it `HSVD`, which contains the following steps:

Step 1: Compute the l-dimensional NHC transformation $\boldsymbol{F}$ by the truncated SVD $\bar{\boldsymbol{X}}_m \approx \boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{F}^\top$, where $\bar{\boldsymbol{X}}_m$ is computed from the binary matrix $\boldsymbol{X}_m = (\boldsymbol{R}_m > 0)$ (see Eq. (2)). Normalize each of the l columns of $\boldsymbol{F}$ to 1 (l_2 norm).

Step 2: For each user $y_i \in Y_t$, we choose the $\boldsymbol{\theta}_i$ that minimizes the error $\|\boldsymbol{R}_{tr}(i,\cdot) - \boldsymbol{\theta}_i^\top \boldsymbol{F}^\top\|_2^2$. The predicted score vector is $\hat{\boldsymbol{r}}_i = \boldsymbol{\theta}_i^\top \boldsymbol{F}^\top$. In matrix form, the full prediction matrix is $\hat{\boldsymbol{R}}_{te} = \boldsymbol{\Theta}^\top \boldsymbol{F}^\top$, where $\boldsymbol{\Theta}$ is obtained by solving the linear system $\boldsymbol{F}\boldsymbol{\Theta} = \boldsymbol{R}_{tr}^\top$ in a least-squares sense.

Step 3: For each user $y_i \in Y_t$, recommend the top-N unrated items with the highest scores in $\hat{\boldsymbol{R}}_{te}$.

There are several normalizations in the `HSVD` algorithm which did not exist in previous matrix factorization methods. When computing $\bar{\boldsymbol{X}}_m$ in step 1, the hyperedge-degree normalization (from the induced graph transformation itself) and the vertex-degree normalization (from the normalized Laplacian of the induced graph) ensure that the anchor latent vectors would have longer lengths. By Proposition 1, the normalization of the columns of $\boldsymbol{F}$ in step 1 is also indispensable for producing an (approximately) orthogonal structure in the latent space.

When computing the $\boldsymbol{\theta}_i$ in step 2, two parts are actually taken into consideration, which is illustrated in Figure 5. Firstly we consider the latent vectors of the items which are rated by user y_i (the triangle points in Figure 5). We would like to choose a $\boldsymbol{\theta}_i$ that is close to these rated $\boldsymbol{\alpha}_k$'s. Since the row vector $\boldsymbol{R}(i,\cdot)$ is weighted by the ratings, $\boldsymbol{\theta}_i$ should be even closer to the $\boldsymbol{\alpha}_k$'s with higher ratings, which can be expressed as $\boldsymbol{\theta}_i = \arg\min_{\boldsymbol{\theta}} \sum_{k,\boldsymbol{R}(i,k)>0} \left(\boldsymbol{R}(i,k) - \boldsymbol{\theta}_i^\top \boldsymbol{\alpha}_k\right)^2$. Because every non-anchor item is approximately a linear combination of some anchor items (e.g. $\boldsymbol{\alpha}_3$ is a combination of $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_5$ in Figure 5), $\boldsymbol{\theta}_i$ should be in the subspace spanned by the latent vectors of the underlying anchor items. Secondly, for those $\boldsymbol{\alpha}_k$'s that are not rated by user y_i (the circle points in Figure 5), $\boldsymbol{\theta}_i$ should stay as far from them as possible, ideally orthogonal to these unrated latent vectors. The orthogonality between the cluster centers of the anchor items suggests that it is possible to find a $\boldsymbol{\theta}_i$ which is orthogonal to the anchor items that the user y_i is not interested in. This can be expressed as $\boldsymbol{\theta}_i = \arg\min_{\boldsymbol{\theta}} \sum_{k,\boldsymbol{R}(i,k)=0} \left(\boldsymbol{\theta}_i^\top \boldsymbol{\alpha}_k\right)^2$. Putting the two parts together, it suggests that in matrix form the $\boldsymbol{\theta}_i$'s can be obtained by solving the linear system $\boldsymbol{F}\boldsymbol{\Theta} = \boldsymbol{R}_{tr}^\top$ in a least-squares sense, which is in the step 2 of the `HSVD` algorithm.

