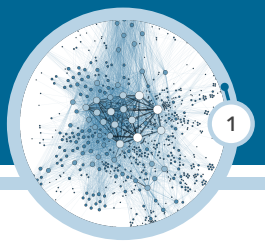
EE-558 : A Network Tour of Data Science
Project Presentation

Movie Recommendation System

Lukas De Loose
Niklas Glaser
Maxime Lamborelle
Nils Ter-Borch

January, 2020

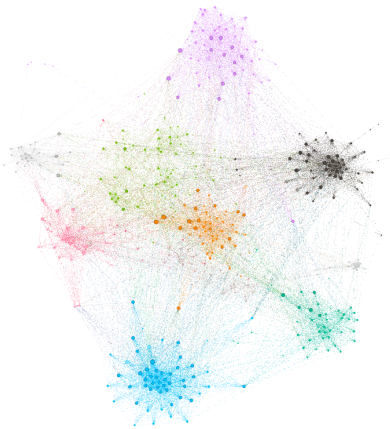
Outline



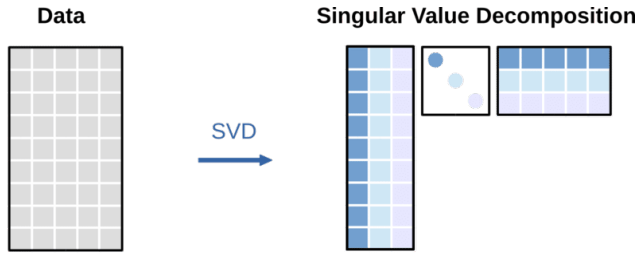
Introduction
On recommendation systems

What movie should I watch? Different approaches are possible.

Our choice: Network based approach



Most popular for recommendation systems: Singular Value Decomposition[1]



- Using Total Value Regularization
- Using user neighborhood ratings

[1] Randomized Matrix Decompositions using R - Scientific Figure on ResearchGate. (accessed 18 Jan, 2020) Available from: https://www.researchgate.net/figure/Conceptual-architecture-of-the-randomized-singular-value-decomposition-The-data-are_fig3_305995021

Part I : From Data Sets to Networks

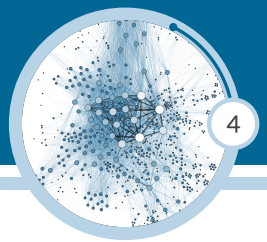
Data Collection

Combining 3 datasets

- **`Movielens 100k`** : movie ratings by 940 users on 1'600 movies
- **`TMDb`** : movie features of 5'000 movies
- **`Movielens-latest`** : provides links between datasets

Final dataset has 480 movies and 940 users

Network Creation

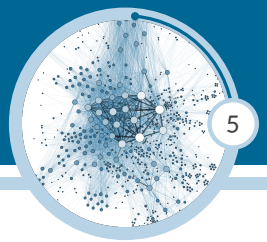


Building three Networks

- **`Movie rating graph`**: Connects movies via user ratings
- **`Movie feature graph`**: Connects movies via movie features
- **`User rating graph`**: Connects users via user ratings

Network Creation

Movie ratings graph



Distance between two movies based on ratings of users who rated both:

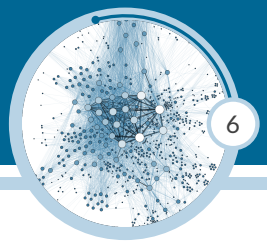
$$d_{m_{ij}} = \frac{\left\| \left[F_{m_i} - F_{m_j} \right]_{\Omega_{m_{ij}}} \right\|_{\ell_2}}{\sqrt{\left| \Omega_{m_{ij}} \right|}}. \tag{1}$$

A RBF kernel transforms distances into weights:

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

with $\sigma = 0.5$ and a cutoff distance $\epsilon = 1.1$.

Network Creation
Movie features graph

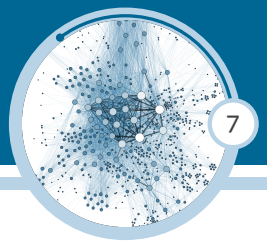


Distance between each pair of movies computed by summing the distances from each feature:

- Genre : hamming distance between genre of both movies
- Keywords : Jaccard distance
- Production company : Jaccard distance

$$J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

RBF kernel ($\epsilon = 2$, $\sigma = 1$) used to compute weights.



