Movie recommender system using signal diffusion

Deniz Ira, Jonathan Labhard, Daniil Dmitriev, Paul Griesser
Department of Data Science, EPFL, Switzerland

January 2020

Introduction

Recommendation systems are of a great popularity nowadays with the spread of online shops, advertising and streaming platforms. Classical methods of movie recommendation rely on matrix completion methods, such as matrix factorization. While they exhibit a strong performance, usually there is no simple way to incorporate explicitly the relations between the users or the movies. The amount of such structured data is enormous nowadays and it can be naturally described using graph representations. The goal of this project is to explore such graph based recommendation system.
Our recommendation system will be built using the Movielens 100k dataset which contains 100,000 ratings from 982 users on 1682 movies. Along these ratings, the data also contains some information about the movies and the users such as the genre and release date for the movies and the gender, age, and occupation for the users. This data matches perfectly with our task as it allows to easily construct different graphs using either movies or users as the nodes, and edges between the nodes can be constructed based on their features.
For the purpose of our project we will create a graph connecting similar movies together. The choice of similarity is crucial and is discussed in details further below. Then, for each user we create a signal on the graph where its values are the ratings of that user for all movies. The problem is that the signal are available only for a small subset of the nodes. Recommending movies to a user then reduces to recover the full signal over the graph.
In order to do so we will use graph signal processing techniques such as Graph Fourier Transform and Chebyshev graph filtering. The details of the methods are given in the "graph exploitation" section.

Data acquisition

The ratings data has been collected directly from the MovieLens web site, it can be downloaded by following this link. The data comes in the form of three tables: the first one contains the user information, the second one contains the features of the movies and the third table provides the ratings which links the users with movies. From these tables, we have multiple ways to create a graph structure. As said in the first section we decided to focus on a graph where the nodes represent the movies and the edges represent the similarity between them.

Building the graph

To build the adjacency matrix representing our graph, we need a good similarity metric between movies. Instead of looking at the features of the movies to describe their similarity, we decided to look at their ratings. So we first constructed the ratings matrix $\mathbf{R} \in \mathbb{S}^{U \times M}$ ($\mathbb{S} = \{0, 1, 2, 3, 4, 5\}, U = \#users, M = \#movies$) such that the entry R_{um} corresponds to the grade given by the user u for the movie m. In the case where the user didn't rate that movie the entry is set to 0. We plotted the distribution of the number of ratings per movies and of the number of ratings per user.

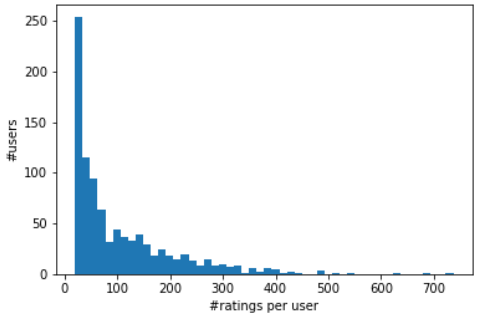

Figure 1: Distribution of number of ratings per user.

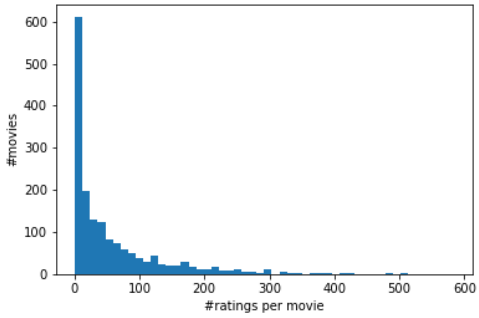

Figure 2: Distribution of number of ratings per movie.

The distributions are power laws and show that most users and movies have a low number of ratings meaning a sparse matrix $\mathbf{R}$. But we also see that the number of ratings for certain movies is near to zero. In fact there is 141 movies that have only 1 rating and 530 that have less than 10 ratings. This is a problem as we decided to use ratings to link movies together. Thus, we removed movies for which the number of ratings is less than 5. Then we weighted the edges between movies according to three metrics.

