[2023.01.22]

KAIST Department of Industrial & Systems Engineering

Social Recommendation using Probabilistic Matrix Factorization(CIKM'08) & Recommender Systems with Social Regularization(WSDM'11)

서지민

ISysE KAIST Industrial & Systems Engineering

CONTENTS

[SOREC] INTRODUCTION

Trust
: user가 다른 user를 신뢰하는 정도
-Local: personal & subjective
-Global: community가 한 user를 trust하는 정도

- **Inherent weaknesses of collaborative filtering**

1. Due to the sparsity of the user-item rating matrix,
   (memory-based) CF cannot handle users who have never rated items (Cold-start)
2. In reality, users always turn to friends we trust for movie, music or book recommendations.
3. User's tastes and characters can be easily affected by the company we keep.

=> User가 independent and identically distribute되었다고 가정하는
전통적인 recommender system은 적합하지 않음

=> Fuse a user's social network graph with the user-item rating matrix
: Integrate social network structure and the user-item rating matrix
based on probabilistic factor analysis

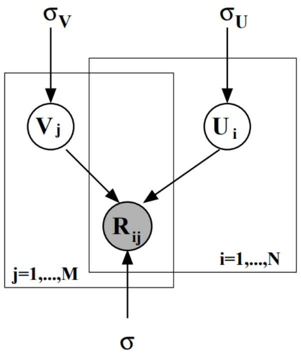
[SOREC] RELATED WORK

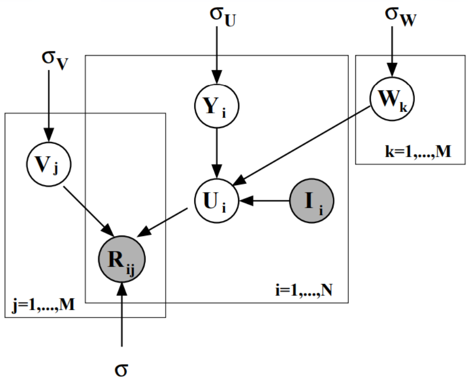
▪ Trust-based recommender systems

[1][14]와 같이 Trust values 와 Similarity 정보를 활용한 모델이 있으나 memory-based method이므로 large dataset에 적합하지 않음

▪ PMF

Rating이 주어진 상황에서 U, V의 확률 최대가 되게 하는 U, V latent feature space 찾기

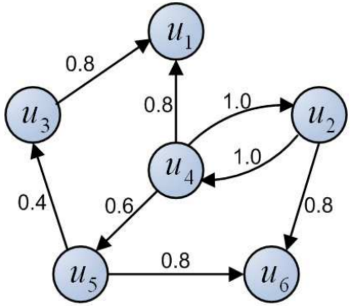




Learn the user/item latent feature space by employing a user social network and a user-item matrix simultaneously

[1] P. Bedi, H. Kaur, and S. Marwaha. Trust based recommender system for semantic web. In IJCAI'07: Proceedings of International Joint Conferences on Artificial Intelligence, pages 2677–2682, 2007
[14] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In Proceedings of CoopIS/DOA/ODBASE, pages 492–508, 2004.

[SOREC] SOCIAL RECOMMENDATION FRAMEWORK



(a) Social Network Graph

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5	2		3		4		
u_2	4	3			5			
u_3	4		2				2	4
u_4								
u_5	5	1	2		4	3		
u_6	4	3		2	4		3	5

(b) User-Item Matrix

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5	2	2.5	3	4.8	4	2.2	4.8
u_2	4	3	2.4	2.9	5	4.1	2.6	4.7
u_3	4	1.7	2	3.2	3.9	3.0	2	4
u_4	4.8	2.1	2.7	2.6	4.7	3.8	2.4	4.9
u_5	5	1	2	3.4	4	3	1.5	4.6
u_6	4	3	2.9	2	4	3.4	3	5

(c) Predicted User-Item Matrix

- **Factorize the social network graph and user-item matrix**

U: user latent feature space

Z: factor matrix in the social network graph

V: item latent feature space

$C \sim U^T Z$: Social network matrix

$R \sim U^T V$: Rating matrix

[SOREC] SOCIAL RECOMMENDATION FRAMEWORK

- **User-Item Matrix Factorization**

$$p(R|U,V,\sigma_R^2) = \prod_{i=1}^{m}\prod_{j=1}^{n}\mathcal{N}\left[\left(r_{ij}|g(U_i^T V_j),\sigma_R^2\right)\right]^{I_{ij}^R}$$

계산 가능한 확률

$p(U,V|R,\sigma_R^2,\sigma_U^2,\sigma_V^2)$ 구하고자 하는 확률

$$\begin{aligned}
&\propto \quad p(R|U,V,\sigma_R^2)p(U|\sigma_U^2)p(V|\sigma_V^2)\\
&= \quad \prod_{i=1}^{m}\prod_{j=1}^{n}\mathcal{N}\left[\left(r_{ij}|g(U_i^T V_j),\sigma_R^2\right)\right]^{I_{ij}^R}\\
&\times \quad \prod_{i=1}^{m}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}) \times \prod_{j=1}^{n}\mathcal{N}(V_j|0,\sigma_V^2\mathbf{I}).
\end{aligned}$$

m: # of user n: # of item

$$p(U|\sigma_U^2) = \prod_{i=1}^{m}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}),$$

$$p(V|\sigma_V^2) = \prod_{j=1}^{n}\mathcal{N}(V_j|0,\sigma_V^2\mathbf{I}).$$

Range [0, 1]

$$f(x) = (x-1)/(R_{max}-1)$$

$$g(x) = 1/(1+\exp(-x)),$$

[SOREC] SOCIAL RECOMMENDATION FRAMEWORK

- **Social Network Matrix Factorization**

$$p(C|U,Z,\sigma_C^2) = \prod_{i=1}^{m}\prod_{k=1}^{m}\mathcal{N}\left[\left(c_{ik}|g(U_i^T Z_k),\sigma_C^2\right)\right]^{I_{ik}^C}$$