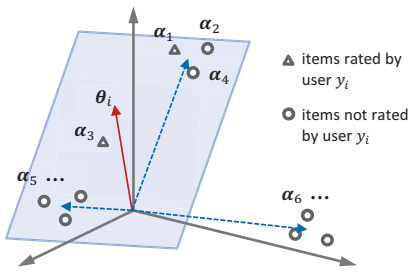

Figure 5: Illustration of the `HSVD` algorithm in 3d latent space. The dashed lines are cluster centers of the anchor items.

Proposition 3 shows that the lengths of the anchor latent vectors are longer than the non-anchor latent vectors, even if the anchor items are less popular. For a fixed $\boldsymbol{\theta}_i$, because the score $\hat{\boldsymbol{R}}_{te}(i,k)$ is simply the inner product of $\boldsymbol{\theta}_i$ and $\boldsymbol{\alpha}_k$, the anchor items of longer length would have bigger chance to be selected in the top-N recommendations. This would allow us to diversify the recommendation list and recommend more less-popular items.

4.1 Scalability

The main computational cost of our algorithm comes from the truncated SVD routine. Usually the numerical method for truncated SVD on matrix $\bar{\boldsymbol{X}}$ is in an iterative manner, and operated as an eigen-decomposition of the Hermitian matrix $\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}}$. The running time of the latter problem depends on the complexity of each iteration and the total number of iterations. In each iteration the basic operation contains two matrix vector multiplications $\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}} \boldsymbol{v}$ where $\boldsymbol{v}$ is a dense vector. If the number of non-zero entries in $\bar{\boldsymbol{X}}$ (or the number of ratings in $\boldsymbol{R}$) is M, the complexity of each iteration would be $O(M)$. On the other hand, the total number of iterations to converge depends on the gap between the l largest singular value and the $l+1$ largest singular value. A bigger gap leads to a smaller number of iterations [6, 11]. If the number of iterations is n_I, the overall complexity of the `HSVD` algorithm is $O(n_I M)$, which is linear to the number of ratings. The parameter l (dimensionality of the truncated SVD) is considered as a constant, which can be determined by cross-validation in practice.

5. RELATED WORK

In the introduction, we have mentioned the SVD [17], the SVD on non-empty entries [5], and the asymmetric SVD [14, 9] for recommender systems. Another matrix factorization technique that is widely used in recommender systems is the *non-negative matrix factorization* (NMF). In NMF, the rating matrix is approximately decomposed into two low-rank, sparse and non-negative matrices $\boldsymbol{R} \approx \boldsymbol{G}\boldsymbol{H}^\top$ such that $\|\boldsymbol{R} - \boldsymbol{G}\boldsymbol{H}^\top\|^2$ is minimized [7, 16]. The idea behind NMF is that the set of orthogonal latent factors controls the values in $\boldsymbol{R}$, and each latent factor takes one axis in the latent space. This idea is essentially similar to the `HSVD` since we have shown that the anchor items also exhibit an orthogonal structure in the latent space, although they are not aligned to the axes.

In order to compare our algorithm with the existing approaches, we adapt the SVD and the NMF into our setting, i.e. split the rating matrix into three parts as in Eq. (7) and use the sub-matrix $\boldsymbol{R}' = \left[\boldsymbol{R}_m^\top \boldsymbol{R}_{tr}^\top\right]^\top$ to compute the SVD or NMF, then evaluate the results on $\boldsymbol{R}_{te}$. The SVD and NMF algorithms in this setting are denoted as `SVDR` and `NMFR`.

To make a better baseline, we also apply the `SVDR` and `NMFR` on $\boldsymbol{R}_m$ and predict with $\boldsymbol{R}_{tr}$ in an asymmetric manner, i.e. follow the same procedures in Section 4 but replace $\bar{\boldsymbol{X}}_m$ with $\boldsymbol{R}_m$ (or replace $\boldsymbol{F}$ with the matrix $\boldsymbol{H}$ in NMF). These methods are denoted as `ASVDR` and `ANMFR`, where the leading "`A`" means "asymmetric".

We use the routine implemented by TimelyDevelopment for computing the SVD on non-empty entries [18], which is denoted as `SVDN`. Note that `SVDN` is exactly the same as `SVDR` except for the SVD computation routine.

