A Hybrid Matrix Factorization via Graph Signal Processing

Ljupche Milosheski **Alen Carin** **Blagoj Mitrevski** **Mladen Korunoski**
École Polytechnique Fédérale de Lausanne
{ljupche.milosheski, alen.carin, blagoj.mitrevski, mladen.korunoski}@epfl.ch

***Abstract*—Predicting whether a user would like a particular item has become significantly important for the success of commercial applications like Netflix, YouTube, Spotify, etc. One of the techniques that have become predominant for this task is formulating the problem as a matrix factorization one and trying to explain each rating as a numerical representation of the corresponding user and item. However, it is a standard problem that could be optimized with stochastic gradient descent (SGD) and there is no novelty in that approach.**

In this project, we try to extend this approach by utilizing well-known techniques from graph signal processing. The extension consists of two parts: 1) utilizing graph Fourier transform to switch from vertex graph domain to spectral graph domain, and 2) applying graph filters to denoise the obtained features. Using these techniques we have achieved RMSE of 0.877, an improvement of 4% over our baseline model – the standard matrix factorization method with SGD.

I. Introduction

With the rise of media service providers such as Netflix, YouTube, and Spotify, recommender systems have become an important aspect of the user experience of these platforms. They are practically unavoidable in the new era of the Internet, from e-commerce – suggesting articles and items that could interest buyers, to the online advertisement – suggesting the right content to users.

Recommender systems are critical for the success of these applications. They let these platforms stand out from their competitors. As a proof of their importance, Netflix has organized a challenge named the 'Netflix prize', where the goal was to model a new recommender system that outperforms their own recommender system by 10%, with a prize of 1 million dollars for the winner. One of the most successful methods developed for the challenge is based on matrix factorization (MF) technique. It is a simple technique without much space for creativity.

In this report, we present the results obtained by using an improved approach of the standard MF method in doing rating prediction by applying novel techniques from the field of graph signal processing. Utilizing the notions of graph frequency and graph filters in order to enhance rating prediction is a relatively new idea in recommender systems. Recent work have also shown that we can employ different deep learning techniques for link prediction on graphs to boost performance [1], [2]. The main difference between the approach of Huang et al. [3] and ours is that we do not exploit the eigenvectors of the correlation matrices, but we use the initial embeddings obtained via MF.

The rest of the report is organized as follows. Section II discusses the structure of the given data. Section III contains an explanation of the construction of graphs and analyses of their properties. Section IV demonstrates the novel approaches in recommendation systems and network science applied. Section V presents, and section VI discusses the results. The report is concluded in Section VII.

II. Data Description

For this project, we used the MovieLens 100k dataset. It contains 100.000 integer-valued ratings in the range from 1 to 5, from 943 users for 1682 movies. Apart from the ratings, it also contains age, gender, occupation, and zip code of each user. It was collected during the seven months period from September 19th, 1997 through April 22nd, 1998 through the MovieLens website. The data was cleaned by the authors in such a way that users that did not have complete demographic information as well as users who had less than 20 ratings in total, were removed [4]. This is a crucial step in the preprocessing phase because it enables us to build a high-quality user graph.

Initially we thought that it would be better if we make use of the additional demographics data or movie genre data. We even crawled the Open Movie Database[1] in order to get additional features for the movies, which we will publish. However, it turned out that it is better if we do not use any additional data, i.e. our method only uses the triplets *(user, movie, rating)*. The reasons are explained in the next section.

III. Graphs analyses

The graph construction is the most crucial factor for the performance of models that involve graph signal processing (GSP) techniques. In this section, we present the construction of the users graphs G_u and movies graph G_m, and we analyze their properties. In addition, we present another possible graph construction technique that may be used for sparse data sets.

The main idea of our method is to use the initial embeddings obtained with MF, feed them into a graph, denoise them with GSP techniques, start the learning algorithm from the new point and keep learning until new convergence. The critical step in this approach is the feature denoising. If our goal is to predict the rating for a new pair of a user and a movie, it would be ideal that both graphs are constructed in such a

[1] http://www.omdbapi.com/

way that the entities, i.e. users or movies, are similar in their ratings. This is an important assumption that we must bear in mind while discussing the graphs constructions.