계산 가능한 확률

$$p(U,Z|C,\sigma_C^2,\sigma_U^2,\sigma_Z^2)$$ 구하고자 하는 확률

$$\begin{aligned}
&\propto \quad p(C|U,Z,\sigma_C^2)p(U|\sigma_U^2)p(Z|\sigma_Z^2) \\
&= \quad \prod_{i=1}^{m}\prod_{k=1}^{m}\mathcal{N}\left[\left(c_{ik}|g(U_i^T Z_k),\sigma_C^2\right)\right]^{I_{ik}^C} \\
&\times \quad \prod_{i=1}^{m}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}) \times \prod_{k=1}^{m}\mathcal{N}(Z_k|0,\sigma_Z^2\mathbf{I}).
\end{aligned}$$

m명의 user

$$p(U|\sigma_U^2) = \prod_{i=1}^{m}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}),$$

$$p(Z|\sigma_Z^2) = \prod_{k=1}^{m}\mathcal{N}(Z_k|0,\sigma_Z^2\mathbf{I}).$$

$$p(C|U,Z,\sigma_C^2) = \prod_{i=1}^{m}\prod_{j=1}^{n}\mathcal{N}\left[\left(c_{ik}^*|g(U_i^T Z_k),\sigma_C^2\right)\right]^{I_{ik}^C}$$

$$c_{ik}^* = \sqrt{\frac{d^-(v_k)}{d^+(v_i)+d^-(v_k)}} \times c_{ik},$$

[SOREC] SOCIAL RECOMMENDATION FRAMEWORK

- **Matrix Factorization for Social Recommendation**

$$f(x)=\frac{1}{\sqrt{2\pi}\,\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\begin{aligned}
&\ln p(U,V,Z|C,R,\sigma_C^2,\sigma_R^2,\sigma_U^2,\sigma_V^2,\sigma_Z^2) = \\
&\quad -\frac{1}{2\sigma_R^2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R (r_{ij} - g(U_i^T V_j))^2 \\
&\quad -\frac{1}{2\sigma_C^2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^C (c_{ik}^* - g(U_i^T Z_k))^2 \\
&\quad -\frac{1}{2\sigma_U^2}\sum_{i=1}^{m} U_i^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{n} V_j^T V_j - \frac{1}{2\sigma_Z^2}\sum_{k=1}^{m} Z_k^T Z_k \\
&\quad -\frac{1}{2}\left(\left(\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R\right)\ln\sigma_R^2 + \left(\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^C\right)\ln\sigma_C^2\right) \\
&\quad -\frac{1}{2}\left(m\ln\sigma_U^2 + n\ln\sigma_V^2 + m\ln\sigma_Z^2\right) + \mathcal{C}, \qquad (8)
\end{aligned}$$

최종 objective function

$$\begin{aligned}
&\mathcal{L}(R,C,U,V,Z) = \\
&\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R (r_{ij} - g(U_i^T V_j))^2 + \frac{\lambda_C}{2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^C (c_{ik}^* - g(U_i^T Z_k))^2 \\
&+\frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2, \qquad (9)
\end{aligned}$$

Maximize (8)= Minimize (9)

=> (9)를 Loss function으로 사용 가능

[SOREG] INTRODUCTION

- **Inherent weaknesses of Trust-aware recommender systems**

1. Trust relationships ≠ Social friendships (mutual)

=> "Social recommendation"이 기대하는 Social friend network에서의 추천과는 다름

=> Trust-aware recommender는 "Social recommendation"을 대표한다고 볼 수 없음

2. They are based on the assumption that users have similar tastes with other users they trust

=> Friends가 매우 큰 user에 대해서는 틀린 가정일 수 있음

3. Interacting with real friends is the most attractive activity on the Web

=> Utilize social information to improve the prediction accuracy of traditional recommender systems

: Employ two social regularization terms

to constrain the matrix factorization objective function

[SOREG] RELATED WORK

- **Trust-based Recommender Systems**

[24] Compute Trust value in addition to similarity between users

[23] Factor analysis method(SoRec)

=> Physical interpretation이 부족

- **Social Recommender Systems**

[20] Add regularization constraints to the text-based predictor

$$\Omega_2(\mathbf{w}) = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left(\mathbf{w}^T\mathbf{r}_i - q_i\right)^2 + \alpha\mathbf{w}^T\mathbf{w} \boxed{+ \beta\sum_{i<j}\mathbf{A}_{ij}\left(\mathbf{w}^T\mathbf{r}_i - \mathbf{w}^T\mathbf{r}_j\right)^2}$$

본 논문에서 “Social Recommender Systems”는 social friends network를 활용하여 Recommender System을 개선시키는 것

“Social Recommendation”이라고 불렸던 기존의 연구들은 실제로 social network 정보를 사용하지 않았거나, 사실은 trust-aware metho이거나 매우 간단한 heuristic만을 활용함

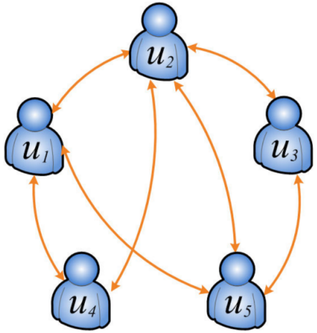
=> Systematically analyze social recommendation problem based on MF

Elaborate the detailed differences between social and trust-aware recommender

[20] Y. Lu, P. Tsaparas, A. Ntoulas, and L. Polanyi. Exploiting social context for review quality prediction. In Proc. of WWW ’10, pages 691–700, North Carolina, USA, 2010
[24] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In Proceedings of CoopIS/DOA/ODBASE, pages 492–508, 2004.

