SoRec: Social Recommendation Using Probabilistic Matrix Factorization

Recommender Systems with Social Regularization

Hao Ma, Haixuan Yang, Dengyong Zhou, Chao Liu, **Michael R. Lyu, Irwin King**

2023.01.31

DSAIL 2022 Winter Lab Intern Seohyun Lim

Table of Contents

01 Introduction

- ## Recommender System
  - Nowadays, many recommender systems are based on Collaborative Filtering, which predicts interests of a user based on similar users or items
- ## Inherent weaknesses of collaborative filtering
  - (1) Due to sparsity of the user-item rating matrix, memory-based collaborative filtering method fail to find similar users
  - (2) Almost all of collaborative filtering methods cannot handle users who have never rated any items
  - (3) In reality, people are influenced by friends and companies

01 Introduction

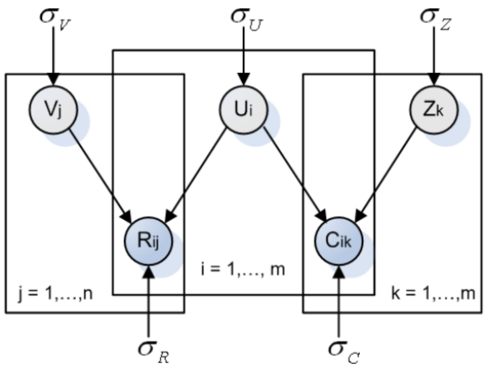
- ## Importance of social interaction

  - Traditional recommender systems assume that users are i.i.d. (independent and identically distributed), so it ignores the social interactions between users
  - One paper showed that people prefer friend’s recommendation than model’s
  - Another paper which researched on social network found out people who chat with each other are more likely to share interests

01 Introduction

▪ Social Recommendation

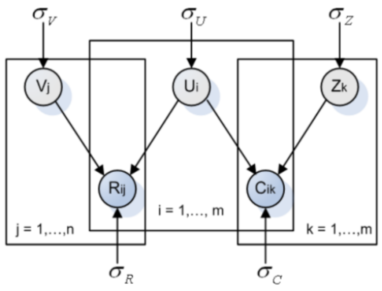
- Proposed a new method that fuse a user's social network graph with user-item matrix which is called "Social Recommendation"
- The paper connect two different data by sharing user latent feature space



01 Introduction

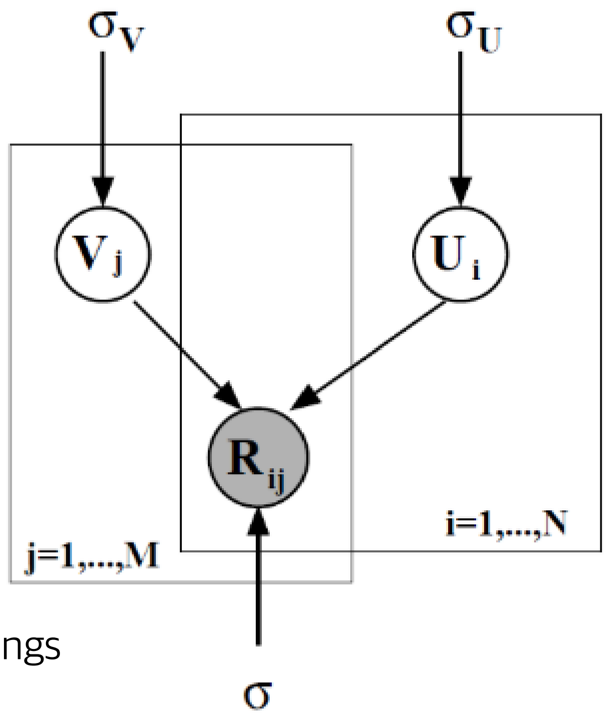
▪ Contribution

- (1) Proposed a successful extended method of PMF using social network information
- (2) Shows good performance on users that have few or no ratings
- (3) Scales linearly with the number of observations and can be applied to large datasets



