

Alternative Similarity Functions for Graph Kernels

Jérôme Kunegis, Christian Bauckhage

DAI-Labor, Technische Universität Berlin

19th International Conference on Pattern Recognition

Tampa, December 2008



CC IRML
Information Retrieval
& Machine Learning

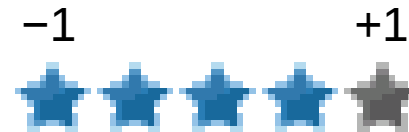


Outline

1. Collaborative Filtering
2. Graph Kernels
3. Similarity Functions
4. Evaluation
5. Conclusion

1. Collaborative Filtering – Problem Formulation

Users rate items



- Task: Predict ratings



Examples: Amazon, Netflix, ...



Users/ Items	U_1	U_2	U_3	U_4	U_5
I_1	+1	+1	+1	?	+1
I_2	-1	-1	?	-1	+1
I_3	?	+1	-1	+1	?

1. Collaborative Filtering – Algebraic Representation

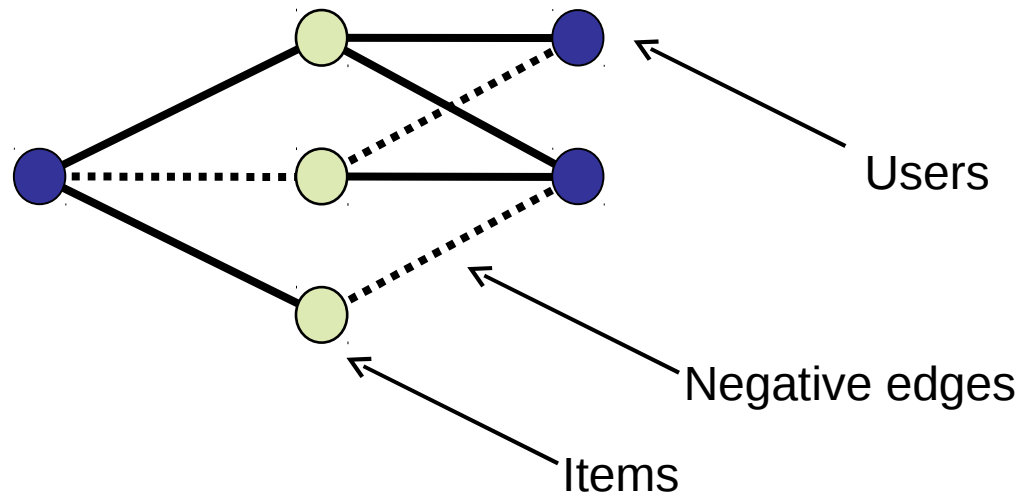
- Represent ratings as a sparse matrix R
- Ratings are signed: positive and negative values

$R =$

Users/ Items	U_1	U_2	U_3	U_4	U_5
I_1	+1	+1	+1	?	+1
I_2	-1	-1	?	-1	+1
I_3	?	+1	-1	+1	?

