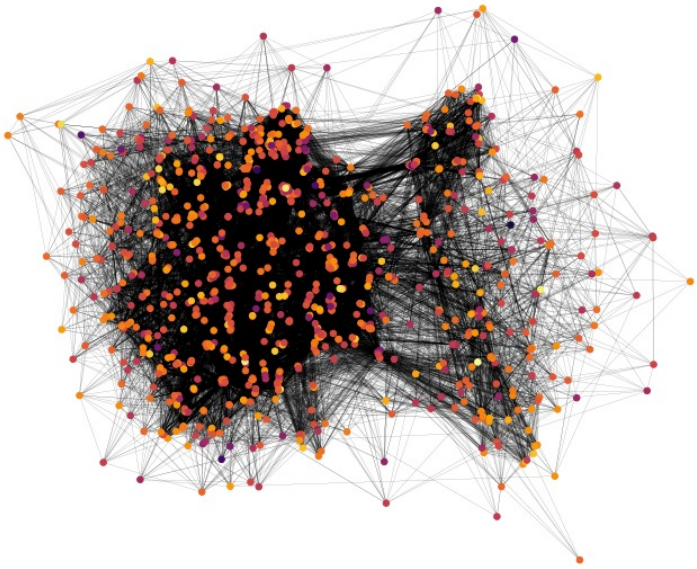
École Polytechnique Fédérale de Lausanne

A Network Tour of Data Science

Movie Recommender Exploration



Team 15
Arthur BABEY
Stanislas DUCOTTERD
Nessreddine LOUDIY
Lamyae OMARI ALAOUI

Lausanne - Switzerland

Fall semester- 2019

***Abstract*—The main goal of this project is to build a film recommender and explore it. We used the MovieLens database, which contains ratings given to movies by users as well as information about those users and movies. The goal is to fill in the missing ratings. To do this we used the implementation of *Berg et al.* [1] with additional information taken from the IMDb website and the zip code of the users to have more features. Using extended movies and users features we were able to get a root mean squared error of 0.900.**

I. Introduction

With the increasing amount of users consuming streaming services such as Netflix, companies are striving to provide the best recommender system for their clients. Our aim in this project is to predict a user rating to a given movie. This problem can be considered as a matrix completion problem. In the context of graphs, matrix completion can be seen as a link prediction problem. We will be using graph convolutional matrix completion in order to reach our goal. More specifically, we are going to use the implementation of *Berg et al.* [1] and their code taken from [2].

II. Data set

A. Movielens

This project is based on data obtained from MovieLens and more specifically the MovieLens 100k data set. MovieLens is a web-based personalized movie recommendation system. Several data sets have been built using their database, the smallest being MovieLens 100k. It contains 100,000 ratings from 943 users on 1682 movies which means that almost 95% of the values are missing in the matrix. Various types of information are available about the users (age, gender, occupation, zip code) and the movies (release date, genre). The ratings range from 1 to 5. MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota. The Movielens 100k data set was collected through the MovieLens website (movielens.umn.edu). The data has already been cleaned up to keep only the users who gave more than 20 ratings and shared their demographic information.

B. Extending features

In order to improve our data and be able to make potentially better predictions, we added additional information about the users that we gathered using their zip codes and additional information about the movies by scraping the IMDb website.

We first used the zip code information in the Movielens data set to find in which US state the users are living. This will allow us to group users by their state and to see if there are any geographical patterns. This information allowed us to use a data set from the American Community Survey containing information about the mean and median household income at the zip code level. This allowed us to use the median income information as a feature. A small minority of the users did not have an American zip code, we set their median income to be equal to the median income of all the users.