1. Cosine similarity $C(m_i, m_j)$: We can view the movies as a vector of ratings, such vector are sparse and are in a high dimension so the cosine similarity metric makes more sense than a simple euclidean distance.

2. Common seen movies $N(m_i, m_j)$: This is just the number of users that watched both movies m_i and m_j. As this metric is not between 0 and 1 we normalized it using min-max scaling.

3. Common ratings similarity $E(m_i, m_j)$: Here, for a pair of movies we only look at the users that rated both movies m_i and m_j, we then take the normalized euclidean distance between those ratings. The euclidean distance is taken in a lower dimension space, this dimension being the number user that rated both movies. Doing this puts a bigger weight on movies that received similar ratings.

Then we combine these metrics to build the adjacency matrix. So our final adjacency matrix $\mathbf{W}$ can be described by the following formula:

$$w_{i,j} = \begin{cases} 0, & \text{if } i = j \quad (1) \\ 1 - (\alpha C(m_i, m_j) * \beta N(m_i, m_j) * \gamma E(m_i, m_j)) * \mathbb{I}(w_{i,j} > \epsilon) & \text{otherwise} \quad (2) \end{cases}$$

Where $\mathbb{I}$ is the indicator function. The "1−" comes from the fact that in the adjacency matrix we consider the weights rather than the distances. So we take an average of our three metrics and then set certain weights to zero if they are under a certain threshold. We then tuned the parameters $(\alpha, \beta, \gamma, \epsilon)$ in order to have a single connected component and to have a relatively low average degree by node. We used grid search to find the best parameters. We kept the 10 adjacency matrix with the lowest average degree in the end we will compare the recommendation given using the different graph and keep the one giving the best results.
We represented the resulting graph on gephi, we also removed some of the edges for visualization purpose. We can see some different clusters appearing in the graph, we can interpret them as representing movies rated by common users.

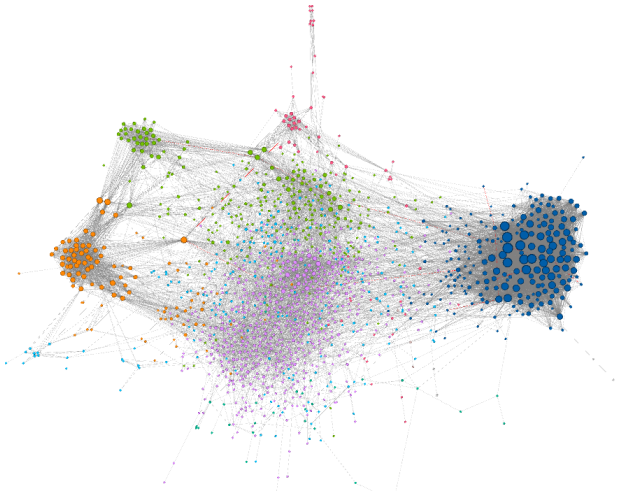

Figure 3: Graph representation using gephi

Graph exploration

The data from the recommendation systems is quite irregular, as we do not have control over certain factors such as which movies were popular and received a lot of ratings and which users have just joined the platform and are not active. Thus, we should be very careful when choosing a way to create the graph structure, since on one hand, we may end up with multiple disconnected components which makes many methods not applicable, or with a very dense graph, where most of the movies are connected to each other, complicating further analysis.

As discussed before, we associate with each edge a weighted average of three distances and we threshold our graphs weights in order to have a single connected component. The graph is quite dense having 1349 nodes and 84841 edges. This can be explained by the fact that a lot of people have watched and rated similar popular movies. When we compare our graph to random graph model, we see the most similarity with scale free networks, which explains that we also have very popular films that serve as hubs. We see that distributions match quite well. All 3 graphs have the same diameter 3. Clustering coefficients are equal to 0.3 in Movies graph, 0.15 in the Barabási-Albert model and 0.19 in the Power-law model. The density of our graph can be seen either by the clustering coefficient, which is twice as big as the one in the Barabási-Albert model, or from the diameter of the graph, which is equal to 3. Both these facts show that films can be divided into very strongly tied clusters and even these clusters are close to each other.