1. Collaborative Filtering – Bipartite Rating Graph

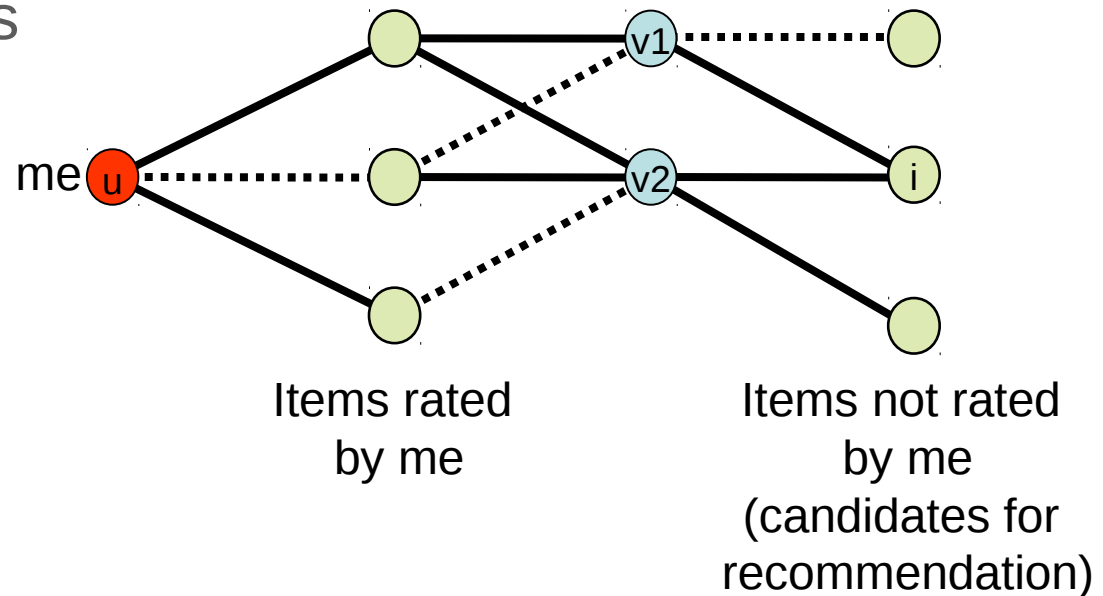
Ratings as a graph



- Rating graph is bipartite and undirected
- Edges are weighted by signed values
- Adjacency matrix A given by $A = \begin{bmatrix} 0 & R \\ R' & 0 \end{bmatrix}$
- Degree matrix $D = \text{diag}(\text{sum}(A))$

1. Collaborative Filtering – Prediction Algorithm

Recommend items



For rating prediction, compute weighted mean of known ratings by other users, weighted by user similarity:

$$P(u,i) = (\sum_v \text{sim}(u,v))^{-1} \sum_v \text{sim}(u,v) R_{vi}$$

2. Graph Kernels – Definition

- Function $\text{sim}(u,v)$ is mostly a **kernel**: symmetric and positive (semi-)definite
- Define graph kernels using the adjacency matrix A :

$$K_{\text{FOR}} = (I + D - A)^{-1} \quad (\text{forest kernel})$$

$$K_{\text{EXP}} = \exp(\alpha A) = \sum_{i=0}^{\infty} \alpha^i / i! A^i \quad (\text{exponential kernel})$$

$$K_{\text{NEU}} = (I - \alpha A)^{-1} = \sum_{i=0}^{\infty} \alpha^i A^i \quad (\text{von Neumann kernel})$$

for $0 < \alpha < 1$

Other, similar variants exist.

3. Similarity Functions

- Graph kernels can be used as a similarity function

$$\text{sim}(u,v) = K_{uv}$$

- To define a distance based on K , write $K = UU'$ and define

$$d(u,v)^2 = (U_u - U_v)^2 = K_{ii} + K_{jj} - K_{ij} - K_{ji}$$

- Use this distance to define alternative similarity functions:

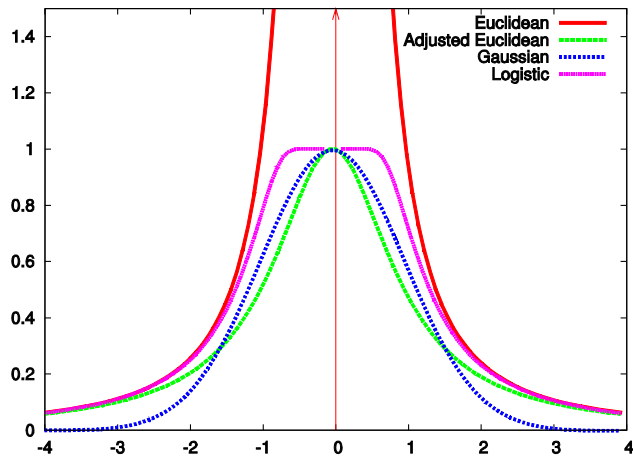
$$\text{sim}_{\text{Eu}} = \sigma^2 / d^2 \quad (\text{inverted Euclidean distance})$$

$$\text{sim}_{\text{EuA}} = 1 / (1 + d^2 / \sigma^2) \quad (\text{adjusted Euclidean distance})$$