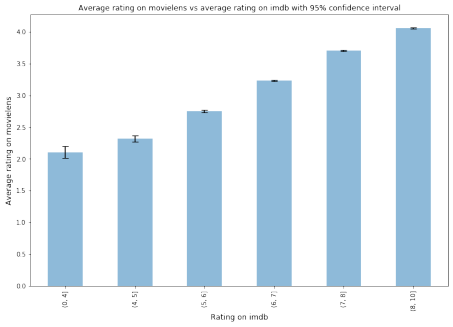


Figure 1. Ratings comparison between IMDb and MovieLens

On the other hand we also expanded the movie features using the IMDb website which is a famous online database of information related to movies and television shows. We did not have the IMDb movie id in our original data set and the IMDb URL that we had was outdated. We managed to find these information by using the movie titles found in the MovieLens data set, we set an automatic procedure to look for each movie title in the IMDb search engine. In order to avoid outliers we implemented a robust procedure; first we used the raw movie title and then we used the movie title without the parentheses and the information between them, this gave us two ids for each movie. Those two ids allowed us to merge the MovieLens movie data set with the IMDb one and to compare their respective information. When the two ids were different we choose the one that was the closest to the MovieLens with respect to the release date. After that procedure, around 10% of the retrieved movies were judged to be too different from the MovieLens one and were checked manually and corrected if needed. Thanks to the new IMDb URLs we were able to obtain additional movie information such as the directors, the writers, the length in minutes, the average ratings on IMDb and in which countries the movie has been published. We did not take the directors and writers into account as there was respectively 1204 and 4708 of them which would have led to almost 6000 dimensions features if we used one-hot encoding. The countries were not used as the data set was too messy and a big part of the MovieLens movies were missing them.

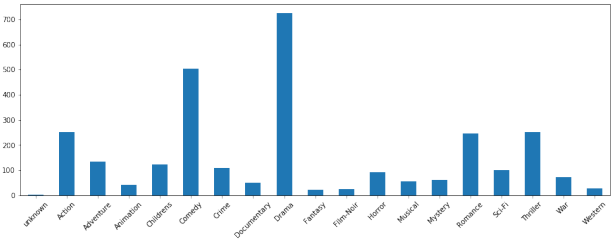


Figure 2. Number of films by genre

III. Exploratory Data Analysis

In order to have a better understanding of our data we looked in details some of our features, how they are

distributed and how they correlate with the ratings :

Gender	Count	Mean	95% CI
Female	273	3.532	[3.52, 3.543]
Male	670	3.530	[3.523, 3.536]

Table I
AVERAGE RATINGS BY GENDER

We computed basic statistical information for our different features. For instance, we can see in figure 2 the number of movies by genre. We explored the relationship between the user features and their ratings. We are interested in understanding how ratings are distributed between users and movies thus we computed the average rating for each feature. We can see in table I that even though the data sets contain almost three times more men than women the average ratings were quite similar. We did not see any patterns on how average ratings were distributed according to age nor household income. By looking at figure 4 we can see that there are some US states where users seem to give better ratings. Highest rates come from West Virginia where the lowest are from North Dakota. All the confidence intervals were computed using the bootstrap technique with 1000 samples.

We did the same with movies and their ratings. We can see in figure 3 that the average rating does not depend so much on the occupation of the user since almost all of the confidence intervals overlap, with the exception of the people working in the healthcare industry. We can see in figure 1 without any surprise that the average rating on IMDb correlates quite well with the average rating on MovieLens.