[SOREG] PROBLEM DEFINITION

(a) Real World Social Recommendation



(b) Social Network

	v_1	v_2	v_3	v_4	v_5
u_1	1		2	3	
u_2		3			1
u_3		4		5	
u_4	5			4	
u_5		2	5		4

(c) User-Item Rating Matrix

- **Friend network**

The edges in social friend network are bidirectional (Trust-network와의 차이)

[SOREG] SOCIAL RECOMMENDATION FRAMEWORK

▪ Model1: Average-based Regularization

$$
\begin{aligned}
\min_{U,V} \mathcal{L}_1(R,U,V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - U_i^T V_j)^2 \\
&\boxed{+ \frac{\alpha}{2}\sum_{i=1}^{m} \|U_i - \frac{1}{|\mathcal{F}^+(i)|}\sum_{f\in\mathcal{F}^+(i)} U_f\|_F^2} \\
&+ \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2, \qquad (5)
\end{aligned}
$$

F+(i) : ui's outlink friends
F-(i) : ui's inlink friends

User의 모든 친구가 비슷한 taste를 갖는다는 보장이 없으므로,
User와의 유사성을 고려하여 계산한 가중평균

$$
\frac{\alpha}{2}\sum_{i=1}^{m} \|U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)}\|_F^2,
$$

최종 objective function

$$
\begin{aligned}
\min_{U,V} \mathcal{L}_1(R,U,V) = &\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - U_i^T V_j)^2 \\
&+ \frac{\alpha}{2}\sum_{i=1}^{m} \|U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)}\|_F^2, \\
&+ \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2. \qquad (8)
\end{aligned}
$$

한 User의 친구가 매우 다양한 taste를 가졌을 때 부적절

[SOREG] SOCIAL RECOMMENDATION FRAMEWORK

▪ Model2: Individual-based Regularization

$$\begin{aligned}\min_{U,V} \mathcal{L}_2(R,U,V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij}-U_i^T V_j)^2 \\ &+ \boxed{\frac{\beta}{2}\sum_{i=1}^{m}\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\|U_i-U_f\|_F^2} \\ &+ \lambda_1\|U\|_F^2+\lambda_2\|V\|_F^2. \qquad (11)\end{aligned}$$

Advantage

1) Diverse tasted를 가진 friends들이 많은 User도 표현 가능
2) 간접적으로 taste propagation이 가능

U_i와 U_f가 friend이고, U_f와 U_g가 friend인 경우 U_i와 U_g사이의 distance도 minimize

$$Sim(i,f)||U_i-U_f||_F^2 \text{ and } Sim(f,g)||U_f-U_g||_F^2.$$

[SOREG] SIMILARITY BETWEEN TWO USERS

- **VSS(Vector Space Similarity)**

$$Sim(i,f) = \frac{\sum_{j \in I(i) \cap I(f)} R_{ij} \cdot R_{fj}}{\sqrt{\sum_{j \in I(i) \cap I(f)} R_{ij}^2} \cdot \sqrt{\sum_{j \in I(i) \cap I(f)} R_{fj}^2}},$$

- **PCC(Pearson Correlation Coefficient)**

$$Sim(i,f) = \frac{\sum_{j \in I(i) \cap I(f)} (R_{ij} - \overline{R}_i) \cdot (R_{fj} - \overline{R}_f)}{\sqrt{\sum_{j \in I(i) \cap I(f)} (R_{ij} - \overline{R}_i)^2} \cdot \sqrt{\sum_{j \in I(i) \cap I(f)} (R_{fj} - \overline{R}_f)^2}},$$

Difference

VSS는 user간 different rating style 고려 X

PCC의 경우, Sim(I, f)의 range [-1, 1]
본 논문에서는 f(x) = (x + 1) / 2를 이용해서
PCC의 range를 [0, 1]로

METRIC

$$MAE = \frac{1}{T}\sum_{i,j}|R_{ij} - \widehat{R}_{ij}|,$$

T: number of tested ratings

$$RMSE = \sqrt{\frac{1}{T}\sum_{i,j}(R_{ij} - \widehat{R}_{ij})^2}.$$

DATASETS

- **Douban**

: Ratings + Friends (대부분 offline으로도 아는 friends)

- **Epinions**

: Ratings + “trust list”

[SOREC] EXPERIMENTAL ANALYSIS

MAE comparison with other approaches (Dataset: Epinions)

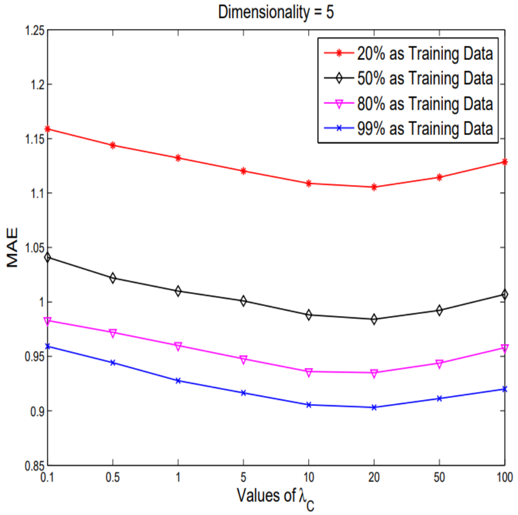
Training Data	Dimensionality = 5				Dimensionality = 10			
	MMMF	PMF	CPMF	SoRec	MMMF	PMF	CPMF	SoRec
99%	1.0008	0.9971	0.9842	**0.9018**	0.9916	0.9885	0.9746	**0.8932**
80%	1.0371	1.0277	0.9998	**0.9321**	1.0275	1.0182	0.9923	**0.9240**
50%	1.1147	1.0972	1.0747	**0.9838**	1.1012	1.0857	1.0632	**0.9751**
20%	1.2532	1.2397	1.1981	**1.1069**	1.2413	1.2276	1.1864	**1.0944**