Distance between two user a and b:

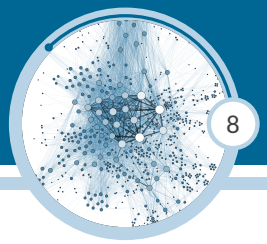
$$d(a,b) = 4 - \sum_{i \in R_a \cap R_b} \frac{4 - |R_{a,i} - R_{b,i}|}{N_a}. \tag{3}$$

A RBF kernel is applied with $\epsilon = 2.1$ and $\sigma = 1.92$ to compute weights

Part II : Network Exploration

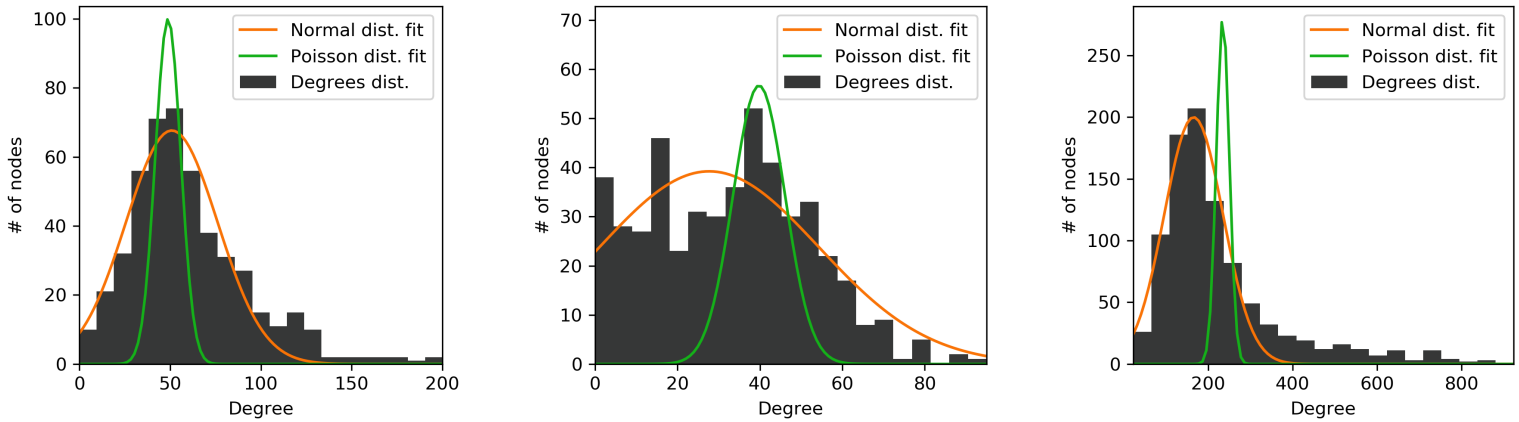
Network Properties

General Properties



	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

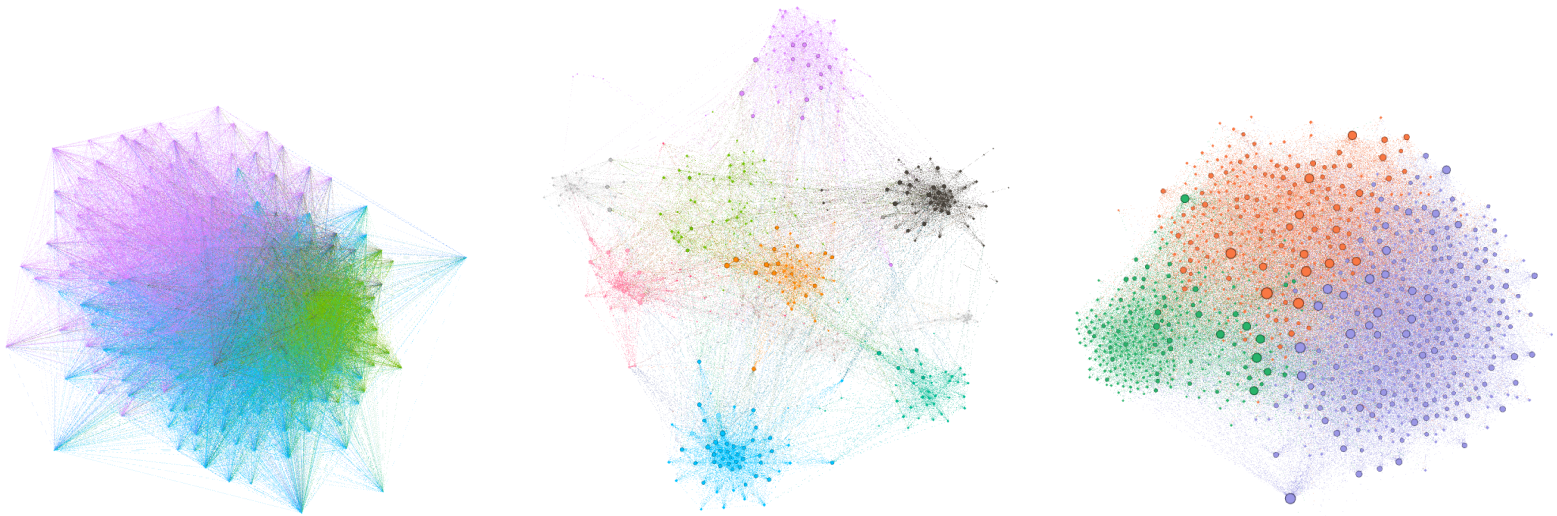
Network Properties
Degree Distribution



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

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

Visualization



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

Figure: Visualisation of the networks by Gephi using the ForceAtlas 2 algorithm to compute the node arangement and the modularity class for colors.