For the asymmetric SVD, we use the approach proposed in [14]. In this approach, a decomposition is first computed by `SVDN` $\boldsymbol{R} \approx \boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{V}_l^\top$, where $\boldsymbol{V}_l = [\boldsymbol{\alpha}_1 \cdots \boldsymbol{\alpha}_n]^\top$. Then a user y_i is represented by the currently rated items: $\boldsymbol{\theta}_i = \sum_{k,\boldsymbol{R}(i,k)>0} \boldsymbol{\alpha}_k$. Finally a score is computed by the rule $\hat{\boldsymbol{R}}(i,k) = a_k + \boldsymbol{\theta}_i^\top \boldsymbol{\alpha}_k$ for each unrated item, where a_k is the average rating of item z_k. We call this method `ASVDN`.

6. EXPERIMENTAL RESULTS

In a top-N recommender system, the primary goal is to discover the unrated items that might be liked by a user [3]. We first test the algorithm's ability to discover the unrated items, which is denoted as the scenario `PREDICT-ALL`. The rating matrix is split into three parts as in Eq. (7), and we randomly select $\boldsymbol{R}_{tr}$ to have 5 ratings in each row (for each user). The remaining ratings in the Y_t part are assigned to $\boldsymbol{R}_{te}$. This setting simulates the situation where new users just come into the system with only a few ratings, which is denoted as `PREDICT-ON-5`. Similarly, we create another setting `PREDICT-ON-20` where each row of $\boldsymbol{R}_{tr}$ has 20 ratings to simulate the users with more available ratings. Thus in the scenario `PREDICT-ALL`, there are two settings, namely `PREDICT-ON-5` and `PREDICT-ON-20`. Note that all the items in $\boldsymbol{R}_{te}$ (rated or unrated) could be recommended in the top-N recommendation list. In all the experiments, $\boldsymbol{R}_{tr}$ and $\boldsymbol{R}_{te}$ are randomly selected in 5 different runs.

To test the results, we check if the set of recommended items coincides with the liked items (ratings higher than the median) for each user in the corresponding row of $\boldsymbol{R}_{te}$. Let $\hat{R}_i^N$ and R_i denote these two sets for user y_i. We measure the average precision

$$\text{precision} = (1/|Y_t|) \sum_{y_i \in Y_t} \left(|\hat{R}_i^N \cap R_i| / |\hat{R}_i^N| \right). \qquad (8)$$

	Lastfm		YahooMusic		BookCrossing	
SVDR	0.113	0.083	0.237	0.207	0.078	0.063
ASVDR	0.114	0.087	0.237	0.210	0.078	0.062
NMFR	0.114	0.090	0.218	0.189	0.073	0.054
ANMFR	0.110	0.089	0.211	0.190	0.071	0.056
SVDN	0.002	0.003	0.001	0.001	0.005	0.003
ASVDN	0.003	0.002	0.007	0.005	0.005	0.003
HSVD	**0.180**	**0.158**	**0.258**	**0.227**	0.075	0.065

Table 1: Average precisions in `PREDICT-ALL`. The left column of each dataset: `PREDICT-ON-5`, the right column: `PREDICT-ON-20`. The bold number indicates the method that performs significantly better than the others (p-value < 0.05 in paired t-test).

	Lastfm		YahooMusic		BookCrossing	
SVDR	0.198	0.140	0.490	0.418	0.743	0.744
ASVDR	0.198	0.139	0.496	0.435	0.743	0.745
NMFR	**0.541**	**0.545**	0.763	0.774	0.731	0.740
ANMFR	0.279	0.222	0.579	0.496	0.743	0.745
SVDN	0.177	0.244	0.249	0.224	0.389	0.165
ASVDN	0.537	0.531	0.755	**0.783**	0.696	0.714
HSVD	0.515	0.524	0.765	0.766	**0.745**	**0.747**

Table 2: Average precisions in `RANK-CANDIDATES`. (see the caption of Table 1 for more details)

In the scenario `PREDICT-ALL`, any item which is not in $\boldsymbol{R}_{tr}$ can be considered in the recommendation list. But sometimes we already know a set of candidate items that a user

has suggested implicitly. For example, a user has viewed a list of pages of artists, but did not make any further actions like purchase a CD or download a track. In this case, we can limit our recommendations on the candidate items and only pick up the items that are truly liked by the user, which could be simulated by only recommending the rated items in $\boldsymbol{R}_{te}$. We try to rank these items such that the liked ones are ranked higher. This scenario is denoted as `RANK-CANDIDATES`. The same measure (precision defined in Eq. (8)) is evaluated in `RANK-CANDIDATES`.