A. Users graph construction

At first, we thought that creating this graph would be difficult since there was a limited amount of data for the users, and there was more, even scraped data for the movies. However, the MovieLens 100k data set has been preprocessed in such a way that only users who have voted for at least 20 movies are included [4]. That is quite a significant limitation which allows us to confidently do the following. For each user, we subtract the mean of ratings of that user for the movies that he has voted for. Then we get a vector representation of the user with a length equal to the number of movies. A similarity matrix can now be easily constructed between every pair of users. Cosine similarity is a great measure for high dimensional spaces. Then we do ϵ-pruning on the similarities so that the resulting graph is connected, but sparse enough. Intuitively, this is a reliable graph because we can evaluate the similarity between users for a significant number of movies in common. Subtracting the mean lets us get the net rating of the user. The good thing about that is it matches users who have similar tastes in movies, but different strictness of their ratings, i.e., one user may be strict and the highest rating he would give to any movie might be 3. However, it theoretically would not work for users who only rate the best movies because then, the slightly worse movies would turn out to be significantly worse when compare the best. Nonetheless, this graph ended up being very reliable empirically.

It is important to note that we did not need any additional data. The data set had also demographic information. If we connected the users based on that information, let us say based on their profession, then it is not true that two users who have the same profession will have similar ratings. In fact, they might be completely opposite. Thus creating a graph based on that information would be contrary to our assumption that there is an edge between the similar users. The same holds for the age.

B. Movies graph construction

When looked from the perspective of movies, the data set did not have the same property similar to the users in the sense that only movies that have been rated by at least 20 people are present. In fact, there are many movies which have been rated by only a few, or even one user. Let us say we would like to calculate the cosine similarity between movies which have been rated by only one user. If we subtract the movie mean, we get 0 vectors whose cosine similarity is undefined because we have division by 0. If we do not subtract the mean, we either get a cosine similarity of 1 if the users have rated the same movie, or most likely 0 if they have rated different movies. Therefore, it is clear that we cannot use the same approach from the users graph.

Our assumption is that if there is an edge between similar movies, then movies which have similar mean ratings should be connected. Based on that, we connected the two movies if the mean rating between these movies did not differ more than ϵ. The parameter ϵ was 0.11, and was chosen such that the resulting graph is sparse enough. Note that this graph construction results in clique between movies whose pairwise difference between average ratings does not differ by more than ϵ. This results in a disconnected graph for the extreme cases where there is a clique of movies with rating close to 5, and the other most similar movie has rating lower than ϵ. In order to fix this problem, we added one edge between the most similar movies from all consecutive groups, i.e. edge between the movie with the highest rating from the component with lower ratings to the movie with the lowest rating from the component with the higher ratings.

Also this graph construction does not use any additional data, e.g., the movie genre. In fact, using that data would make the graph less reliable because there is no reason to believe that movies sharing a same genre would have similar ratings.

We also experimented with another way to construct the movies graph, which may also be applied to the users graph. It would be very useful for cases when there is no minimum number of ratings for every entity. Since the user representation as described in the G_u construction would not work, we could find the user representations with an initial matrix factorization on the raw data. The resulting matrices give us embeddings of the users in the latent space of the movies and the other way around. These embeddings could be used as representations, and the cosine similarity can be computed based on them. We tried constructing our movies graph with this approach, and it gave us similar results.

C. Graph properties

The user graph G_u ended up being very reliable due to the easy and natural way of construction. The movies graph G_m improved the results, but not by a huge margin. The graphs are not only different in the quality, but also in their properties.

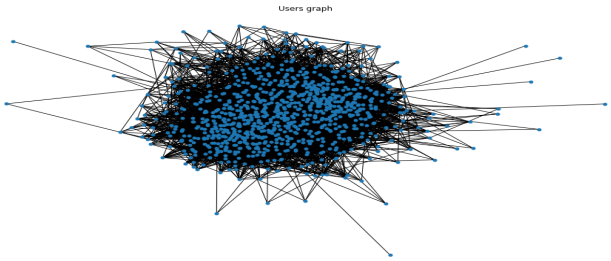

Fig. 1. Structure of the users graph G_u.