[SOREC] EXPERIMENTAL ANALYSIS

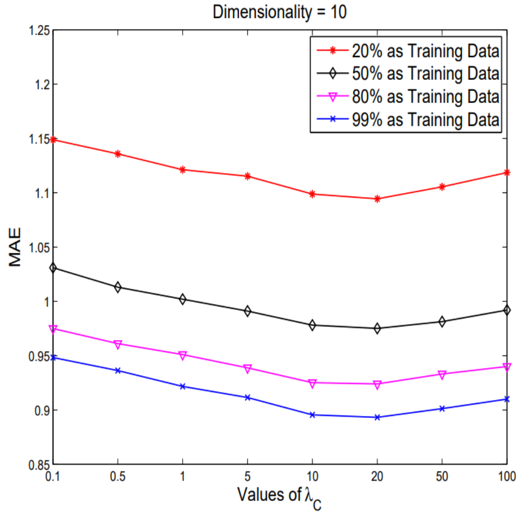
Impact of Parameter λ_C

$\lambda_C = 0$
: only mine user-item rating matrix
$\lambda_C = inf$
: only extract information from the social network

$$\mathcal{L}(R, C, U, V, Z) =$$
$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(r_{ij} - g(U_i^T V_j))^2 + \boxed{\frac{\lambda_C}{2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^{C}(c_{ik}^{*} - g(U_i^T Z_k))^2}$$
$$+ \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2, \quad (9)$$



(a) Dimensionality=5



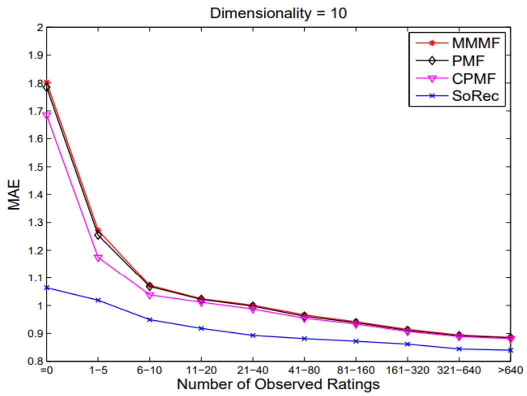
(b) Dimensionality=10

Best performance when $\lambda_C \in [10, 20]$

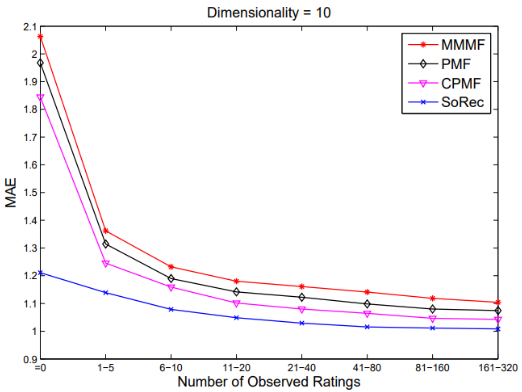
=> fusing two resources together generate better performance

[SOREC] EXPERIMENTAL ANALYSIS

Performance on Different Users



(a) Performance Comparison on Different User Rating Scales (99% as Training Data)



(g) Performance Comparison on Different User Rating Scales (20% as Training Data)

SoRec performs much better when users have no rating records("=0")

[SOREG] EXPERIMENTAL ANALYSIS

Performance Comparisons

Table 5: Performance Comparisons (Dimensionality = 10)

Dataset	Training	Metrics	UserMean	ItemMean	NMF	PMF	RSTE	$SR1_{vss}$	$SR1_{pcc}$	$SR2_{vss}$	$SR2_{pcc}$
Douban	80%	MAE	0.6809	0.6288	0.5732	0.5693	0.5643	0.5579	0.5576	0.5548	**0.5543**
		Improve	18.59%	11.85%	3.30%	2.63%	1.77%				
		RMSE	0.8480	0.7898	0.7225	0.7200	0.7144	0.7026	0.7022	0.6992	**0.6988**
		Improve	17.59%	11.52%	3.28%	2.94%	2.18%				
	60%	MAE	0.6823	0.6300	0.5768	0.5737	0.5698	0.5627	0.5623	0.5597	**0.5593**
		Improve	18.02%	11.22%	3.03%	2.51%	1.84%				
		RMSE	0.8505	0.7926	0.7351	0.7290	0.7207	0.7081	0.7078	0.7046	**0.7042**
		Improve	17.20%	11.15%	4.20%	3.40%	2.29%				
	40%	MAE	0.6854	0.6317	0.5899	0.5868	0.5767	0.5706	0.5702	0.5690	**0.5685**
		Improve	17.06%	10.00%	3.63%	3.12%	1.42%				
		RMSE	0.8567	0.7971	0.7482	0.7411	0.7295	0.7172	0.7169	0.7129	**0.7125**
		Improve	16.83%	10.61%	4.77%	3.86%	2.33%				
Epinions	90%	MAE	0.9134	0.9768	0.8712	0.8651	0.8367	0.8290	0.8287	0.8258	**0.8256**
		Improve	9.61%	15.48%	5.23%	4.57%	1.33%				
		RMSE	1.1688	1.2375	1.1621	1.1544	1.1094	1.0792	1.0790	1.0744	**1.0739**
		Improve	8.12%	13.22%	7.59%	6.97%	3.20%				
	80%	MAE	0.9285	0.9913	0.8951	0.8886	0.8537	0.8493	0.8491	0.8447	**0.8443**
		Improve	9.07%	14.83%	5.68%	4.99%	1.10%				
		RMSE	1.1817	1.2584	1.1832	1.1760	1.1256	1.1016	1.1013	1.0958	**1.0954**
		Improve	7.30%	12.95%	7.42%	6.85%	2.68%				