We have tested all the methods listed in Section 5 with the above settings on three datasets. `Lastfm` [1] contains the relations between users and artists collected from `last.fm`, and the ratings in `Lastfm` are the actual counts of plays with which a user has listened to an artist. `YahooMusic` [2] is also in the music domain, but includes relations between users and tracks. The ratings in `YahooMusic` are in the range of 1 to 100. The last dataset `BookCrossing` contains ratings from users to books [23]. The range of ratings in `BookCrossing` is from 1 to 10. In each dataset, we take a subset such that the induced graph of the hypergraph representation is connected. When the dataset contains a hypergraph with multiple connected components, we could simply apply the algorithm to each connected component respectively.

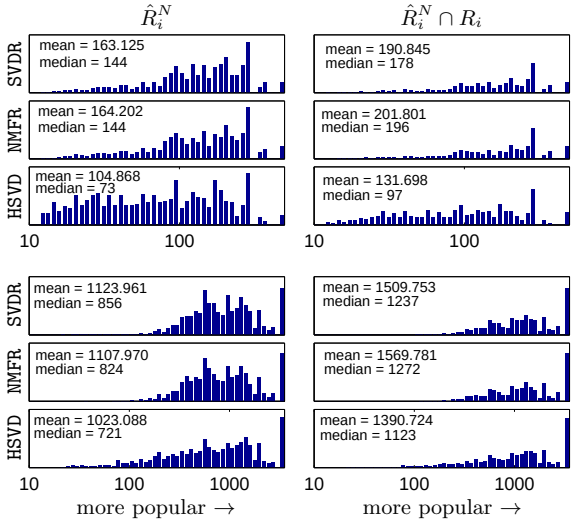

Figure 6: The distribution of popularity of the items (in the `PREDICT-ALL` scenario with $N = 20$). The x-axis (popularity) is in log-scale. The mean and median of the popularity of the items are also shown in each sub-figure. Top: `BookCrossing`. Bottom: `YahooMusic`.

Table 1 and Table 2 show the results of `PREDICT-ALL` and `RANK-CANDIDATES` respectively (with $N = 20$). In the first case, our proposed method performs best within the music domain, and shows results as good as the best methods with `BookCrossing`. The approaches ignoring the empty values (`SVDN` and `ASVDN`) does not perform very well. These methods are originally designed to minimize the root mean squared error on a testing set of known ratings, which is not fully compatible with the scenario `PREDICT-ALL`. In the `RANK-CANDIDATES`, no method performs significantly better than the others over all datasets. But the NMF-based method `NMFR` and two asymmetric methods `ASVDN` and `HSVD` are always as good as the best one.

In recommender systems, we are mainly interested in recommending non-popular (long-tail) items, because the users are usually already aware of the popular items from other sources [19]. For the purpose of diversifying recommendations, we measure the popularity distribution of the items in $\hat{R}_i^N$ and $\hat{R}_i^N \cap R_i$, i.e. the set of all recommended items and the set of successfully recommended items (Figure 6). The popularity of an item z_k is the number of ratings in the k-th column of the rating matrix, i.e. $\deg(z_k)$. We observe that `HSVD` produces more diverse recommendations with the same (or better) level of precision. Especially with `BookCrossing`, there are much more less-popular recommendations from `HSVD` compared to `SVDR` and `NMFR`.

Figure 7 explains why our approach produces more diverse recommendations. By using SVD without normalization, the lengths of the anchor items are not necessarily the longest ones. In fact, we can observe that there is a strong correlation between the popularity and the length of the latent vector with `ASVDR`, which would promote the popular items in the recommendations. In our approach `HSVD`, the longest latent vectors correspond to the anchor items of smaller popularity.