Part III : Exploitation Methods and Results

TV Regularization
Method

Idea: user ratings are signals on movie graphs

- One signal per user
- Try to reconstruct signal (predict ratings)
- Piecewise constant signal
- Initialization with the mean ratings of other users

Method: Total Variation (TV) regularization:

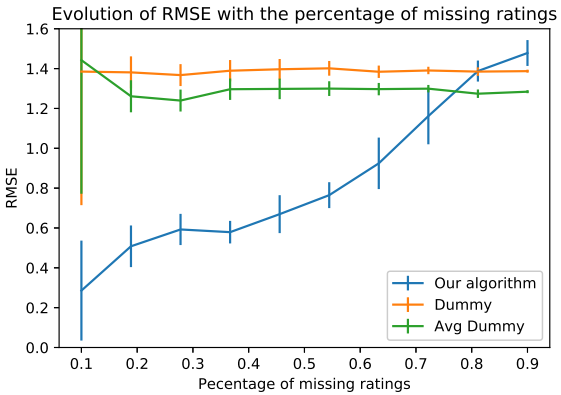
$$\tilde{x} = \underset{x \in \mathbb{R}^N}{\arg\min} \left\{ \|Ax - y\|_2^2 + R_{\text{tv}}(x; G) \right\}, \tag{4}$$

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

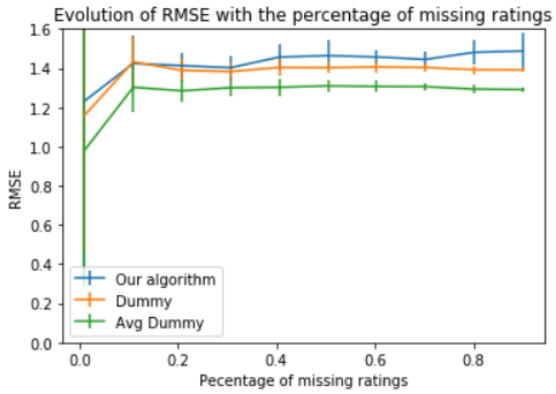
TV Regularization

Results on a single user network

Testing the algorithm on sub-networks for a single user, against two dummy algorithms. User 13 rated most movies. The sub-network contains only movies that user 13 has rated.