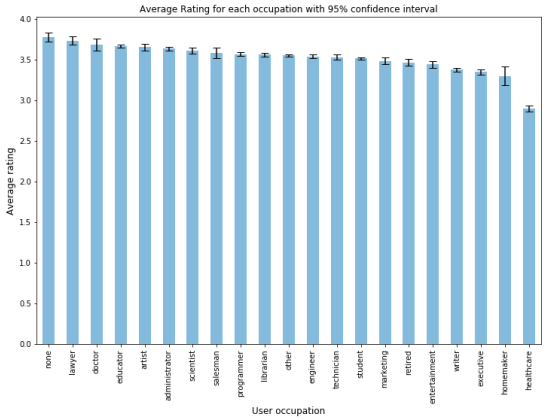


Figure 3. Average ratings according to the user occupation

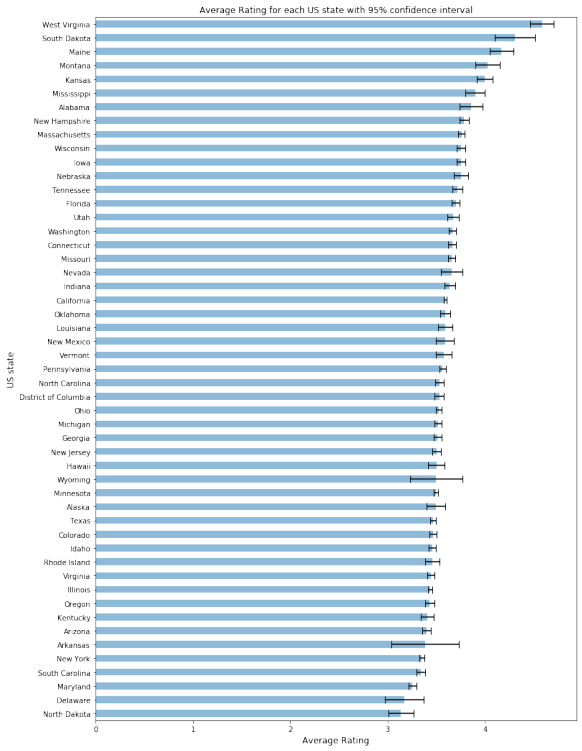


Figure 4. Average ratings according to their state

IV. FEATURE MATRICES

Based on our data we built two feature matrices. These matrices will be used as side information for our model. The first matrix contains 74 dimensional user features; the age, the gender, a one-hot encoding of the 50 US states and Washington, the median income per year of its location given by the zip code and a one-hot encoding of 21 possible occupations. We used a one hot encoding to represent the US state information by doing this we had to add 51 features. We removed one of the 21 occupations from the one-hot encoding because it does not bring any information. We did not do the same for the US state as having zero everywhere tells us that the user does not live in the US. The second matrix is about the movies and contains 22 dimensional features; the age of the movie in 1998, the year from which the data set is taken, the genre which is also a one-hot encoding (19 different genre including unknown which was removed), the average ratings from IMDb website, the length of the movie in minutes and finally the number of vote on IMDb.

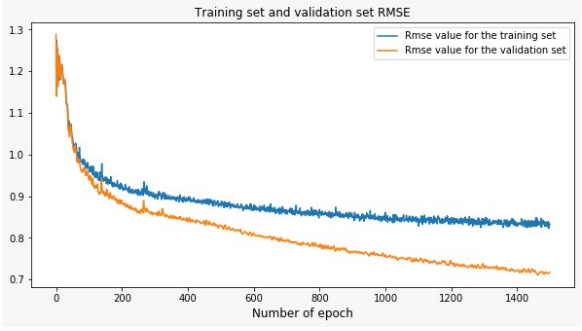


Figure 5. Loss

V. Model description

The model uses a variational graph autoencoder; the encoder maps every user and movie to an embedding space of dimension E, the decoder learns five different matrices each corresponding to a rating in order to predict the probability of each rating using the softmax function :

$$p\left(\check{M}_{ij} = r\right) = \frac{e^{u_i^T Q_r v_j}}{\sum_{s \in R} e^{u_i^T Q_s v_j}}$$

Where $\check{M}_{ij}$ is the predicted rating of user i for the item j and u_i, v_j are vectors corresponding to the user i and item j in the embedding space.

The model uses a two layer neural network. The first one is the convolutional encoder while the second one is the dense decoder.

VI. Training and results

We used the authors code that we modified in order to be compatible with python 3 and to output the embeddings for visualization purposes.

We used the same hyperparameters as the authors except for the epochs that we set to 1500 and the 35 hidden units for the dense side information. We split our data set folowing the canonical u1.base/u1.test MovieLens. To train our model we used a learning rate of 0.01, the Adam optimizer and a dropout of 0.7. We evaluated our model on the test set using an exponential moving average with a decay factor of 0.995. We can see in figure 12 that both training and validation RMSE are decreasing. Surprisingly the validation loss is always smaller than the training one. This is because the training is submitted to dropout which is not the case for the validation. We were able to obtain a 0.900 RMSE average over 5 runs on the test set with standard deviation of 0.0015 . Thanks to our extended feature we were able to obtain a slightly better result than the original paper [1].

Model	RMSE
GC-MC without feature	0.910
GC-MC+Feature	0.905
GC-MC+ Extended Feature	0.900

Table II
Results

VII. Visualization

A. Users feature graph

We built a epsilon-similarity graph, where the nodes represent the 943 different users. We used the euclidean metric to compute the distance between the 74 dimensional features of each user and create the adjacency matrix. The graph weights were computed using a Gaussian kernel whose expression is as follows:

$$w(i,j) = \exp(-\frac{|| x_i - x_j ||_2^2}{\sigma^2}) \quad (1)$$

Here σ denotes the kernel width, it was set to 0.4 times the mean over all distances. In order to control the level of sparsity of the graph, we used a threshold of 0.15.