[SOREG] EXPERIMENTAL ANALYSIS

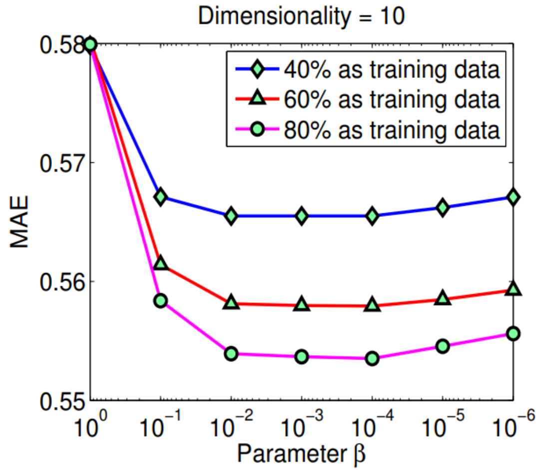
Impact of Parameter α *and β*

$\alpha = \beta = 0$

: only mine user-item rating matrix

$\alpha = \beta = inf$

: only extract information from the social network



(a) Douban (MAE)

$$\begin{aligned}\min_{U,V} \mathcal{L}_1(R,U,V) = &\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - U_i^T V_j)^2 \\ &+ \boxed{\frac{\alpha}{2}}\sum_{i=1}^{m} \|U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f) \times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)}\|_F^2, \\ &+ \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2. \qquad (8)\end{aligned}$$

$$\begin{aligned}\min_{U,V} \mathcal{L}_2(R,U,V) = &\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - U_i^T V_j)^2 \\ &+ \boxed{\frac{\beta}{2}}\sum_{i=1}^{m}\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\|U_i - U_f\|_F^2 \\ &+ \lambda_1\|U\|_F^2 + \lambda_2\|V\|_F^2. \qquad (11)\end{aligned}$$

CONCLUSION

[SOREC]

Outperforms the other state-of-the-art collaborative filtering algorithms
Scalable to very large datasets

In the future, need to consider distrust information

[SOREG]

Actually utilize all the social connections of each user

Only constrain user feature vectors while ignoring the item side

In the future, need to design an effective algorithm to identify the most suitable group of friends for different recommendation task

SOREC

$$\mathcal{L}(R,C,U,V,Z) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(r_{ij}-g(U_i^T V_j))^2 + \frac{\lambda_C}{2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^{C}(c_{ik}^{*}-g(U_i^T Z_k))^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2, \quad (9)$$

Trust matrix 완성
Missing value handle 가능

Model-based

SOREG

$$\begin{aligned} \min_{U,V} \mathcal{L}_2(R,U,V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - U_i^T V_j)^2 \\ &+ \frac{\beta}{2}\sum_{i=1}^{m}\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\|U_i - U_f\|_F^2 \\ &+ \lambda_1\|U\|_F^2 + \lambda_2\|V\|_F^2. \quad (11) \end{aligned}$$