(a) Movie rating graph

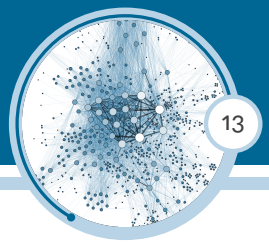


(b) Movie feature graph

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

TV Regularization

Results on the entire network



- results are unstable
- Value of γ depends highly on signal
- We use a fixed value of $\gamma = 8$
- Train / test set split 70% / 30%
- High computation load
- RMSE : **1.13**

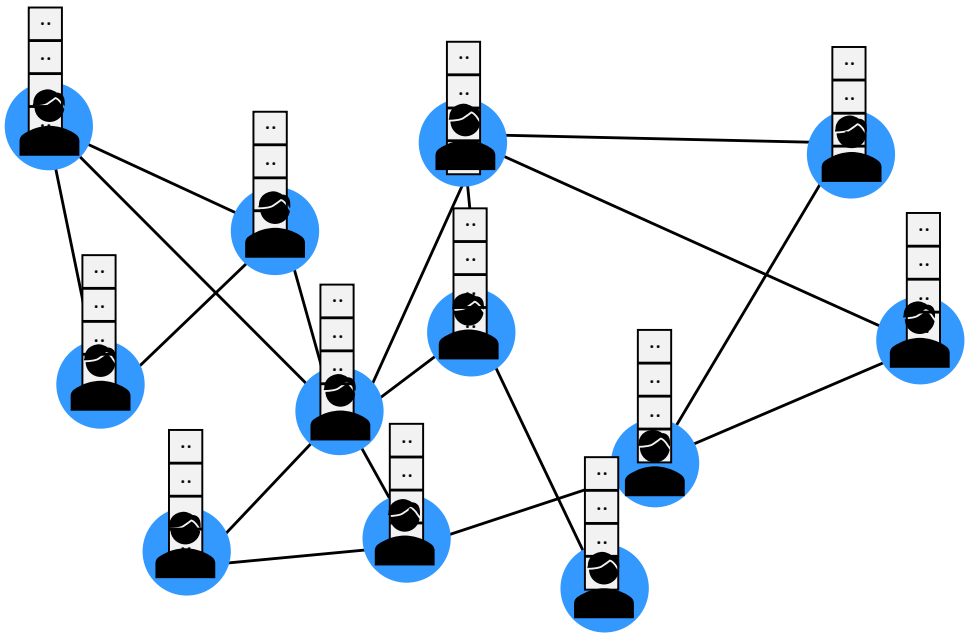
Neighborhood Prediction
Method

- Neighbors have similar taste
- Compute weighted average

$$R_{a,i} = \frac{1}{\mathcal{N}} \sum W_{ab} R_{b,i}$$

- Tune by graph design

User rating graph



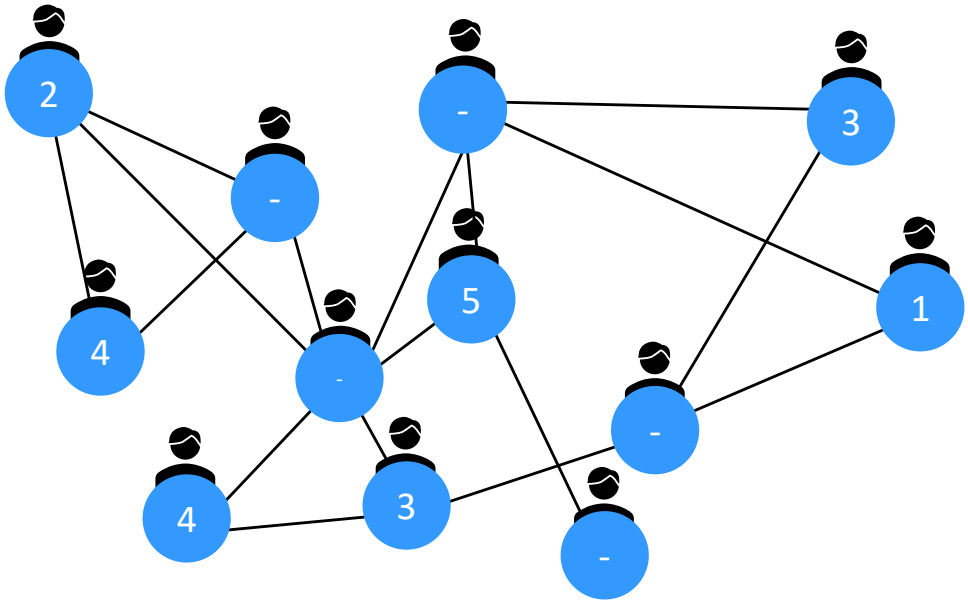
Neighborhood Prediction
Method

- Neighbors have similar taste
- Compute weighted average

