École Polytechnique Fédérale de Lausanne

EE558 - Network Tour of Data Science

Project : Build a network based recommendation system

Authors:
Lukas De Loose
Niklas Glaser
Maxime Lamborelle
Nils Ter-Borch

Professors:
Frossard Pascal
Vandergheynst Pierre

Assistants:
Michaël Defferrard

January, 2020

1 Introduction

What movie should I watch tonight? A very common, yet personal, question that movie recommendation systems try to answer. In our project, we want to get insights on the way users rate movies, predict ratings for a movie and identify what influences that rating.

Most recommendation algorithms are based on Singular Value Decomposition (SVD). We chose a network based approach. Two movie similarity graphs (nodes are movies) and a user similarity graph (nodes are users) are created. We focus on applying the user ratings on a movie graph, and attempt to recover the user ratings of unrated movies with Total Value Regularization (TVReg). In another approach, we try to predict the movie ratings using a user's neighborhood on the user graph.

2 Dataset

We use three datasets, the first is `Movielens 100k`[1], which provides 100'000 user ratings by 943 users on 1'682 movies, on a 1 to 5 scale. In addition this dataset contains a few movie features (genre) and user features (age, occupation, zip code). These where too general and didn't vary enough to be usable. To get better movie features, we use the `TMDb`[2] dataset, containing 5'000 movies and many features. We join the `TMDb` and `Movielens 100k` datasets, using the `Movielens-latest`[3] dataset as a bridge. The resulting dataset contains 483 movies, with the following features: budget, genres, keywords, original language, popularity, production companies, production countries, release date, revenue, runtime, spoken languages, title, vote average and vote count.

3 Creating networks

3.1 Movie rating graph

The *movie rating graph* connect movies (nodes) based on user ratings (edges). To compute the distance between two movies, the group of users who rated both movies is selected. For every user, the the ratings difference is computed and the L2 norm evaluated. The distance is normalized by the square root of users in common:

$$d_{m_{ij}} = \frac{\left\| \left[F_{m_i} - F_{m_j} \right]_{\Omega_{m_{ij}}} \right\|_{\ell_2}}{\sqrt{|\Omega_{m_{ij}}|}}, \tag{1}$$

where $\Omega_{m_{ij}} = \Omega_{m_i} \cap \Omega_{m_j}$ is the set of users who rated both movies. As the movie difference in the rating is always calculated for the same user, we do not have to account for a bias, i.e. a user rating always below average. To get a sparse adjacency matrix from the distance matrix a Gaussian kernel is used:

$$w_{ij} = \exp\left[-\frac{d_{ij}^2}{\sigma^2} \right]. \tag{2}$$

The edges with distance $d_{ij} > \epsilon$ are removed. For this graph $\epsilon = 1.1$ is chosen and $\sigma = 0.5$ is set to include weights up to a small value of 0.01. The *movie rating graph* is visible in Fig. 2(a).

3.2 Movie feature graphs

The *movie feature graphs* connect movies (nodes) based on movie features (edges). Certain features are removed. For example, only six movies are not in English, the economic data is only available for 25% of movies and the production country is strongly correlated to the production company. The remaining selected features are **Genre**, **Keywords** and **Production companies**, they are used to create three interactive graphs (to see them, click on any of the bold paragraph names below), and are eventually combined in one feature graph.

Genre The dataset contains 19 different genres, the Hamming distance is used as a measure of distance, i.e. the number of non-matching genre categories.