Collaborative filtering process informed by the reputation of users

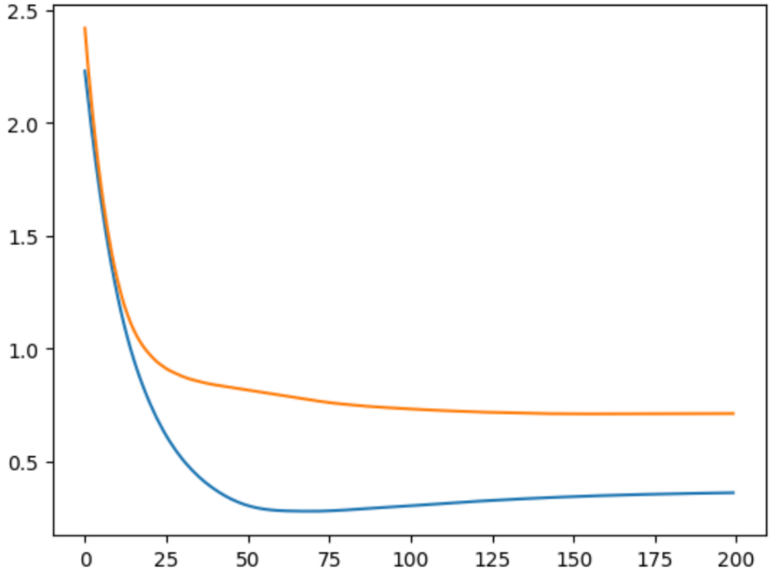
Memory-based

IMPLEMENTATION
SOREC

```python
class SOREC(nn.Module):
  def __init__(self, train_R = scaled_train_rating_matrix_full, test_R = scaled_test_rating_matrix_full, trust = ui_uk_matrix, l=10, lambC = 10, lambdaUVZ =0.001, learnir
    '''
    SOREC
    논문에서는 gradient descent 직접 업데이트하는 방식
    구현은 pytorch의 loss.backward()
    '''
    super(SOREC, self).__init__()

    # 논문은 특이하게 m이 user 수, n이 item 수
    self.m, self.n = train_R.shape # Rating
    self.test_n, self.test_m = test_R.shape

    self.latent_dimension = l

    self.train_R = train_R # 80          (100, 1000)
    self.test_R = test_R   # 20 비율      (100, 1000)
    self.trust = trust     # trust ratings (100, 100)

    self.lr = learning_rate
    self.epoch = epochs
    self.lambdaC = lambC
    self.lambdaUVZ = lambdaUVZ # 공통으로 사용

    self.U = nn.Parameter(torch.randn(self.latent_dimension, self.m)) # (l, m)
    self.Z = nn.Parameter(torch.randn(self.latent_dimension, self.m)) # (l, m)
    self.V = nn.Parameter(torch.randn(self.latent_dimension, self.n)) # (l, n)

    self.optimizer = torch.optim.Adam(self.parameters(), lr= self.lr)

  def get_complete_matrix(self):
    self.completed_rating_matrix  = torch.matmul(self.U.T, self.V)
    return self.completed_rating_matrix

  def forward_user_item(self, u):
    # user 들어올 때마다 모든 item에 대한 rating 차이 구하기
    predicted_rating = torch.matmul(self.U[:, u].T, self.V) # (l,) (l, n) => (n)
    loss = 0
    for v in range(self.n):
      # R이 있는 경우만
      if self.train_R[u, v]:
        loss += (self.train_R[u, v] - g(predicted_rating[v])) ** 2
        #print("u, v,", u, v, (self.train_R[u, v] - g(predicted_rating[v])) ** 2)

    return loss / 2

  def forward_social(self, u):
    # user 들어올 때마다 trusted user 대한 trust 차이 구하기
    predicted_trust = torch.matmul(self.U[:, u].T, self.Z) # (m)
    loss = 0
    for z in range(self.m):
      if self.trust[u, z]:
        loss += (self.trust[u, z] - g(predicted_trust[z])) ** 2

    return self.lambdaC * loss / 2

  def test_accuracy(self):
    completed_rating_matrix = torch.matmul(self.U.T, self.V)
    mae_loss = 0
    test_num = 0
    for u in range(self.m):
      for v in range(self.n):
        if self.test_R[u, v]:
          test_num += 1 # .2
          mae_loss += abs(completed_rating_matrix[u, v] - self.test_R[u, v])

    return mae_loss / test_num

  def train_accuracy(self):
    completed_rating_matrix = torch.matmul(self.U.T, self.V)
    mae_loss = 0
    train_num = 0
    for u in range(self.m):
      for v in range(self.n):
        if self.train_R[u, v]:
          train_num += 1 # .8
          mae_loss += abs(completed_rating_matrix[u, v] - self.train_R[u, v])

    return mae_loss / train_num


  def fit(self):
    train_loss_list = []
    test_loss_list = []

    for epoch in range(self.epoch):
      total_loss = 0
      for u in range(self.m):

        # user-item matrix loss
        loss1 = self.forward_user_item(u)

        # trust matrix loss
        loss2 = self.forward_social(u)

        total_loss = loss1 + loss2 + self.lambdaUVZ * (torch.sum(self.U ** 2) + torch.sum(self.V ** 2) + torch.sum(self.Z ** 2))

        self.optimizer.zero_grad()
        total_loss.backward()
        self.optimizer.step()

      test_mae = self.test_accuracy()
      train_mae = self.train_accuracy()

      train_loss_list.append(train_mae)
      test_loss_list.append(test_mae)
```

IMPLEMENTATION
SOREC

```
Epoch [0/200], total_loss: 11.483431816101074, train_mae: 2.22940731048584, test_mae: 2.4190123081207275
Epoch [40/200], total_loss: 1.762978434562683, train_mae: 0.37445834279060364, test_mae: 0.838713526725769
Epoch [80/200], total_loss: 1.2129321098327637, train_mae: 0.2844747304916382, test_mae: 0.7525971531867981
Epoch [120/200], total_loss: 0.8773898482322693, train_mae: 0.3227185010910034, test_mae: 0.7190210223197937
Epoch [160/200], total_loss: 0.7156668901443481, train_mae: 0.34875792264938354, test_mae: 0.710475742816925
[<matplotlib.lines.Line2D at 0x7f89322bd7e0>]
```



- Dataset: Epinions
- Hidden dimension: 16
- Epoch: 200
- Max User id: 100
- Max Item id: 100

IMPLEMENTATION-TOY EXAMPLE

```
train_toy = np.array([
    [5, 2, 0, 3, 0, 4, 0, 0],
    [4, 3, 0, 0, 5, 0, 0, 0],
    [4, 0, 2, 0, 0, 0, 2, 4],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [5, 1, 2, 0, 4, 3, 0, 0],
    [4, 3, 0, 2, 4, 0, 3, 5]
], dtype=float)


trust_toy = np.array([
            [0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1.0, 0.8],
            [0.8,0, 0,0,0,0],
            [0.8, 1.0, 0, 0, 0.6, 0],
            [0,0, 0.4, 0, 0, 0.8],
            [0,0,0,0,0,0]])
```

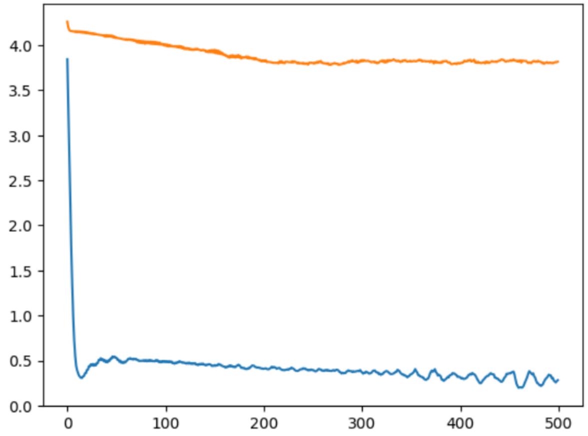
```
tensor([[ 4.7275, -0.0793, -0.7028,  0.2901,  2.8730,  1.9012,  0.1861,  3.1234],
        [ 5.7301,  0.6402, -1.6350, -1.0304,  6.2876,  1.4954,  0.0516,  5.4728],
        [ 4.9503,  0.1597, -1.3526, -0.2928,  3.6354,  1.8273, -0.5242,  4.3511],
        [ 5.2442, -0.2687, -1.4796,  0.0831,  2.7164,  2.0856, -0.8501,  3.8742],
        [ 6.1930, -0.1604, -1.5142, -0.3318,  4.5732,  2.0072, -0.0606,  4.4682],
        [ 4.3795,  0.4742, -0.8959, -0.2419,  3.9223,  1.5596,  0.0589,  3.9636]],
```