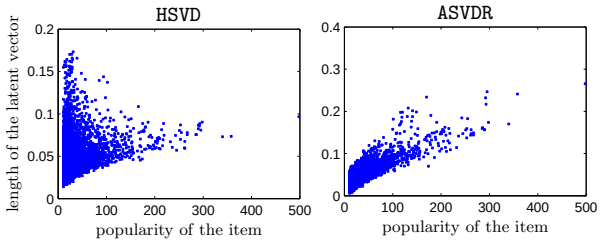

Figure 7: The length of the latent vectors and the popularity of all the items in `BookCrossing`.

7. CONCLUSION

In this work, we present a new perspective of SVD for constructing a latent space from the training data, which is justified by the transformation of normalized hypergraph cut. We show that the latent vectors representing the items in the latent space can be grouped into (approximately) orthogonal clusters, and the lengths of the vectors can be used to distinguish the anchor items and the non-anchor items. These properties are then utilized for making top-N recommendations in the proposed algorithm `HSVD`. Instead of explicitly constructing the clusters, the recommendations are generated by solving a linear system. We provide new explanations for the significantly better performance of the asymmetric SVD approaches and better diversity in top-N recommendations. Experiments conducted with three dataset on two domains also confirm our analysis.

Future works could include further studies of the latent space structures, and engineering methods that exploit such structures. For example, we can explicitly extract the anchor items to construct some clusters, and analyze the features of a non-anchor item by studying the underlying anchor clusters. It is also possible to include additional information,

[1] Available at `http://mtg.upf.edu/node/1671`
[2] Yahoo! Webscope dataset ydata-ymusic-kddcup-2011-track2 `http://labs.yahoo.com/Academic_Relations`

e.g. contextual information [2] and social information [21], or rating bias corrections [10] into our scheme for further improvements.

8. REFERENCES

[1] S. Agarwal, K. Branson, and S. Belongie. Higher order learning with graphs. In *ICML*, 2006.

[2] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *Recsys*, 2011.

[3] P. G. Campos, F. Díez, and M. Sánchez-Montañés. Towards a more realistic evaluation: testing the ability to predict future tastes of matrix factorization-based recommenders. In *Recsys*, 2011.

[4] F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[5] S. Funk. Netflix update: Try this at home, `http://sifter.org/~simon/journal/20061211.html`. 2006.

[6] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.

[7] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[8] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. In *Proc. of the 34th annual Design Automation Conference*, 1997.

[9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, 2008.

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

[11] D. Mavroeidis. Mind the eigen-gap, or how to accelerate semi-supervised spectral learning algorithms. In *IJCAI*, 2011.

[12] B. K. Mohan, B. J. Keller, and N. Ramakrishnan. Scouts, promoters, and connectors: The roles of ratings in nearest-neighbor collaborative filtering. *ACM Transactions on the Web*, 1(2):8, 2007.

[13] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.

[14] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. of KDD Cup and Workshop*, 2007.

[15] L. Pu and B. Faltings. Hypergraph learning with hyperedge expansion. In *ECML-PKDD*, 2012.

[16] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Recsys*, 2008.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. In *WebKDD*, 2000.

[18] TimelyDevelopment. `http://www.timelydevelopment.com/demos/NetflixPrize.aspx`. 2006.

[19] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Recsys*, 2011.

[20] L. Wu, X. Ying, X. Wu, and Z. Zhou. Line orthogonality in adjacency eigenspace with application to community partition. In *IJCAI*, 2011.

[21] Q. Yuan, L. Chen, and S. Zhao. Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation. In *Recsys*, 2011.

[22] D. Zhou, J. Huang, and B. Scholkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, 2007.

[23] C. Ziegler, S. McNee, J. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW*, 2005.

APPENDIX

Proof for Proposition 1:

Firstly, it can be shown that $\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_{k'}$ if $z_k, z_{k'} \in C_j$, because the k-th column and the k'-th column of $\boldsymbol{S}_s\boldsymbol{F}^\top$ must be the same to obtain the same $\bar{\boldsymbol{x}}_k$ and $\bar{\boldsymbol{x}}_{k'}$, which implies that $\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_{k'}$. Secondly, we consider the full SVD of $\bar{\boldsymbol{X}}$ and list the rows of $\boldsymbol{V}$ corresponding to items in C_j:

$$\boldsymbol{V}_{(j)}^\top = \begin{bmatrix} \boldsymbol{\beta}_j & \boldsymbol{\beta}_j & \cdots & \boldsymbol{\beta}_j \\ \boldsymbol{\gamma}_1 & \boldsymbol{\gamma}_2 & \cdots & \boldsymbol{\gamma}_{|C_j|} \end{bmatrix}. \tag{9}$$