02 Related Work

- **PMF**
  - Popular method for matrix factorization
  - Models the user-item matrix as a product of two lower-rank user and movie matrices
  - The latent factors and observed ratings are modeled as Gaussian random variables to estimate the uncertainty in the latent factors and ratings



02 Related Work

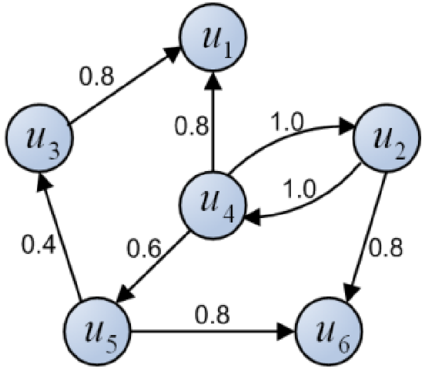
- ## Trust-aware collaborative filtering method

  - Takes into account the trust between users when making recommendations
  - Bedi et al. proposed *Trust based recommender system for semantic web*
  - However, this method is memory-based model which uses similarity function, not latent factor
  - Our model uses probabilistic factor analysis and can deal with missing value

03 Social Recommendation Framework

▪ Problem Definition - Toy Example

- (a) Social network graph: User 1~6 and 8 edges
- (b) User-Item Matrix: matrix used in matrix factorization (MF)



(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

$$U = \begin{bmatrix} 1.55 & 1.22 & 0.37 & 0.81 & 0.62 & -0.01 \\ 0.36 & 0.91 & 1.21 & 0.39 & 1.10 & 0.25 \\ 0.59 & 0.20 & 0.14 & 0.83 & 0.27 & 1.51 \\ 0.39 & 1.33 & -0.43 & 0.70 & -0.90 & 0.68 \\ 1.05 & 0.11 & 0.17 & 1.18 & 1.81 & 0.40 \end{bmatrix},$$

$$V = \begin{bmatrix} 1.00 & -0.05 & -0.24 & 0.26 & 1.28 & 0.54 & -0.31 & 0.52 \\ 0.19 & -0.86 & -0.72 & 0.05 & 0.68 & 0.02 & -0.61 & 0.70 \\ 0.49 & 0.09 & -0.05 & -0.62 & 0.12 & 0.08 & 0.02 & 1.60 \\ -0.40 & 0.70 & 0.27 & -0.27 & 0.99 & 0.44 & 0.39 & 0.74 \\ 1.49 & -1.00 & 0.06 & 0.05 & 0.23 & 0.01 & -0.36 & 0.80 \end{bmatrix},$$

(c) 5 dimensions latent feature space

03 Social Recommendation Framework

▪ Social Recommendation

- *m* users, *n* movies, *l*-dimension
- U : shared user latent feature space (l * m)
- V : item latent feature space (l * n)
- Z : factor matrix in the social matrix graph (l * m)
- R : target rating(preference) matrix
- C : social network matrix

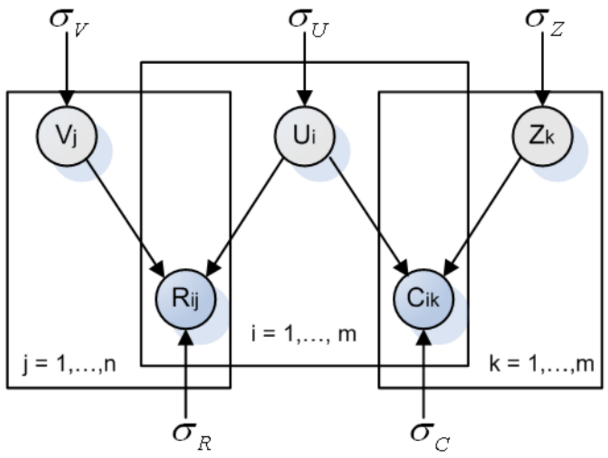


Figure 2: Graphical Model for Social Recommendation

03 Social Recommendation Framework

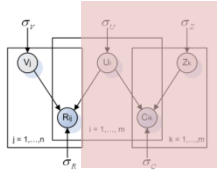


Figure 2: Graphical Model for Social Recommendation

▪ Social Network Matrix Factorization

- Bayes' rule for parameter estimation:

$$\underbrace{P(\theta \mid D)}_{posterior} = \frac{P(D \mid \theta)\, P(\theta)}{P(D)} \propto \underbrace{P(\theta)}_{prior}\, \underbrace{P(D \mid \theta)}_{likelihood}$$