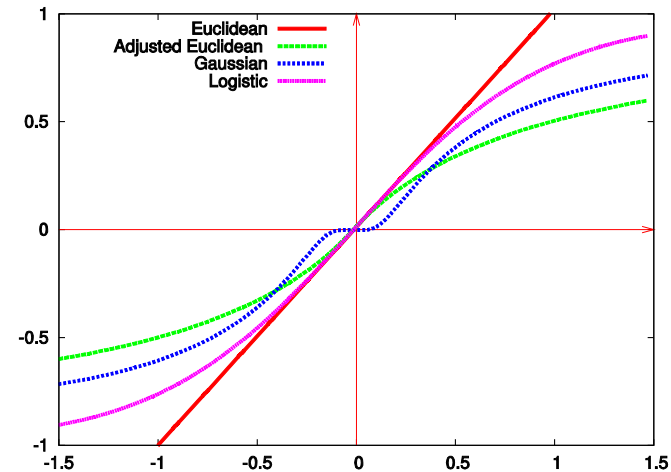
$$\text{sim}_{\text{Ga}} = \exp(-1/2 d^2 / \sigma^2) \quad (\text{Gaussian kernel})$$

The parameter σ can be thought of as a scale parameter

3. Similarity Functions – Comparison



(a) d



(b) $1/d$

Show the similarity function in function of d and $1/d$

- For comparison, show $\text{sim}_{\text{Sigmoid}} = \tanh(\sigma^2 / d^2)$
- Note: Gaussian function is double sigmoid in function of $1/d$
- Note: Unadjusted Euclidean gives unbounded weight for small distances

4. Evaluation

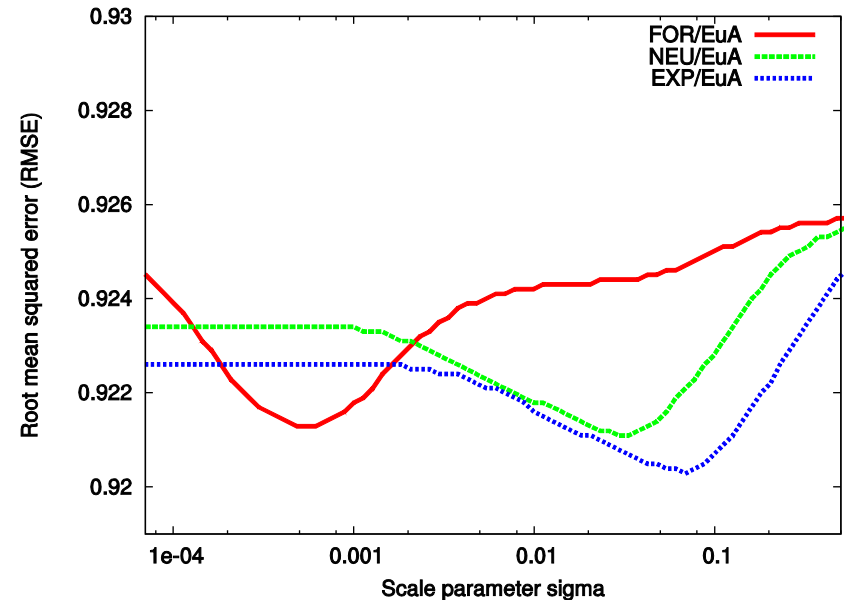
Use the Netflix Prize corpus of user-movie ratings

- Subset of 3,216 users, 1,307 movies and 57,507 ratings
- Compute the Root mean squared error (RMSE)
- Cross-validation

4. Evaluation – (Adjusted) Euclidean

RMSE for adjusted Euclidean in function of σ for all three graph kernels.