Keywords With 2'400 distinct keywords, many are very specific and often similar, e.g. 'alcoholic' and 'alcoholism'. Natural Language Processing (NLP) could match synonyms, but is outside of the scope of this study. Instead, only the first word is read such that 'alien contact' and 'alien life' become 'alien', reducing the the number of keywords

[1] https://grouplens.org/datasets/movielens/100k/
[2] https://www.kaggle.com/tmdb/tmdb-movie-metadata
[3] https://grouplens.org/datasets/movielens/latest/ dataset

	Movie rating graph	Movie feature graph	User rating graph
Number of nodes	480	480	943
Number of edges	14452	7880	106640
Mean degree	60.2	32.9	226
2nd moment of degrees	4716	1451	73040
Number of connected components	6	3	1
Clustering coefficient	0.2837	0.5915	0.5763
Mean degree centrality	0.1257	0.0686	0.2400

Table 1: Properties of the graphs

to 2'100. Each movie has 9.8 keywords on average. The Jaccard distance is used to calculate the distance between movies:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{3}$$

Production companies Movies are associated with 447 distinct production companies, e.g. 'Columbia Pictures'. Each movie is produced by 2.26 companies on average. Again, the Jaccard distance is used.

Combining the features We create the adjacency matrix by first summing the three distances and then subjecting them to the Gaussian kernel, equation (2), with $\sigma = 1$. To get a sparse graph, distances above $\epsilon = 2$ are removed. The result is shown in Fig. 2(b).

3.3 User rating graph

For the user neighborhood prediction, the *user rating graph* connects users (nodes) with similar ratings (egdes). The distance of user a to user b is defined by:

$$d(a, b) = 4 - \sum_{i \in R_a \cap R_b} \frac{4 - |R_{a,i} - R_{b,i}|}{N_a}, \tag{4}$$

where $R_{a,i}$ is the rating of user a for the i-th movie and N_a is the number of rated movies by user a. Meaning, that over the movies watched in common, the absolute value of the difference in the ratings is summed up. This is normalized by the number of movies user a to avoid that a few users, who have rated many movies, dominate the distances. Furthermore, the distance is conceptualized to be $d = 0$ if user b has exactly the same ratings as user a. The maximum distance is $d = 4$. To create a sparse adjacency matrix, a Gaussian kernel (2) is used. In this graph, all edges with weights below a value of 0.3 are dropped. Tests on the dataset lead to equation (4) and $\sigma = 1.92$.

4 Exploring networks

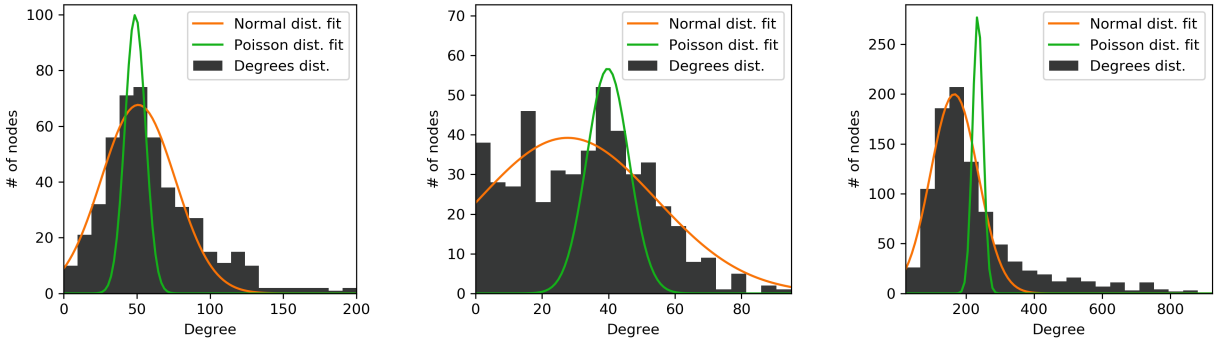
4.1 Properties of the graphs and nodes

The main graphs features are summarized in Table 1. In combination with the graphical representation of the degree distribution, see Fig. 1, we see that the graphs follow more or less a Gaussian distribution. Except for the *movie feature graph*, we have only a few nodes with a very low degree, a centralized peak degree, with a wider distribution than the random network following a Poisson distribution. As we have a high 2nd moment of the degrees, in comparison to the mean, we conclude, that our graph can be described as a mixture of scale-free with preferential attachment (big 2nd moment) and random, equally distributed weights (localized distribution). For this task, it helps that only the user network provides hubs (users that rated many movies), so that all provide good interconnectivity between the nodes.