$$R_{a,i} = \frac{1}{\mathcal{N}} \sum W_{ab} R_{b,i}$$

- Tune by graph design

User rating graph



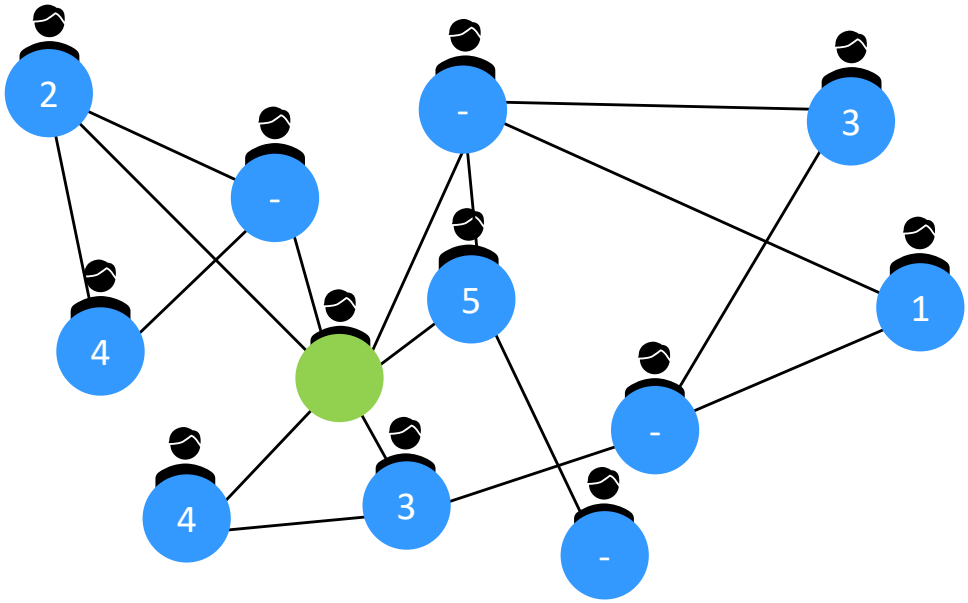
Neighborhood Prediction
Method

- Neighbors have similar taste
- Compute weighted average

$$R_{a,i} = \frac{1}{\mathcal{N}} \sum W_{ab} R_{b,i}$$

- Tune by graph design

User rating graph



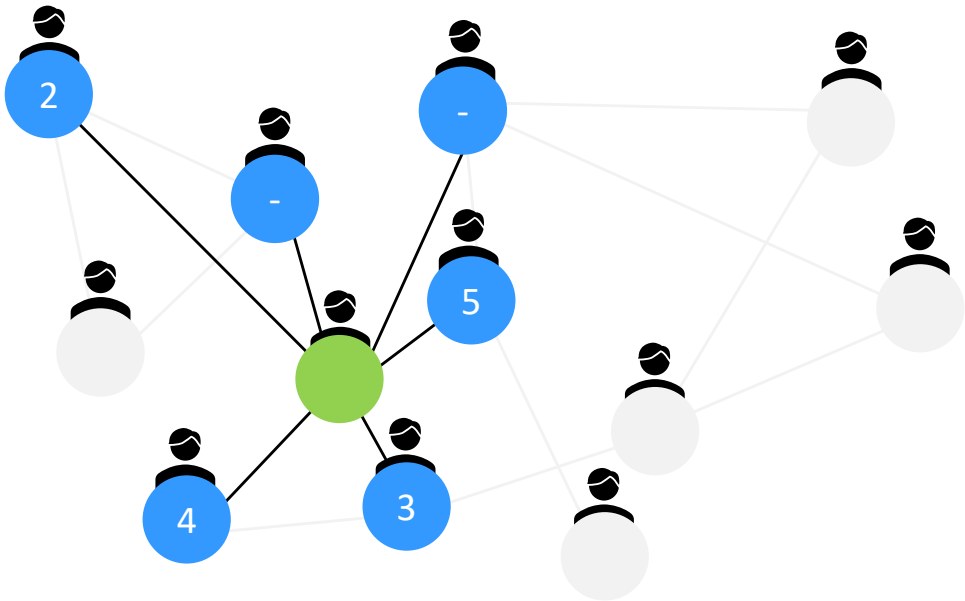
Neighborhood Prediction
Method

- Neighbors have similar taste
- Compute weighted average

$$R_{a,i} = \frac{1}{\mathcal{N}} \sum W_{ab} R_{b,i}$$

- Tune by graph design

User rating graph



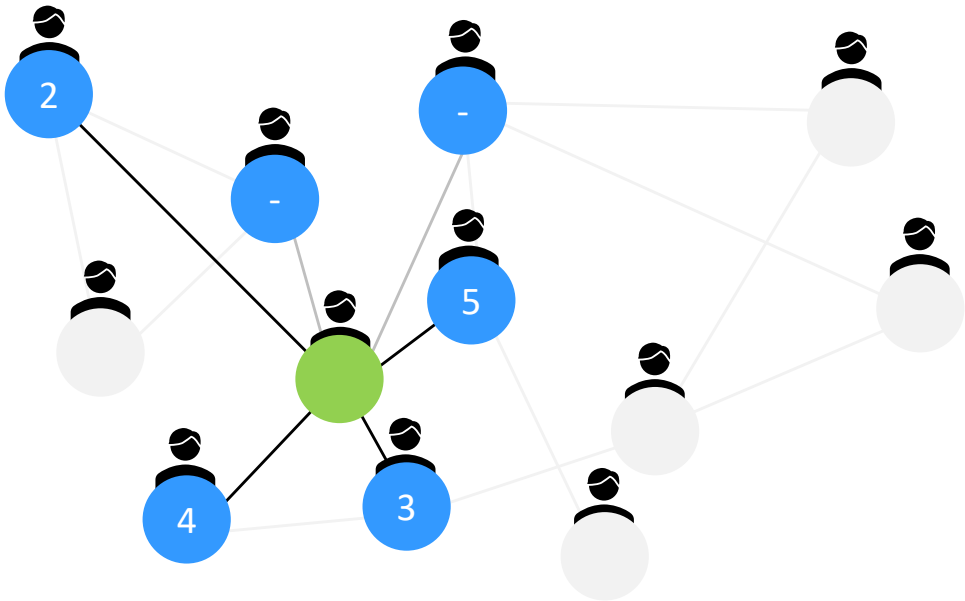
Neighborhood Prediction
Method

- Neighbors have similar taste
- Compute weighted average

$$R_{a,i} = \frac{1}{\mathcal{N}} \sum W_{ab} R_{b,i}$$

- Tune by graph design