- **posterior distribution** : out belief about how likely each value of parameter Θ is given D
- **prior distribution** : initial belief about how likely each value of parameter Θ might be
- **likelihood** : how likely each observation D is for a fixed parameter Θ

03 Social Recommendation Framework

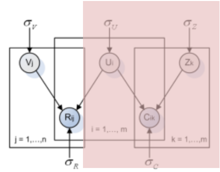


Figure 2: Graphical Model for Social Recommendation

▪ Social Network Matrix Factorization

- Bayes' rule for parameter estimation:

$$\underbrace{P(\theta \mid D)}_{posterior} = \frac{P(D \mid \theta)\, P(\theta)}{P(D)} \propto \underbrace{P(\theta)}_{prior}\, \underbrace{P(D \mid \theta)}_{likelihood}$$

- For our model, U and Z will be parameters, C and σ is given dataset

$$p(U, Z|C, \sigma_C^2, \sigma_U^2, \sigma_Z^2) \propto p(C|U, Z, \sigma_C^2)p(U|\sigma_U^2)p(Z|\sigma_Z^2)$$

03 Social Recommendation Framework

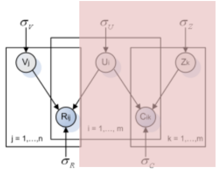


Figure 2: Graphical Model for Social Recommendation

- ## Social Network Matrix Factorization

  - The conditional distribution over the observed social network relationships as:

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

c_{ik} how much a user i trusts or knows user k in a social network

$\mathcal{N}(x|\mu,\sigma^2)$ probability density function of the Gaussian distribution with mean μ and variance σ2

I_{ik}^C indicator function that is equal to 1 if user i trusts user k, otherwise 0

$g(x) = 1/(1+\exp(-x))$ logistic function that make $U_i^T Z_k$ within the range [0,1]

03 Social Recommendation Framework

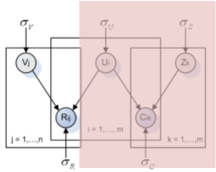


Figure 2: Graphical Model for Social Recommendation

- **Social Network Matrix Factorization**
  - We also place zero-mean spherical Gaussian priors on user and factor feature vectors:

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

$\mathcal{N}(x|\mu,\sigma^2)$ probability density function of the Gaussian distribution with mean μ and variance σ2

03 Social Recommendation Framework

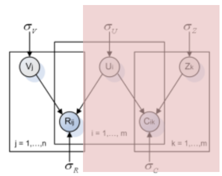


Figure 2: Graphical Model for Social Recommendation

▪ Social Network Matrix Factorization

- Hence, through a simple Bayesian inference, we have:

$$
\begin{aligned}
p(U, Z | C, \sigma_C^2, \sigma_U^2, \sigma_Z^2) & \\
\propto \; & p(C|U, Z, \sigma_C^2) p(U|\sigma_U^2) p(Z|\sigma_Z^2) \\
= \; & \prod_{i=1}^{m} \prod_{k=1}^{m} \mathcal{N}\left[\left(c_{ik} | g(U_i^T Z_k), \sigma_C^2\right)\right]^{I_{ik}^C} \\
\times \; & \prod_{i=1}^{m} \mathcal{N}(U_i | 0, \sigma_U^2 \mathbf{I}) \times \prod_{k=1}^{m} \mathcal{N}(Z_k | 0, \sigma_Z^2 \mathbf{I}). \qquad (3)
\end{aligned}
$$