First, let's look at the structures of the graphs. Figure 1 shows the structure of the G_u, and Figure 2 shows the structure of G_m. It is expected that the movie graph looks like a line with some dense components – cliques along the path. Its diameter is 39, which is quite high compared to the diameter of G_u which is 5. The results are accordingly to what we would expect given the construction procedure. The degree distribution of G_u looks like a scale-free network, whereas the

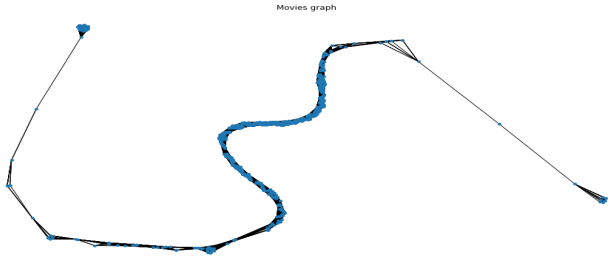

Fig. 2. Structure of the movies graph G_m.

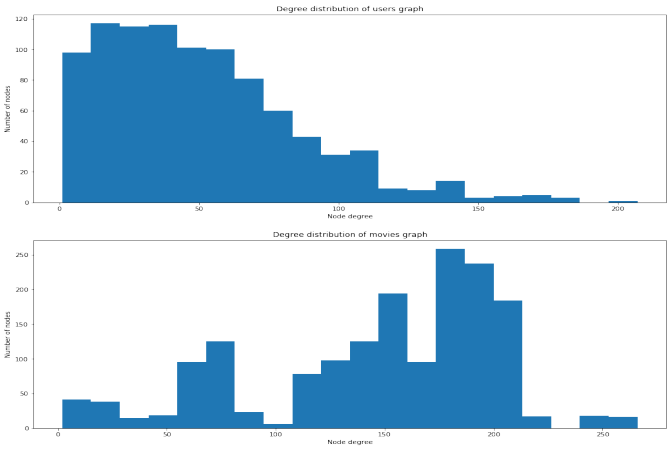

Fig. 3. Distribution of node degrees of G_u and G_m.

one of G_m does not remind us of anything and looks more like an artificially created graph. What is not very obvious from the figure is that it has a much higher clustering coefficient and many more edges than the user graph. The higher number of edges can be identified from the adjacency matrix as it is denser, or the degree distribution in 3 as we can see many points having a high degree. The clustering coefficient is 0.79, whereas the clustering coefficient of G_u is 0.25. The distribution of the clustering coefficient can be seen in Figure 4. This result is also expected because the clustering coefficient is defined as the ratio between the number of edges and the number of possible edges between the neighbours of a node. It is not surprising since G_m has many cliques or almost cliques.

IV. Models and Methods

A. Standard Matrix Factorization (With Bias Trick)

Matrix Factorization was a technique developed for the Netflix challenge [5]. Given a set of triplets *(users, item, rating)*, we need to create a matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ with possibly missing entries, where n is the number of users and m is the number of items. A matrix field $\mathbf{M}_{i,j}$ represents a rating from the user i for the item j. The end result of matrix factorization are two matrices: $\mathbf{P} \in \mathbb{R}^{n \times k}$ is the user embedding matrix, and $\mathbf{Q} \in \mathbb{R}^{m \times k}$ is the item embedding matrix. The i-th row of $\mathbf{P}$ represents the representation – embedding of the i-th user in the latent space of movies. Similarly, the j-th row in $\mathbf{Q}$ represents the representation of the j-th movie in the latent space of users. These embeddings are vectors in $\mathbb{R}^k$.

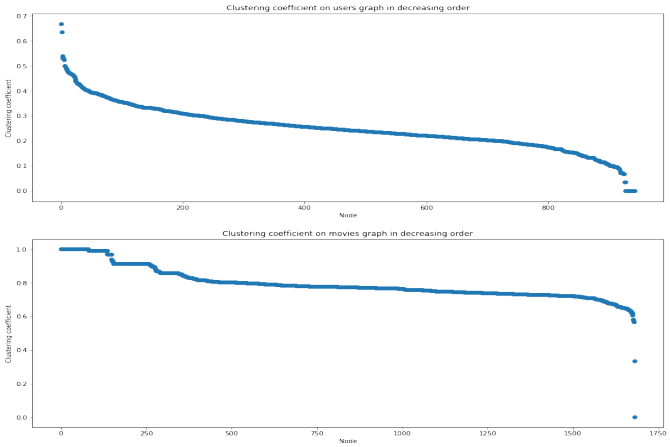

Fig. 4. Distribution of clustering coefficients of G_u and G_m.