These characteristics support the idea of using signals as rates on the graph. Since the unrated movies will have many rated movies in close neighbourhood, the propagation of the signal for non rated movies will make the estimated prediction accurate.

Graph exploitation and method

In this section we explain in more detail how we use our graph and the known ratings of a user to recommend him movies. As stated in the first section, known ratings for a user are modeled as a signal on the movie graph. For each movie node, the signal value will be the rating of that user for that movie, and zero if the movie hasn't been rated.
To learn the zero values of the signal, we use graph filters with Chebyshev Polynomials. We use a filter bank of heat kernels, which simulates heat diffusion when applied to a given signal. Diffusing

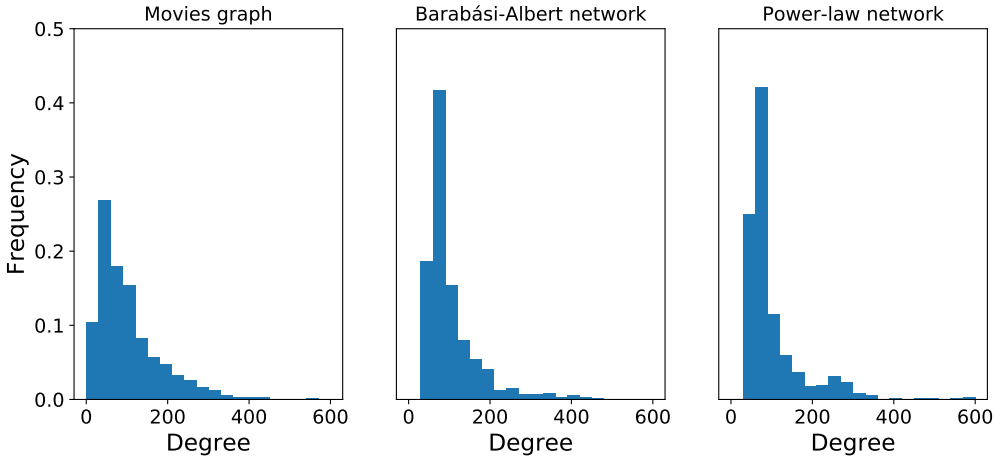

Figure 4: Comparison of degree distribution between movies graph and random graph models.

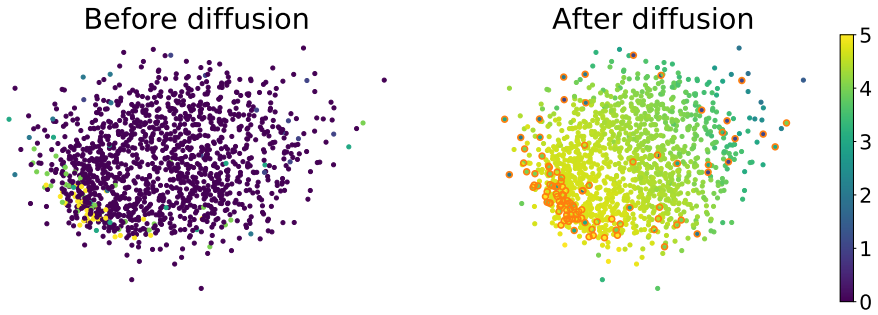

Figure 5: Example of graph signal before and after heat diffusion for a fixed user. We see that the closer the node is to the region with many rated movies, the more diverse are the estimated marks. On the second figure, nodes that are given in the train set are highlighted.

the signal to the nodes that have zero values in a smooth manner corresponds to infering the ratings of all non-rated movies for a user. Given the signal of a user — aka the vector corresponding to his movie ratings — we first constuct orthogonal Chebyshev polynomials. Then we build corresponding heat graph filters. Then the values for unknown nodes are computed recursively with the Chebyshev graph filtering method. We then scale the newly obtained values to the range of ratings used in our dataset, i.e. between 1 and 5. At this step we also replace the values of the signal corresponding to the known ratings with the original ratings, we do this because those ratings are changed by the heat diffusion kernel. We iterate this process multiple times, each time diffusing the signal locally.