03 Social Recommendation Framework

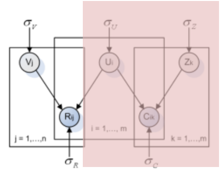


Figure 2: Graphical Model for Social Recommendation

▪ Social Network Matrix Factorization

- We employ term cik* which incorporates local authority and local hub value
- cik value decreases if user i trust lots of users
- cik value increases if user k is trusted by lots of users

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

$$c_{ik}^* = \sqrt{\frac{d^-(v_k)}{d^+(v_i) + d^-(v_k)}} \times c_{ik}, \qquad (4)$$

d+(vi) represents the outdegree of node vi, while d−(vk) indicates the indegree of node vk

03 Social Recommendation Framework

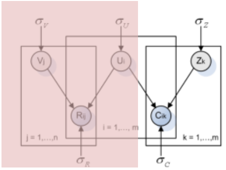


Figure 2: Graphical Model for Social Recommendation

▪ User-Item Matrix Factorization

– For our model, U and V will be parameters, R and σ is given dataset

$$p(U, V|R, \sigma_R^2, \sigma_U^2, \sigma_V^2) \propto p(R|U, V, \sigma_R^2)p(U|\sigma_U^2)p(Z|\sigma_V^2)$$

03 Social Recommendation Framework

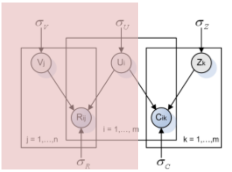


Figure 2: Graphical Model for Social Recommendation

▪ User-Item Matrix Factorization

– Define the conditional distribution over the observed ratings as:

$$p(R|U,V,\sigma_R^2) = \prod_{i=1}^{m} \prod_{j=1}^{n} \mathcal{N}\left[\left(r_{ij}|g(U_i^T V_j), \sigma_R^2\right)\right]^{I_{ij}^R}, \quad (5)$$

r_{ij} rating of user i for movie j

03 Social Recommendation Framework

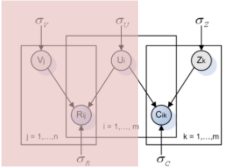


Figure 2: Graphical Model for Social Recommendation

▪ User-Item Matrix Factorization

- We also place zero-mean spherical Gaussian priors on user and movie feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{m} \mathcal{N}(U_i|0, \sigma_U^2\mathbf{I}), \qquad p(V|\sigma_V^2) = \prod_{j=1}^{n} \mathcal{N}(V_j|0, \sigma_V^2\mathbf{I}). \qquad (6)$$

03 Social Recommendation Framework

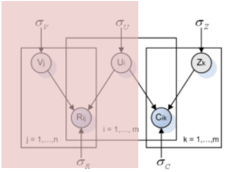


Figure 2: Graphical Model for Social Recommendation

▪ User-Item Matrix Factorization

- Hence, similar to Social Network MF, through a simple Bayesian inference, we have:

$$
\begin{aligned}
p(U, V | R, \sigma_R^2, \sigma_U^2, \sigma_V^2) \\
&\propto p(R|U, V, \sigma_R^2) p(U|\sigma_U^2) p(Z|\sigma_V^2) \\
&= \prod_{i=1}^{m} \prod_{j=1}^{n} \mathcal{N}\left[\left(r_{ij} | g(U_i^T V_j), \sigma_R^2\right)\right]^{I_{ij}^R} \\
&\times \prod_{i=1}^{m} \mathcal{N}(U_i | 0, \sigma_U^2 \mathbf{I}) \times \prod_{j=1}^{n} \mathcal{N}(V_j | 0, \sigma_V^2 \mathbf{I}). \qquad (7)
\end{aligned}
$$

03 Social Recommendation Framework

▪ Matrix Factorization for Social Recommendation

- However, calculating is difficult because of the *exp* function in the Gaussians
- The log of the posterior distribution for social recommendation:

$$\mathcal{N}(X|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right)$$

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

03 Social Recommendation Framework

▪ Matrix Factorization for Social Recommendation

- Maximizing this log-posterior over hyper parameters kept fixed is equivalent to minimizing the following objective functions:

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

where $\lambda_C = \sigma_R^2/\sigma_C^2$, $\lambda_U = \sigma_R^2/\sigma_U^2$, $\lambda_V = \sigma_R^2/\sigma_V^2$, $\lambda_Z = \sigma_R^2/\sigma_Z^2$

03 Social Recommendation Framework

▪ Matrix Factorization for Social Recommendation

- Gradient descent in Ui, Vj and Zk
- For experiment, we set $\lambda_U = \lambda_V = \lambda_Z$.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial U_i} &= \sum_{j=1}^{n} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})V_j \\
&+ \lambda_C \sum_{j=1}^{m} I_{ik}^{C} g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^{*})Z_k + \lambda_U U_i, \\
\frac{\partial \mathcal{L}}{\partial V_j} &= \sum_{i=1}^{m} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})U_i + \lambda_V V_j, \\
\frac{\partial \mathcal{L}}{\partial Z_k} &= \lambda_C \sum_{i=1}^{m} I_{ik}^{C} g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^{*})U_i + \lambda_Z Z_k, (10)
\end{aligned}$$

03 Social Recommendation Framework

▪ Complexity Analysis

- ρ(rho) means number of nonzero entries in matrices
- For $\frac{\partial \mathcal{L}}{\partial U}$, the complexity is $O(\rho_R l + \rho_C l)$
- For $\frac{\partial \mathcal{L}}{\partial V}$, the complexity is $O(\rho_R l)$
- For $\frac{\partial \mathcal{L}}{\partial Z}$, the complexity is $O(\rho_C l)$
- For the computational complexity of evaluating the objective function is $O(\rho_R l + \rho_C l)$
- In general, the complexity of the method is linear with the observations

04 Experimental Analysis

▪ Description of Epinions Dataset

- Knowledge sharing and review site that was established in 1999
- 40,163 users who rated 139,529 items with total 664,824 ratings
- 18,826 users, which is almost half of the population, submitted less that 5 reviews
- In Epinions, the user can trust or block users
- The total number of issued trust statements is 487,183

04 Experimental Analysis

- **Metrics**
  - Mean Absolute Error

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \widehat{r}_{i,j}|}{N}$$

04 Experimental Analysis

- ## Comparison

  - The parameter settings of our approach are $\lambda_C = 10$, $\lambda_U = \lambda_V = \lambda_Z = 0.001$

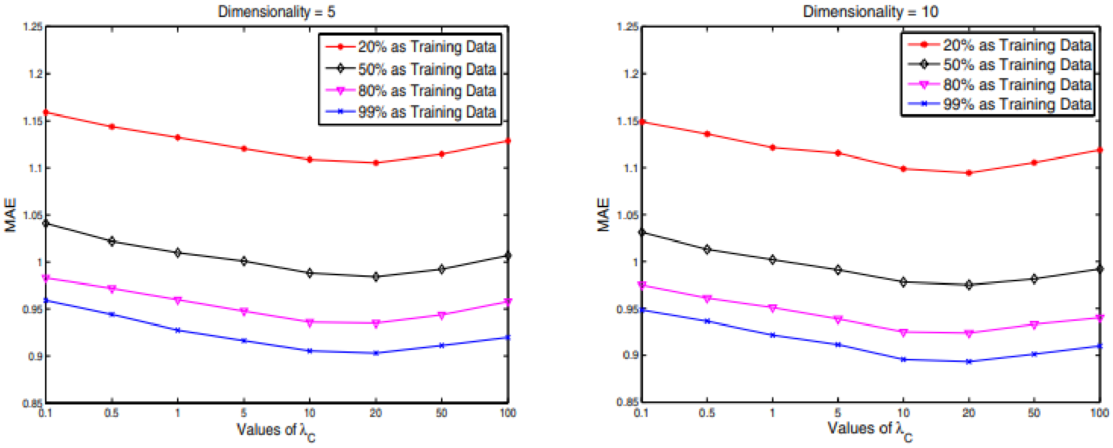
Table 2: MAE comparison with other approaches (A smaller MAE value means a better performance)