The parameter k is called the latent space dimension and is much smaller compared to m and n, thus it allows us to approximate the matrix M with a small number of weights. The loss function of our model is defined as

$$L(P, Q) = \min_{P,Q} \|M - PQ^\intercal\|_F ,$$

where $\|X\|_F$ is the Frobenius norm of X.

This is, unfortunately, a non-convex and non-identifiable problem. However, we can always use stochastic gradient descent (SGD) and converge to a local minimum. We call this the standard MF method. The SGD update rules can be trivially changed so that the model takes into consideration the bias of the users and movies. It is important because some of the users are strict when giving the ratings. Thus we would like to be interested in the net rating of each user, obtained by subtracting the mean rating of each user. We will refer to this model as the MF with the bias trick.

B. Graph Signal Processing

Graph signal processing (GSP), although based on techniques from signal processing, which were developed in the previous century, it slowly started to find its way into data science. The basic element of GSP is the *signal* which is a vector $\mathbf{x} \in \mathbb{R}^N$, where N is the number of nodes in the graph. The i-th value in the signal vector is the value for the i-th node in the graph. The two most important concepts of GSP are graph Fourier transform (GFT) and graph filters.

The GFT of a graph signal $\mathbf{x}$ is defined as $\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$ and $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$, where $\mathbf{U} \in \mathbb{R}^{N \times N}$ is the eigenvector matrix associated with a graph shift matrix $\mathbf{GS} = \mathbf{U}\Lambda\mathbf{U}^\top$. There are two possible options for the $\mathbf{GS}$, one is a weighted adjacency matrix, $\mathbf{W}$, and the other one is Laplacian matrix, $\mathbf{L}$. Using the GSP we can transfer the signal from the vertex domain to the graph spectral (frequency) domain, where it is easier and faster to solve some complex problems and, afterward, convert the modified signal back to the original vertex domain.

Let $h : \mathbb{R} \to \mathbb{R}$ be a scalar function. A Graph Filter, $h(\mathbf{L})$, is an $N \times N$ matrix that takes the form: $h(\mathbf{L}) = \mathbf{U}h(\Lambda)\mathbf{U}^\top$.

Filtering, therefore, amounts to multiplication in the spectral domain. Graph filtering is an important technique for denoising graph signals. Multiplying graph filters with the original signal in the spectral domain, we can get rid of the unwanted frequencies which result in a denoised signal. Because $h(\mathbf{L})x$ requires full eigendecomposition, and in practice, we prefer to work in the vertex domain, we work with polynomial graph filters. A polynomial graph filter of order P requires $P + 1$ sparse matrix-vector multiplications or the complexity of the computation is $O(PM)$ for M edges.

C. Hybrid MF via GSP

Our method is based on both MF and GSP. We start by creating an initial matrix $\mathbf{M}$ with users and movies. The next step is to find embedding matrices for the users and the movies. We call it *hybrid* because we use graph information from the users – collaborative filtering, and graph information from the items – content filtering.

The idea is the following. Obtain initial user and movie embedding matrices P and Q respectively. The rows of P represent the embedding of the users in the latent space of movies. However, we can look at the other way around and say that the users are embedded in each of the latent dimensions of movies. Thus we can look at one of these embeddings as a signal on the graph G_u. We apply a low-pass filter on the signal in order to denoise it. It is in accordance with the meaning of the edges in the graph, i.e., two users are connected if they are likely to give similar ratings. Note that in the worst case we could use a constant filter of 1, and then our worst-case result would be the best MF result without any denoising. However, we hope that using a low-pass filter will move us away from the current local minimum, ideally to a new point in the parameter space, where the performance will be better. Since the new point will likely not be a local minimum, we could continue running SGD until convergence. But it is the same as before, i.e., as if we ran SGD and converged some point. Therefore we repeat the previous step, that is, apply a filter and run SGD until convergence, multiple times.

The only difference in this approach is how we apply the filters. For instance, we could only denoise the users' embeddings, or we could apply two filters at the same time on both, the users and movies embeddings, or perhaps use one filter until convergence, then apply the other to get out of the local minimum, and continue using the first filter. The latter gave us the best results. We call the models where we use either user or movie filters as MF + User filtering and MF + Movie filtering respectively. We name the model where we use both filters simultaneously as MF + User & Movie filtering. The model denoted by (MF + User) Movie filtering denotes the model where we keep using the user filters until convergence. If there is no further improvement when using the user filter, we apply the movie filter once and keep applying the user filter with SGD until the new convergence. The same holds for (MF + Movie) User filtering model. All of the models except the standard MF use the bias trick, which is implemented as adding to columns to P and Q, which we do not denoise. Algorithm 1 describes the (MF + User) Movie filtering model.