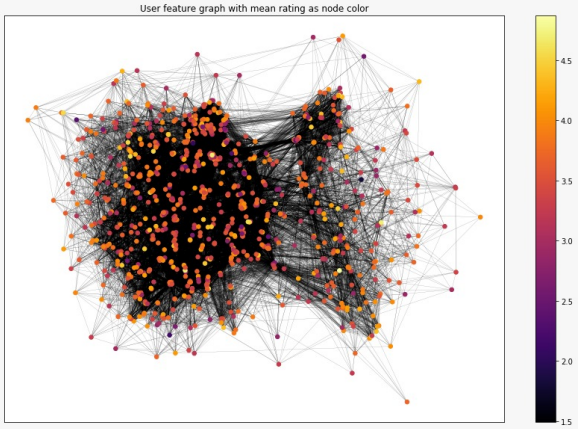


Figure 6. User feature graph with mean ratings as node color

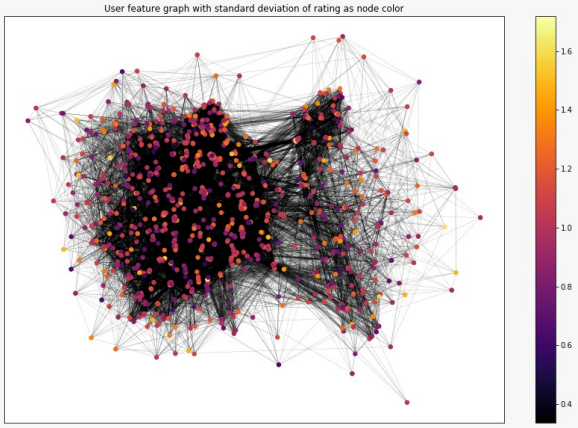


Figure 7. User feature graph with standard deviation of ratings as node color

B. Movies feature graph

We used The same method as the users to create the movies graph. σ and the threshold were chosen the same way.

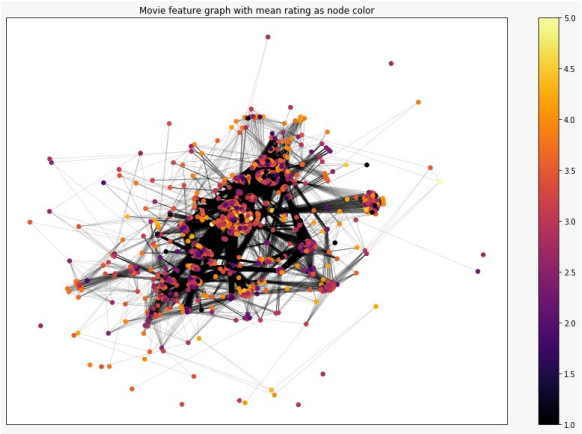


Figure 8. Movie feature graph with standard deviation of ratings as node color

C. Users embedding without features graph

We used to the embedding representation that we obtained by training the model without adding side information of each user to build an other epsilon-similarity graph using euclidean distance on the 75 dimensional latent representation. σ and the threshold were chosen the same way as before.

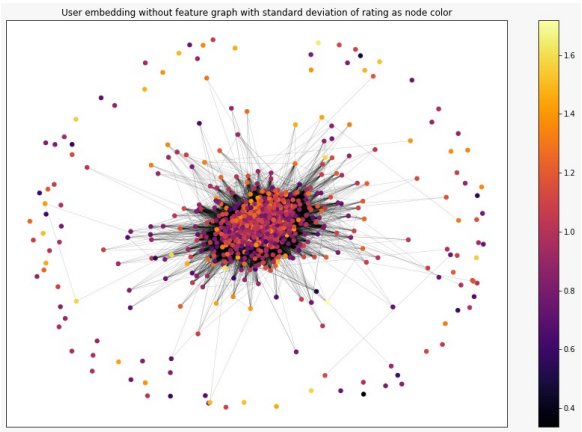


Figure 9. User embeddings without features using standard deviation of ratings as node color