- Lambda 작을수록 overfit 돼서 성능이 잘 나올 것이라고 예상했으나, 무관

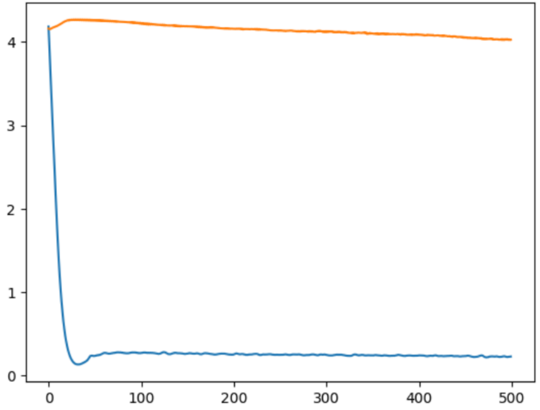
IMPLEMENTATION

SoREG(model1 / model2)

```
Epoch [0/500], total_loss: 13.671034812927246, train_mae: 3.8446009298242947, test_mae: 4.2605112834149095
Epoch [50/500], total_loss: 0.42230188846588135, train_mae: 0.5322291155269637, test_mae: 4.086493924918424
Epoch [100/500], total_loss: 0.5490496158599854, train_mae: 0.49368707679125123, test_mae: 4.010850776151481
Epoch [150/500], total_loss: 0.6898825168609619, train_mae: 0.45021573671571014, test_mae: 3.9485505498556726
Epoch [200/500], total_loss: 0.2687221169471741, train_mae: 0.41076853117590284, test_mae: 3.8322739198983435
Epoch [250/500], total_loss: 0.1870841085910797, train_mae: 0.3963620968829797, test_mae: 3.8047813813849145
Epoch [300/500], total_loss: 0.23465074598789215, train_mae: 0.3690140720471334, test_mae: 3.83563673639872
Epoch [350/500], total_loss: 0.21615494787693024, train_mae: 0.34936119703003404, test_mae: 3.817663966412525
Epoch [400/500], total_loss: 0.334757000207901, train_mae: 0.31295084674999848, test_mae: 3.8014440574799195
Epoch [450/500], total_loss: 0.21815988421440125, train_mae: 0.3594125235590953, test_mae: 3.8261463824046187
[<matplotlib.lines.Line2D at 0x7cf3cdf6d5a0>]
```




```
Epoch [0/500], total_loss: 12.219196319580078, train_mae: 4.182924975680934, test_mae: 4.159323818712349
Epoch [50/500], total_loss: 0.2266174554824829, train_mae: 0.23630272831898255, test_mae: 4.25349640750502
Epoch [100/500], total_loss: 0.25265055894851685, train_mae: 0.27136966690479086, test_mae: 4.215173683013303
Epoch [150/500], total_loss: 0.2430100440979004, train_mae: 0.26470270416616004, test_mae: 4.185150759287149
Epoch [200/500], total_loss: 0.17994911968708038, train_mae: 0.25429686312545596, test_mae: 4.155940166917671
Epoch [250/500], total_loss: 0.17217816412448883, train_mae: 0.24887597050648255, test_mae: 4.133611594816767
Epoch [300/500], total_loss: 0.24058625102043152, train_mae: 0.249089222473857, test_mae: 4.115684417356928
Epoch [350/500], total_loss: 0.1929163783788681, train_mae: 0.24174449044906676, test_mae: 4.098492112983183
Epoch [400/500], total_loss: 0.250471830368042, train_mae: 0.24187292848579614, test_mae: 4.082918098173946
Epoch [450/500], total_loss: 0.14441131055355072, train_mae: 0.23369156061907223, test_mae: 4.048666590188881
[<matplotlib.lines.Line2D at 0x78ed92922170>]
```



IMPLEMENTATION-TOY EXAMPLE

```python
train_toy = np.array([
    [5, 2, 0, 3, 0, 4, 0, 0],
    [4, 3, 0, 0, 5, 0, 0, 0],
    [4, 0, 2, 0, 0, 0, 2, 4],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [5, 1, 2, 0, 4, 3, 0, 0],
    [4, 3, 0, 2, 4, 0, 3, 5]
], dtype=float)


trust_toy = np.array([
            [0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1.0, 1],
            [0.8,0, 0,0,0,0],
            [0.8, 1.0, 0, 0, 0.6, 0],
            [0,0, 0.4, 0, 0, 0.8],
            [0,0,0,0,0,0]])
```

```
tensor([[ 4.9975e+00,  1.9973e+00,  7.7953e-01,  2.9975e+00,  2.2003e+00,
          3.9973e+00, -4.7482e-01,  1.2301e+00],
        [ 4.0000e+00,  2.9972e+00,  1.1869e+00,  1.5208e+00,  4.9968e+00,
          3.2282e+00,  2.1699e+00,  4.2108e+00],
        [ 4.0008e+00,  6.8932e-01,  2.0006e+00,  1.9215e-02,  3.0715e+00,
          2.1926e+00,  2.0005e+00,  4.0002e+00],
        [-8.0774e-04,  4.4515e-03, -6.5338e-03, -9.0913e-03,  1.1973e-02,
          5.0368e-03, -2.3553e-03,  2.2077e-04],
        [ 4.9980e+00,  9.9992e-01,  2.0002e+00,  9.7675e-01,  3.9987e+00,
          2.9988e+00,  2.3078e+00,  3.8891e+00],
        [ 3.9990e+00,  2.9975e+00,  9.7347e-01,  1.9987e+00,  3.9993e+00,
          2.9354e+00,  2.9989e+00,  4.9981e+00]], grad_fn=<MmBackward0>)
```

- Missing value 많을수록 similarity 연산이 어려워져 SoRec이 적절한 경우도 발생

APPENDIX

Recommendation with social information

Trust-aware Collaborative Filtering for Recommender Systems(2004)