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**

04 Experimental Analysis

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

- Impact of Parameter λC
  - The method achieves the best performance when λC ∈ [10, 20]



(a) Dimensionality=5

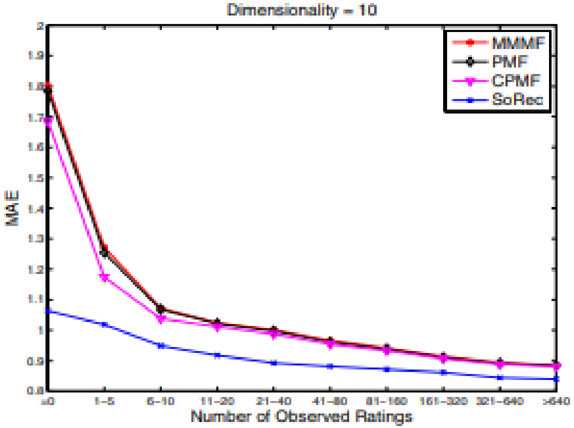
(b) Dimensionality=10

Figure 4: Impact of Parameter λ_C

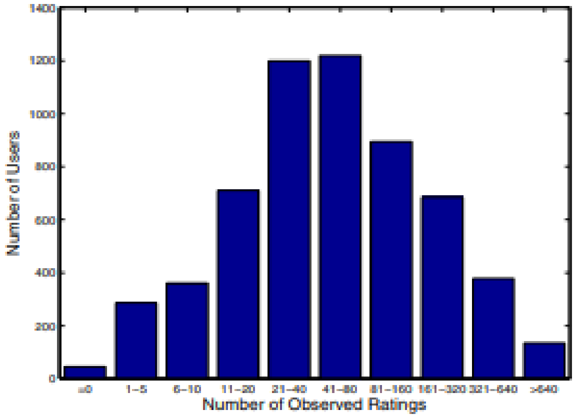
04 Experimental Analysis

▪ Performance of Different Users

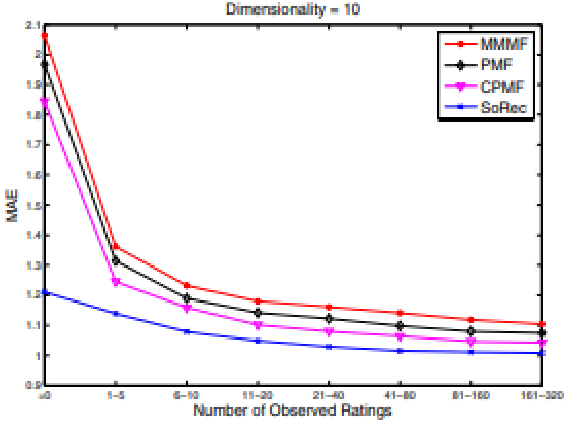
- Group all the users with 10 classes by their number of ratings
- 10 classes: “= 0”,“1 – 5”, “6 – 10”, “11 – 20”, “21 – 40”, “41 – 80”, “81 – 160”, “160 – 320”, “320 – 640”, and “> 640”



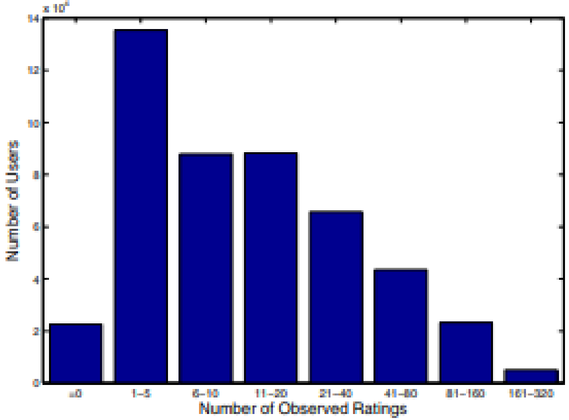
(a) Performance Comparison on Different User Rating Scales (99% as Training Data)