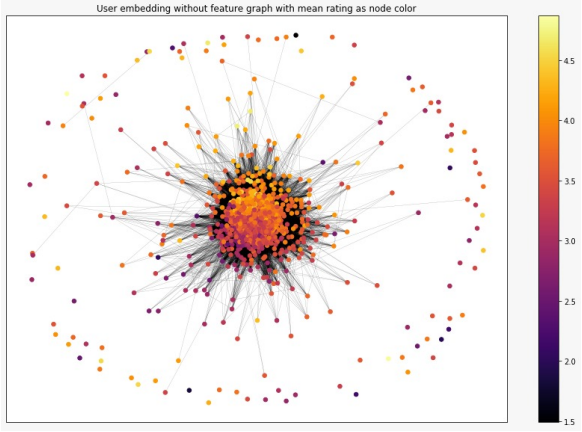


Figure 10. User embeddings without features using the mean of ratings as node color

D. Users embedding with features graph

Same as before except this time we train the model with our extended features. σ and the threshold were chosen the same way as before.

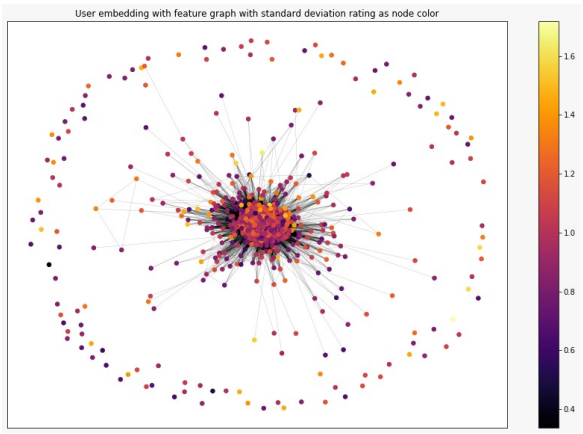


Figure 11. User embeddings with features using standard deviation of ratings as node color

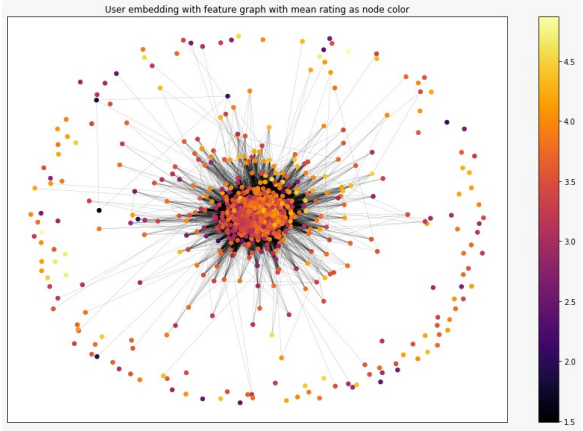


Figure 12. User embeddings with features using the mean of ratings as node color

Graph	#Nodes	#Edges	Average degree	Degree standard deviation	#Connected components	Clustering coefficient	Diameter
Features	943	42400	89.93	62.38	1	0.66	3
Embedding without features	943	136929	290.41	214.20	93	0.71	infinite
Embedding with features	943	125491	266.15	217.88	120	0.70	infinite

Table III
NETWORK

VIII. Discussion and conclusion

We were able to reproduce the results from *Berg et al.* and her team since we also obtained a root mean squared error of 0.905 with the MovieLens-100k data set. We slightly improved this result by extending features as described in section II this increase was the same as the increase the authors got when they added features to the purely collaborative model they made. Table II resumes the performance of both models and we can see that our

final result has a 0.900 RMSE which outperforms Berg's RMSE.

During this project the exploration of our data allowed us to understand them better and it gave us the idea of extending the features. We were able to do so by website scrapping and using the different public APIs available. This led us to a more complete data set. During the graph visualization we extracted some graph information such as the number of edges and the clustering coefficient as you can see in II. .

References

[1] R. van den Berg, T.N. Kipf, and M. Welling. Graph Convolutional Matrix Completion *arXiv preprint arXiv:1706.02263*, 2017.

[2] https://github.com/riannevdberg/gc-mc