4.2 Insights from the visualisation

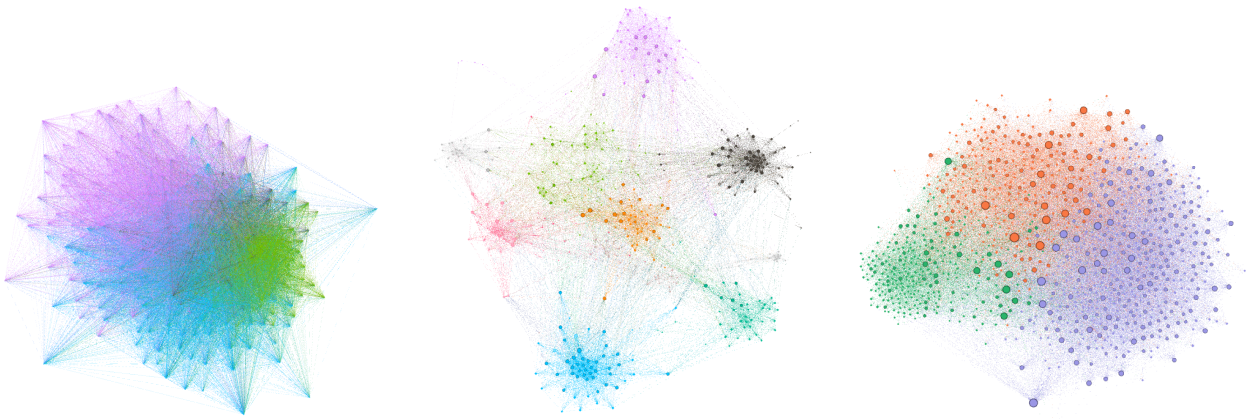
Fig. 2 show the three graphs (see also the interactive version[4]). The arrangement of the nodes is determined by the ForceAtlas 2 algorithm in the Gephi[5] software and the colors represent different modularity classes. Judging from the visual representation we have a high interconnectivity between the modularity classes, showing no significant clustering for the rating based graphs, while the movie feature graph shows 8 distinct clusters of movies. The ratings seem to reflect more subtle criterias than the available set of movie features, hence movie genres are not separated based on ratings.

[4] `https://lambo97.github.io/MovieRecommendation/visualization/`
[5] `https://gephi.org/`



(a) Movie rating graph (b) Movie feature graph (c) User rating graph

Figure 1: Histograms of the graph node degrees. A Poisson (green) and normal (orange) distribution was fitted to the degree distribution for the three individual graphs.



(a) Movie Rating Graph (b) Movie Feature Graph (c) User Rating Graph

Figure 2: Visualisation of the networks by Gephi using the ForceAtlas 2 algorithm to compute the node arangement.

5 Exploitation

5.1 Different approaches

5.1.1 TV Regularization

To solve the movie recommendation system problem, each set of ratings coming from a user, is considered as a signal lying on the *movie rating graph*. The problem is that a lot of ratings are missing. The known ratings are between 1 and 5 and the unknown ratings are equal to 0. We try to reconstruct the signal of the ratings for all movies using a TVReg method. Because the signal of ratings is a piecewise constant function, it is the most appropriate approach to recover this signal. The TVReg method tries to recover the signal x by solving the optimization problem :

$$\tilde{x} = \underset{x \in \mathbb{R}^N}{\arg\min} \left\{ \|Ax - y\|_2^2 + R_{\text{tv}}(x; G) \right\}, \tag{5}$$

where $R_{\text{tv}}(x; G) = \alpha \|Sx\|_1 = \alpha \sum_{w_{ij} \in E} \sqrt{w_{ij}} |x(i) - x(j)|$.