Algorithm 1. A pseudocode for the (MF + User) Movie model. $\mathbf{P}$ is the factorized users embedding matrix and $\mathbf{Q}$ is the factorized movies embedding matrix. A low-pass filter is used to denoise the signals. RMSE is used as an error function.

procedure (MF + USER) MOVIE MODEL
  $\mathbf{P}, \mathbf{Q} \leftarrow$ *learn_SGD*$(\mathbf{M})$
  outer_iter $\leftarrow 0$
  while *outer_iter* $\leq$ *filter_max_iter* **do**
    outer_iter $\leftarrow$ *outer_iter* $+ 1$
    $\mathbf{Q}^* \leftarrow$ *denoise*$(\mathbf{Q})$
    inner_iter $\leftarrow 0$
    while *inner_it* $\leq$ *filter_max_it* **do**
      inner_iter $\leftarrow$ *inner_iter* $+ 1$
      $\mathbf{P}^* \leftarrow$ *denoise*$(\mathbf{P})$
      $\hat{\mathbf{P}}, \hat{\mathbf{Q}} \leftarrow$ *learn_SGD*$(\mathbf{M}, \mathbf{P}^*, \mathbf{Q}^*)$
      $\mathbf{P} \leftarrow \hat{\mathbf{P}}$
    $\mathbf{Q} \leftarrow \hat{\mathbf{Q}}$
end procedure

The movie graph is created with the following procedure: for each movie, we calculate its average rating. Then, we create a node for each movie, sort them by ratings and connect the neighbouring nodes so that we have one big component containing all of the movies. The final step is to connect each movie to other movies whose average ratings are within some radius of the one we are observing.

Both of these graphs are used in the learning process to find the embeddings that will best approximate user preferences for movies. An important part of the learning process is denoising for which we use an approximation of ideal low-pass filter with polynomial graph filter.

The pseudocode of the learning process is given in Algorithm 1.

V. RESULTS

Table I summarizes the results for different models. The models were evaluated with 5-fold cross-validation, with the same partitioning. The hyperparameters of the models were tuned for each model separately. We used hand-crafted low-pass filters that were the same across all models, and for all latent dimensions. However, there were different filters for the users and movie embeddings. The low-pass filters were approximated by a polynomial, i.e. polynomial graph filters. We also obtained a much better performance of 0.858 RMSE on a single testing dataset, but then we lost the hyperparameters and also used different partitions in the reported cross-validation results.

VI. DISCUSSION

The standard MF model serves as a solid baseline for the quality of our new methods. With trivial changes, it can be improved so that we then consider the bias of users and movies, which notably improves the results. Any of the methods involving GSP clearly give us better performance than

TABLE I
ROOT MEAN SQUARE ERROR FOR THE DIFFERENT MODELS.

Algorithm	RMSE
Standard MF	0.914
MF with bias trick	0.906
MF + Movie filtering	0.903
MF + User filtering	0.886
MF + User & Movie	0.887
(MF + Movie) User filtering	0.882
(MF + User) Movie filtering	**0.877**

our baselines. Comparing the quality of the models that use only one graph, it is safe to say that the movie graph is not as good as the user graph. As the discussion in Section III, it looks more artificial and does not have the desired properties. However, at least to some degree improves the performance. The reason for the bad quality of this graph is that the data set did not have the desired property of the minimum number of users that have rated each movie. The same conclusion can be made for the methods that involve both graphs. The model where movie filtering is applied more often gives worse results due to the same reasons.

When evaluating our approaches against similar ones, we were surprised by the results reported by Monti et al. [1] and Berg et al. [2]. Monti et al. [1] reported a RMSE of 0.905, while Berg et al. [2] reported a RMSE of 0.929 of their best model on the same MovieLens 100k dataset. We believe the reason for that might be if they rounded the predictions in order to get the final predicted ratings, whereas our predictions are real numbers. Our results are a bit worse than the ones from Huang et al. [3], which report RMSE of 0.8674 on their best model, which is based on the eigenvectors of the covariance matrices to find users and movies representations.