It is always possible to find a linear combination of the first s columns of $\boldsymbol{V}$ (denoted as $[\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_s] = \boldsymbol{F}$) such that $\boldsymbol{F}\boldsymbol{t}_j = \left[\boldsymbol{0}^\top \cdots \boldsymbol{1}^\top \cdots \boldsymbol{0}^\top\right]^\top$, where $\boldsymbol{t}_j$ are the coefficients, and the 1's on the right hand side correspond to the items in C_j. Because all the columns in $\boldsymbol{V}$ are orthogonal to each other, a column is also orthogonal to a linear combination of some other columns (e.g. $\boldsymbol{F}\boldsymbol{t}_j$). Thus the entries in each dimension of the vectors $\{\gamma_1, ..., \gamma_{|C_j|}\}$ sum up to 0. In other words, $\sum_{k=\{1,2,...,|C_j|\}} \gamma_k = 0$, and $\boldsymbol{V}_{(j)}^\top \boldsymbol{1} = |C_j| \left[\boldsymbol{\beta}_j^\top \boldsymbol{0}^\top\right]^\top$. Since all the rows in $\boldsymbol{V}$ are also orthogonal to each other, we have $\left(\boldsymbol{V}_{(j)}^\top \boldsymbol{1}\right)^\top \left(\boldsymbol{V}_{(j')}^\top \boldsymbol{1}'\right) = 0$, which implies that $\boldsymbol{\beta}_j^\top \boldsymbol{\beta}_{j'} = 0$.

Proof for Proposition 3:

By Eq. (6) we know that $\boldsymbol{\beta}'_k = \sum_{j \in I(z'_k)} \sqrt{\frac{\deg(z_j)}{\deg(z'_k)}} \boldsymbol{\beta}_j$ (the residual $\boldsymbol{r}$ is zero because the involved anchor items are non-overlapping). Thus $\|\boldsymbol{\beta}'_k\|_2^2 = \sum_{j \in I(z'_k)} \frac{\deg(z_j)}{\deg(z'_k)} \|\boldsymbol{\beta}_j\|_2^2 + \sum_{j,j' \in I(z'_k), j \neq j'} 2\frac{\sqrt{\deg(z_j)\deg(z_{j'})}}{\deg(z'_k)} \boldsymbol{\beta}_j^\top \boldsymbol{\beta}_{j'}$. Since the anchor items are non-overlapping, we have $\sum_{j \in I(z'_k)} \deg(z_j) = \deg(z'_k)$. The above equation can be written as

$$\|\boldsymbol{\beta}'_k\|_2^2 = b_0^2 + \sum_{j,j' \in I(z'_k), j \neq j'} 2\frac{\sqrt{\deg(z_j)\deg(z_{j'})}}{\deg(z'_k)} \boldsymbol{\beta}_j^\top \boldsymbol{\beta}_{j'}. \tag{10}$$

Following the proof for Proposition 1, it can be shown that

$$\boldsymbol{\beta}_j^\top \boldsymbol{\beta}_{j'} = \frac{-1}{|C_j||C_{j'}|} \left(\sum_{p \in I^{-1}(j)} \sqrt{\frac{\deg(z_j)}{\deg(z'_p)}} \boldsymbol{\beta}'_p\right)^\top \left(\sum_{q \in I^{-1}(j')} \sqrt{\frac{\deg(z_{j'})}{\deg(z'_q)}} \boldsymbol{\beta}'_q\right), \tag{11}$$

where $I^{-1}(j)$ is the set of non-anchor items that contain a fraction of an anchor item in cluster C_j. Since each $\boldsymbol{\beta}'_p$ is a linear combination of the $\boldsymbol{\beta}_j$'s of the anchor items, and we know from Proposition 2 that $\boldsymbol{\beta}_j^\top \boldsymbol{\beta}_{j'} \approx 0$ for $\forall j \neq j'$, it is easy to show that $\boldsymbol{\beta}_j^\top \boldsymbol{\beta}_{j'} < 0$. Applying this result to Eq. (10), it concludes to $\|\boldsymbol{\beta}'_k\|_2 < b_0$.