5.1.2 User Neighborhood

Another approach is to use the neighborhood of a user to create a weighted average over the ratings that were given by a users "soul mates". We can compare this model to Amazon's, 'other users who watched, also liked this...' . The key argument of this algorithm is to say, that in the neighborhood of a user the rating behavior will barely change and be strongly correlated with nearby users. For every user in the *user rating graph* the ratings of connected users are merged to a weighted average:

$$R_{a,i} = \frac{\sum_{b,E_{ab}\neq 0 \wedge R_{b,i}\neq 0} R_{b,i}}{\sum_{b,E_{ab}\neq 0 \wedge R_{b,i}\neq 0} 1}, \tag{6}$$

with $R_{a,i}$ the rating of user a on movie i and E_{ab} the edge from user a to b. As this model has no tuning parameters we have designing the graph very carefully, using basic assumptions and tests with the training subset.

5.1.3 Matrix Factorization

When talking about recommendation systems, the matrix factorization approach which has been popularized by Simon Funk during the Netflix Prize in 2006, cannot be avoided[6] The idea is very simple, factorize the matrix $R \in \mathbb{R}^{(n\times m)}$ of ratings by n users who rated m movies into two matrices $q \in \mathbb{R}^{(m\times f)}$ and $p \in \mathbb{R}^{(n\times f)}$ where the dimension f such that $f \ll m$ and $f \ll n$ is a parameter of the model. So now each user is represented as a set of f feature $p_u \in \mathbb{R}^f$ and each movie as a set of f feature $q_i \in \mathbb{R}^f$. Then the predicted ratings are simply :

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u,$$

where μ is the mean of the ratings, b_i and b_u represent the bias of the users and the movies and have to be learned. q and p are the two matrix factors which also have to be learned. These parameters are simply learned by minimizing the regularized error :

$$\min_{b_*,q_*,p_*(u,i)\in\mathcal{K}} \left(r_{ui} - \mu - b_i - b_u - q_i^T p_u\right)^2 + \lambda_4 \left(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2\right).$$

The parameter λ_4 is the regularization parameter and is determined using cross validation. The minimization is performed using a stochastic gradient descent algorithm as in many other machine learning models.

5.2 Results

5.2.1 TV Regularization on a Single User Network

First we test the TVReg approach on a subset of each network, using only one signal (one user's rating). We choose the user which has rated the highest number of movies and keep only the nodes corresponding to these movies. User 13, rated 304 movies out of 480. The number of missing ratings is controlled by applying a mask on this signal to hide some correct ratings. We can then use the TVReg to reconstruct the ratings that have been hidden. Finally we can compute the Root Mean Squared Error (RMSE) between the correct hidden ratings and the predicted ratings.

To improve the stability of our model, we compute the mean ratings that each movie obtained, without considering user 13. Then we apply this mean signal on the noisy signal (signal where some ratings have been hidden) before the regularization.

The regularization parameter γ of the model has to be tuned. We tested several possible values, from 1 to 20, and get the best results for a value of $\gamma = 13$, irrespective of the percentage of missing values or the signal.

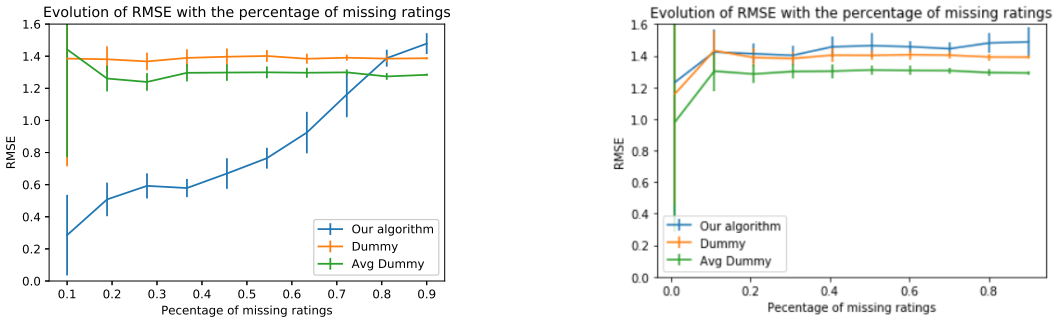
In Fig. 3, we vary the percentage of missing ratings, for each value we created 10 different masks and apply the TVReg model. Then we evaluate each prediction using the RMSE. We also make these tests for two dummy models, the first one (`Dummy`) just returns the mean of the signal for all missing ratings. The other one (`Avg Dummy`) returns the signal with the mean ratings of each movie present (just before passing the regularization model). We can see that for a low percentage of missing values, the model perform very well and is able to reconstruct the signal with very low error. However when the percentage of missing values goes higher than 70%, the quality of the results becomes equal or even less good than the dummy algorithms we tested.