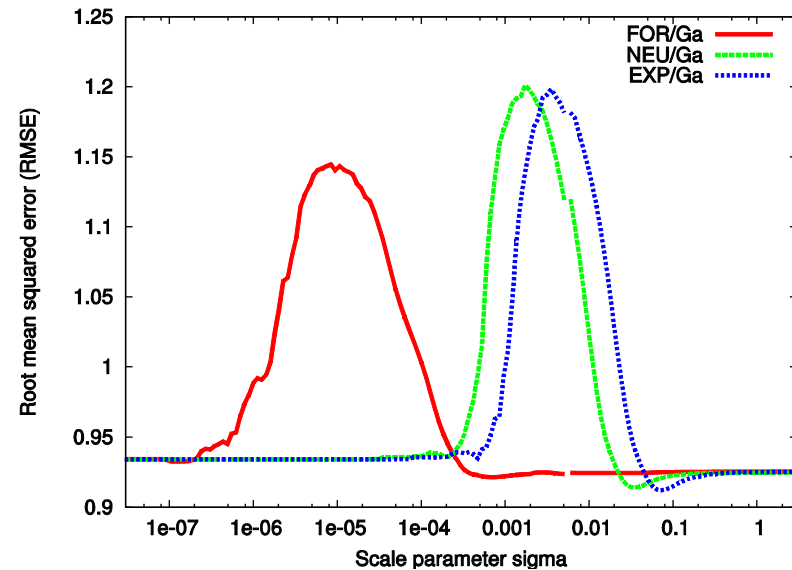
- For small σ , corresponds to unadjusted Euclidean
- For large σ , corresponds to uniform weighting
- Best performance for specific values of σ
- Exponential kernel is best



4. Evaluation – Gaussian

RMSE for Gaussian similarity in function of σ for all three graph kernels.

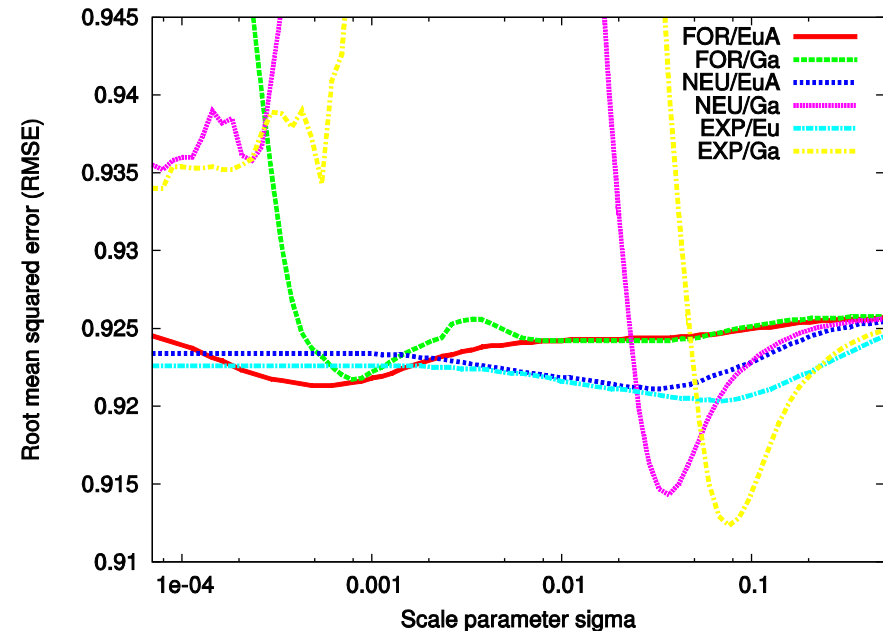
- For large σ , corresponds to uniform weighting



- Best performance for specific values of σ
- Exponential kernel is best
- Very bad performance for many values of σ

4. Evaluation – Overall Results

RMSE for all



- Best performance by Gaussian similarity when tuned properly
- Exponential kernel is best

5. Conclusion

Summary:

- Use the distance spanned by graph kernels
- Have to adjust σ for good results

Future work:

- Estimate σ in closed form
- Other recommendation scenarios
- Use directly (no weighted mean)

THANK YOU