The underlying assumption for this method to work is that the final signal should be globally smooth on the graph, that is that movies connected by strong edge weights will tend to have similar ratings for a given user.

To recommend movies to a user, we will learn how this user would rate all movies using the method described above and then output the movies that he didn't see that have highest learnt rating.

```
[5.         5.         4.98165316 4.97122001 4.95865196]
1188                        Prefontaine (1997)
1557                          Aparajito (1956)
1511    World of Apu, The (Apur Sansar) (1959)
923                         White Squall (1996)
1125          Old Man and the Sea, The (1958)
```

Figure 6: Recommended movie using signal diffusion

```
[5.         4.90247276 4.83384131 4.78890741 4.75378828]
1188                Prefontaine (1997)
174                     Brazil (1985)
318     Everyone Says I Love You (1996)
169             Cinema Paradiso (1988)
172          Princess Bride, The (1987)
```

Figure 7: Recommended movie using matrix-factorization

Evaluation

In order to evaluate our recommender system, we compare it to a matrix-factorization based recommendation system as well as a biclustering based recommender. Both methods are known to perform well on this kind of problem and it make sense to compare our model with it. We use the root mean-square error (RMSE) to measure the performance of the models. Given true test ratings x_{um} for the u'th user and m'th movie, and our prediction $\mathbf{P} \in \mathbb{R}^{U \times M}$ either filled using our model or by the matrix-factorization algorithm ($\mathbf{P} = \mathbf{W}\mathbf{Z}^T, \mathbf{W} \in \mathbb{R}^{M \times K}, \mathbf{Z} \in \mathbb{R}^{U \times K}$), we compute the RMSE as follows:

$$RMSE(\mathbf{P}) = \sqrt{\frac{1}{|S|} \sum_{(u,m) \in S} \frac{1}{2}[x_{um} - P_{um}]^2}$$

We compute the RMSE for known ratings, so before applying our model we hide a part of the signal for each user, by simply setting it to zero. This part will be used as a test set and is denoted by S.

Results and conclusion

Model	RMSE
Matrix-Factorization	0.756_{437}
Signal Diffusion	0.992_{612}

We see that both method gives good results but the matrix-factorization based recommender system is still better than ours. But it should be noted that the RMSE compute the differences between the true ratings and the predicted ones, but for our recommender system we care less about the predicted ratings, indeed we are more interested in the relative order of the prediction, so that we can give a user some recommendation. We can see this in the figures 6 and 7.

One way to improve our recommender system would be to incorporate more information about the movies when computing their similarity. For example one could explore the genre of the movies, and add a term in (2) that takes into account genre-similarity of movies. One also could explore other graph signal processing techniques for signal retrieval on graph, here are some related papers: [1] [2] [3].

References

[1] Eduard Sanou Antonio Ortega Sunil K. Narang, Akshay Gadde. Localized iterative methods for interpolation in graph structured data. 2013. https://arxiv.org/abs/1310.2646.

[2] José M. F. Moura Jelena Kovacevic Siheng Chen, Aliaksei Sandryhaila. Signal recovery on graphs. 2014. https://www.researchgate.net/publication/268988465_Signal_Recovery_on_Graphs.

[3] Farokh Marvasti Mahdi Boloursaz Mashhadi, Maryam Fallah. Interpolation of sparse graph signals by sequential adaptive thresholds. 2017. http://ee.sharif.edu/~imat/papers/9.pdf.