(b) Distribution of Testing Data (99% as Training Data)



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



(h) Distribution of Testing Data (20% as Training Data)

04 Experimental Analysis

▪ Efficiency Analysis

- On a normal PC with Intel Pentium D CPU, 1 Giga bytes memory
- When using 20% data as training data -> less than 5 minutes to train
- When using 99% data as training data -> less than 20 minutes to train
- The computational complexity is linear with the number of data

05 Social Regularization

- **Problem Definition**

SoRec

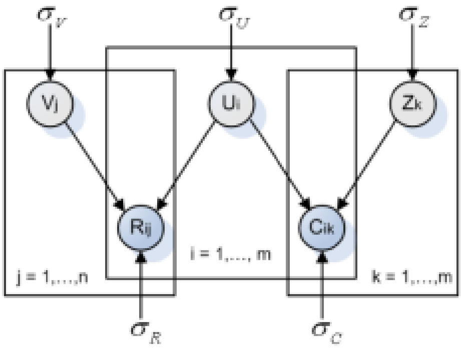


Figure 2: Graphical Model for Social Recommendation

SoReg

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

05 Social Regularization

▪ Problem Definition

- Original Matrix Factorization lost function:

$$\min_{U,V} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij}-U_i^T V_j)^2 + \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2, \quad (4)$$

- Adding social regularization term instead of making new latent factor:

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

Model 1: Average-based Regularization

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

Model 2: Individual-based Regularization

05 Social Regularization

- **Model 1: Average-based Regularization**
  - Suggested from intuition that people is likely to have the average tastes of friends, a new social regularization term:

$$\frac{\alpha}{2}\sum_{i=1}^{m}\|U_i - \frac{1}{|\mathcal{F}^+(i)|}\sum_{f\in\mathcal{F}^+(i)} U_f\|_F^2 \qquad (6)$$

$\mathcal{F}^+(i)$ user i's friend list

$\frac{1}{|\mathcal{F}^+(i)|}\sum_{f\in\mathcal{F}^+(i)} U_f$ average taste of all the friends in friend list

05 Social Regularization

- **Model 1: Average-based Regularization**
  - However, some friends may have totally different interest
  - A more realistic model should treat friends differently based on how similar they are:

$$\frac{\alpha}{2}\sum_{i=1}^{m}||U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)}||_F^2, \qquad (7)$$

$Sim(i,f)$ the similarity function, bigger the similar

05 Social Regularization

▪ Model 2: Individual-based Regularization

- Model 2 tackle each friend individually:

$$\frac{\beta}{2}\sum_{i=1}^{m}\sum_{f\in\mathcal{F}^{+}(i)} Sim(i,f)\|U_i - U_f\|_F^2, \qquad (10)$$

$\mathcal{F}^{+}(i)$ user i’s friend list

$Sim(i,f)$ the similarity function, bigger the similar

05 Social Regularization

▪ Similarity Function

- Vector Space Similarity (VSS):

$$Sim(i,f) = \frac{\sum_{j \in I(i) \cap I(f)} R_{ij} \cdot R_{fj}}{\sqrt{\sum_{j \in I(i) \cap I(f)} R_{ij}^2} \cdot \sqrt{\sum_{j \in I(i) \cap I(f)} R_{fj}^2}}, \qquad (13)$$

$j \in I(i) \cap I(f)$ item j from subset of items which user I and f both rated

R_{ij} rating user i gave item j

05 Social Regularization

▪ Similarity Function

- Pearson Correlation Coefficient (PCC):

$$Sim(i,f) = \frac{\sum_{j \in I(i) \cap I(f)} (R_{ij} - \overline{R}_i) \cdot (R_{fj} - \overline{R}_f)}{\sqrt{\sum_{j \in I(i) \cap I(f)} (R_{ij} - \overline{R}_i)^2} \cdot \sqrt{\sum_{j \in I(i) \cap I(f)} (R_{fj} - \overline{R}_f)^2}} \quad (14)$$