User rating graph



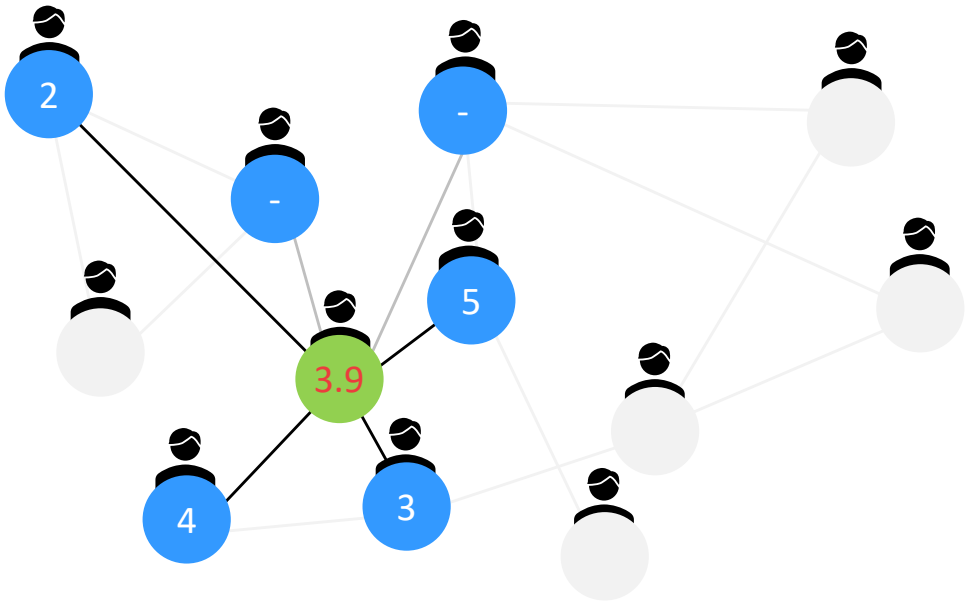
Neighborhood Prediction
Method

- Neighbors have similar taste
- Compute weighted average

$$R_{a,i} = \frac{1}{\mathcal{N}} \sum W_{ab} R_{b,i}$$

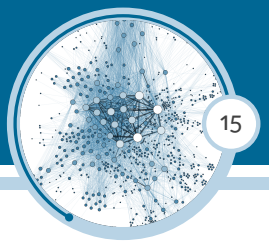
- Tune by graph design

User rating graph



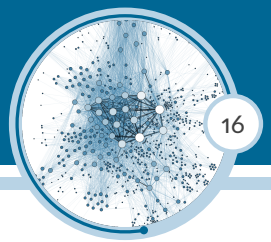
Neighborhood Prediction

Results



- Better than TVreg
- Train / test set split 70% / 30%
- Low computation load
- RMSE : **0.976**
- Further graph filtering: no further improvement

Matrix Factorization
Method



Factorize matrix $R \in \mathbb{R}^{(n \times m)}$ into two matrices $q \in \mathbb{R}^{(m \times f)}$ and $p \in \mathbb{R}^{(n \times f)}$

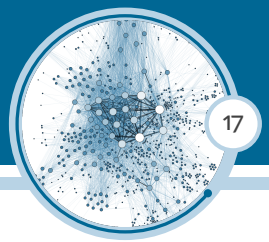
Predicted ratings:

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

Parameters b_i, b_u, q_i and p_u are computed by optimizing:

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

Matrix Factorization
Results

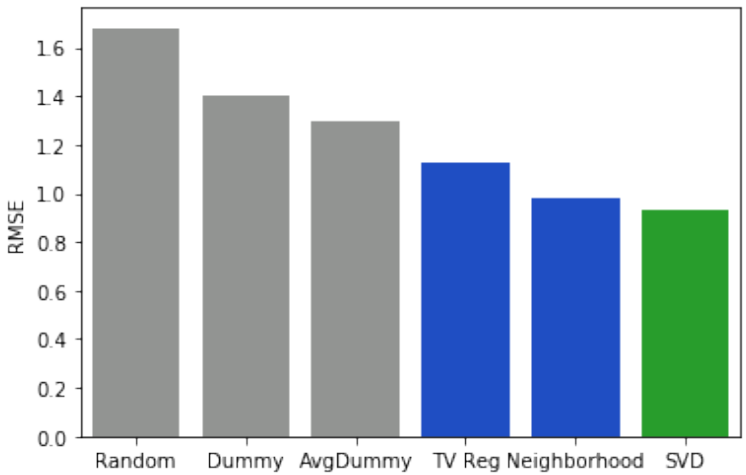


- Using the `surprise` library which implements SVD
- Train / test split of 80% / 20%
- Best number of latent factors : 13
- Low computation load
- RMSE on test set : **0.93**

Part IV : Conclusion

Conclusion

Overview on the errors



- RMSE of network prediction gets close to benchmark
- Movie features not diverse enough
- TVreg good for not too sparse signal
- More data necessary

Outlook

- Use NLP on written reviews
- Combine movie ratings and movie feature graphs

Thanks for your attention !