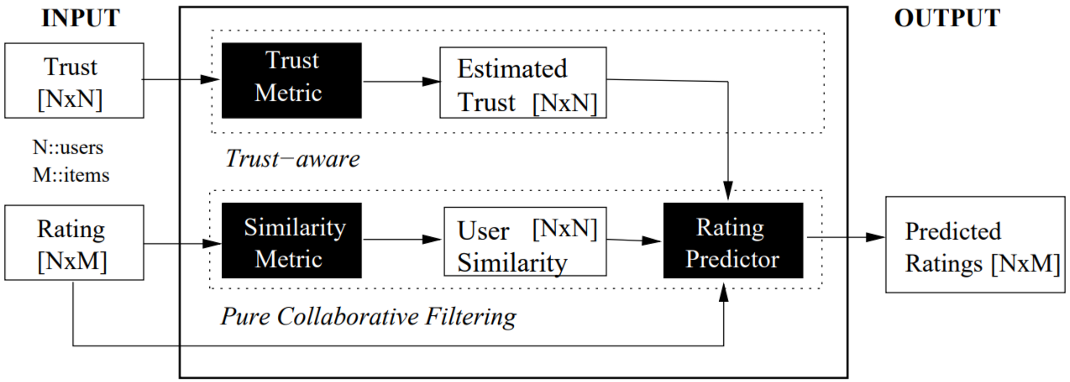


Figure 2. Trust-Aware Recommender Systems Architecture.

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u=1}^{k} w_{a,u}(r_{u,i} - \overline{r}_u)}{\sum_{u=1}^{k} w_{a,u}}$$

U는 user a의 neighbor
User a가 item i에 준 점수
Rating [0, 1]

Similarity matrix에 trust matrix를 더하여 weight로 사용

APPENDIX

Recommendation with social information

SoRec(CIKM'08)

$$\mathcal{L}(R,C,U,V,Z) =$$
$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(r_{ij}-g(U_i^T V_j))^2+\frac{\lambda_C}{2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^{C}(c_{ik}^{*}-g(U_i^T Z_k))^2$$

Learning to Recommend with Social Trust Ensemble (SIGIR'09)

$$\mathcal{L}(R,S,U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - g(\alpha U_i^T V_j + (1-\alpha)\sum_{k\in\mathcal{T}(i)} S_{ik}U_k^T V_j))^2$$

Learning to Recommend with Trust and Distrust Relationships(RecSys'09)

$$\min_{U,V} \mathcal{L}_{\mathcal{D}}(R,S^{\mathcal{D}},U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - g(U_i^T V_j))^2 + \frac{\beta}{2}\sum_{i=1}^{m}\sum_{d\in\mathcal{D}^{+}(i)} (-S_{id}^{\mathcal{D}}\|U_i - U_d\|_F^2)$$

SoReg(WSDM'11)

$$\min_{U,V} \mathcal{L}_1(R,U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\alpha}{2}\sum_{i=1}^{m}\|U_i - \frac{\sum_{f\in\mathcal{F}^{+}(i)} Sim(i,f)\times U_f}{\sum_{f\in\mathcal{F}^{+}(i)} Sim(i,f)}\|_F^2,$$

APPENDIX

Learning to Recommend with Trust and Distrust Relationships(RecSys'09)

Table 4: RMSE Comparison with other popular algorithms. The reported values are the RMSE on the Epinions Dataset achieved from dividing the data into 5%, 10%, and 20% for training data, respectively.

Dataset	Traning Data	Dimensionality	PMF	SoRec	RWD	RWT
Epinions	5%	5D	1.228	1.199	1.186	**1.177**
		10D	1.214	1.198	1.185	**1.176**
	10%	5D	0.990	0.944	0.932	**0.924**
		10D	0.977	0.941	0.931	**0.923**
	20%	5D	0.819	0.788	0.723	**0.721**
		10D	0.818	0.787	0.723	**0.720**

$$\begin{aligned}\min_{U,V} \mathcal{L}_{\mathcal{D}}(R, S^{\mathcal{D}}, U, V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - g(U_i^T V_j))^2 \\ &+ \frac{\beta}{2}\sum_{i=1}^{m}\sum_{d\in\mathcal{D}^+(i)}(-S_{id}^{\mathcal{D}}\|U_i - U_d\|_F^2) \\ &+ \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2. \qquad (3)\end{aligned}$$

$$\begin{aligned}\min_{U,V} \mathcal{L}_{\mathcal{T}}(R, S^{\mathcal{T}}, U, V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - g(U_i^T V_j))^2 \\ &+ \frac{\alpha}{2}\sum_{i=1}^{m}\sum_{t\in\mathcal{T}^+(i)}(S_{it}^{\mathcal{T}}\|U_i - U_t\|_F^2) \\ &+ \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2. \qquad (7)\end{aligned}$$

RWD (Recommendation With Distrust)
RWT (Recommendation With Trust)

distrust information is at least as important as the trust information

RECSYS

2008

SESSION: Social networks and recommenders

SESSION: User studies

SESSION: Conversational systems

2009

SESSION: Tags and social networks

SESSION: Applications

SESSION: Algorithms II

SESSION: Privacy and security

2010

SESSION: Beyond prediction accuracy

SESSION: Algorithms

SESSION: All about groups

SESSION: Recommending in social networks

SESSION: Recommending non-standard items

2011~2013

SESSION: Recommenders and the social web

SESSION: Multi-dimensional recommendation, context-awareness and group recommendation

SESSION: Methodological issues, evaluation metrics and tools

SESSION: Human factors

SESSION: Emerging recommendation domains

2023

SESSION: Side Information, Items structure and Relations

SESSION: Sequential Recommendation

SESSION: Click-Through Rate Prediction

SESSION: Trustworthy Recommendation

SESSION: Collaborative filtering

SESSION: Graphs

SESSION: Interactive Recommendation

SESSION: Reinforcement Learning

SESSION: Cross-domain Recommendation

SESSION: Multimedia Recommendation

SESSION: Knowledge and Context

SESSION: Multi-task Recommendation