Since MF is a non-convex problem, we have shown that with an exploit of the structure of the graphs, we could jump from one local minimum to usually a better one. Figure 5 compares an initial user embedding obtained with standard MF to an embedding obtained after the convergence of the best model. It is obvious that the coefficients corresponding to high eigenvalues are significantly decreased, i.e. we have decreased the noise. However, there is still some noise present that comes from the SGD steps as we keep running SGD until new convergence after we apply the filters. Also, it can be noticed that the coefficients of more smooth eigenvectors with eigenvalues of around 0.6 are boosted. Similar thing happened to almost all of the user embedding signals. Figure 6 shows what typically happened in the movie features. We can see that smooth eigenvectors are boosted to some degree, and the non-smooth are shrunk. However, there is still significant amount of noise that is learned at about eigenvalues of 1. That suggests that we might have escaped the previous local minimum only by limited degree. The same thing happens for many of the movie embedding signals, which is in accordance with our assumption that the movie graph is not very reliable.

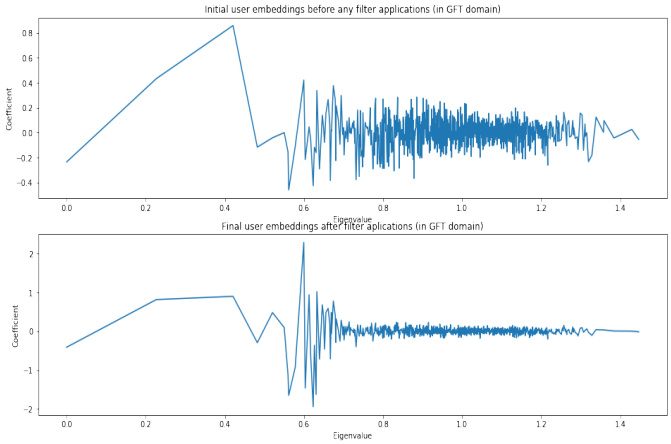

Fig. 5. A user embedding signal without any filter applications in comparison to the same signal after convergence of the algorithm.

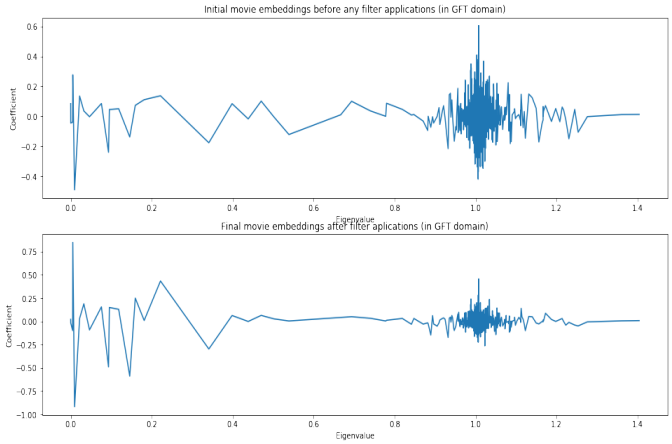

Fig. 6. A movie embedding signal without any filter applications in comparison to the same signal after convergence of the algorithm.

VII. CONCLUSION

In this paper, we presented a new hybrid method which is based on the standard MF technique to further improve the performance via GSP techniques. Our results are almost on par with the recent approaches by Huang et al. [3].

An important property of the method is that it works without any additional data and can be applied to any recommender system problem. It is dependent on the data set in the sense that one of our graph construction techniques requires all entities to have been rated at least some number of times. We also discussed an idea that mitigates this problem to some extent.

Convolutional neural networks (CNN) have been shown to work well on recommender systems [2]. We used hand-crafted filters which are difficult to tune. As future work, we believe that using CNN to find the best filters will further improve our results. In addition, learning one filter for each latent dimension should give the best results because each dimension is likely to have different best filters. We do not know what the latent dimension represents, so it would be safer to learn different filters.

REFERENCES

[1] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 3697–3707.

[2] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[3] W. Huang, A. G. Marques, and A. Ribeiro, "Matrix completion via graph signal processing," in *43rd IEEE Int. Conf. Acoust., Speech and Signal Process., Calgary, AB*, 2018, pp. 15–20.

[4] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, p. 19, 2016.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.