The results of the *movie feature graph* are much worse than the *movie rating graph*. This can be partly explained by the quality of the feature graph. Genre, keywords and production company are not very representative of a movie's success. There are multiple ways to improve this, using more advanced NLP techniques on the keywords or using more features (e.g. popularity, budget and actors). From here on, we leave the *movie feature graph* out of the TVReg analysis.

5.2.2 TV Regularization on the Entire Network

Now that we assessed the performance of the TVReg method on a subset of the network, we will test it on the entire network. Here the algorithm is much more unstable and the best value of γ highly depends on the signal, so we decided to take a constant $\gamma = 8$. To test our method we decided to take for each signal a test set of 30% of the ratings we have for the network and use the other 70% to predict the ratings for the whole network. We then average the RMSE obtained for each signal to get the final RMSE. However the results we got are not satisfying, the model

[6] Koren et al. "Matrix Factorization Techniques for Recommender Systems". Computer 42 (2009). https://doi.org/10.1109/MC.2009.263



(a) Movie rating graph

(b) Movie feature graph

Figure 3: Evolution of the RMSE of the predicted ratings given the percentage of missing ratings

predicts almost always a value near to the mean rating of 3 and the final RMSE is **1.13** which is just below the one we obtained using the Dummy model. One reason is that for some signals (users) we have very little data (ratings) and the algorithm has trouble reconstructing the signal based on them. Moreover, this method does not use collaborative filtering (except for the construction of the *movie rating graph*). The information that some models could extract with collaborative filtering is much higher than the one we use here.

5.2.3 Neighborhood Prediction

Using a training-validation split of 70%:30%, this method results in an RMSE of **0.976** which performs better than TVReg. Despite fairly simple assumptions, this method could actually be used to predict ratings, although it does not perform quite as well as the Matrix Factorization. Many attempts to reduce the RMSE where made, yet they did not improve the results. An iterative approach was tried by recalculating the weights on the predicted users, which led to a very high similarity between users and high RMSE. Using the *movie feature graph* as basis to filter the predicted signal did not lead to an improvement, as the optimization problem of the filter using the training data led to a flat filter. Concluding on this method it performs better than the TVReg, but there is still potential to improve. The comparatively fast calculation is a plus and makes it possible to optimize the network.

5.2.4 Matrix Factorization

We also test the matrix factorization algorithm SVD to compare the achievable RMSE with a method which does not use the graph structure developed before. We use the `surprise`[7] library which implements the SVD algorithm.

We use a training set with 20% of the ratings. The other 80% are used to first search by cross validation the best value for the number of latent feature and then to train the algorithm using the best value. After the cross validation, we got an optimal number of latent factors of 13. With that number, we compute the RMSE on the ratings predicted for the test set and we obtained **0.93**. We can see that using the quite simple algorithm which does not use any graph structure nor information about movies can achieve better results than the previous two approaches.

6 Conclusion

From our analysis we observed that the method with best results do not directly correlate well rated and poorly rated movies. No gender based difference was observed. Relating movies based on their features gave the best graph clustering but did not allow for performance TV regularization. Better results might be achieved with more information on movies. Movie prediction, especially for users who rated few movies, remains a hard topic. It highlights the importance of a large community and accurate data collection on both users and movies to arrive at relevant recommendations. Other approaches with NLP, including keywords, movie description and even user based reviews to construct user and movie graphs may give better results.

[7] https://surprise.readthedocs.io/en/stable/