$j \in I(i) \cap I(f)$ item j from subset of items which user I and f both rated

$\overline{R}_i$ the average rate of user i

06 Experimental Analysis of SoReg

Comparisons

- SR2 with PCC similarity function showed best result

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%				

06 Experimental Analysis of SoReg

▪ Impact of Similarity Functions

- Setting similarities to all the same value or random were worse than using functions

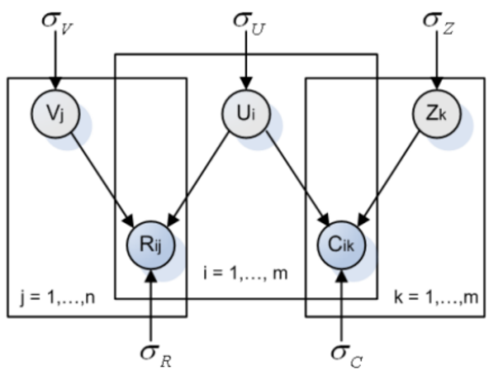
Table 6: Similarity Analysis (Dimensionality = 10)

Dataset	Training	Metrics	SR2 Sim=1	SR2 Sim=Ran	$SR2_{vss}$	$SR2_{pcc}$
Douban	80%	MAE	0.5579	0.5592	0.5548	0.5543
		RMSE	0.7034	0.7047	0.6992	0.6988
	60%	MAE	0.5631	0.5643	0.5597	0.5593
		RMSE	0.7083	0.7098	0.7046	0.7042
	40%	MAE	0.5724	0.5737	0.5690	0.5685
		RMSE	0.7195	0.7209	0.7129	0.7125
Epinions	90%	MAE	0.8324	0.8345	0.8258	0.8256
		RMSE	1.0794	1.0809	1.0744	1.0739
	80%	MAE	0.8511	0.8530	0.8447	0.8443
		RMSE	1.1002	1.1018	1.0958	1.0954

07 Conclusion and Future Work

▪ Conclusion

- SoRec incorporated social network graph into latent factor using PMF
- SoReg introduced a new regularization social term



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

Model 2: Individual-based Regularization

07 Conclusion and Future Work

Future Work

- (1) In SoRec, model did not use block relationship in Epinions, but future work may consider this kind of relationship
- (2) SoReg used traditional similarity functions like PCC and VSS but future work may develop a new method to compute similarity
- (3) Papers are considering user similarity but not as much to item similarity

08 Implementation

```
model = SoRec(n_users, n_items, social_user)
```

- SoRec init

```
class SoRec(torch.nn.Module):
  def __init__(self, n_users, n_items, social_user, n_factors=10, lr=0.01,
               lambda_C=10, lambda_UVZ=0.001, sparse=False, device=torch.device("cuda")):
    super(SoRec, self).__init__()

    self.n_users = n_users
    self.n_items = n_items
    self.social_user = social_user
    self.n_factors = n_factors
    self.lr = lr
    self.lambda_C = lambda_C
    self.lambda_UVZ = lambda_UVZ
    self.alpha = 0.1
    self.sparse = sparse
    self.device = device

    self.user_embeddings = nn.Embedding(self.n_users, self.n_factors, sparse=self.sparse)
    self.item_embeddings = nn.Embedding(self.n_items, self.n_factors, sparse=self.sparse)
    self.social_embeddings = nn.Embedding(self.n_users, self.n_factors, sparse=self.sparse)

    nn.init.normal_(self.user_embeddings.weight, std=0.01)
    nn.init.normal_(self.item_embeddings.weight, std=0.01)
    nn.init.normal_(self.social_embeddings.weight, std=0.01)
    self = self.to(self.device)
    self.loss_list = []
```

08 Implementation

- **Loss Function first part**
  - preds : dot product of U and V
  - val : rij
  - loss : MSE loss

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

```python
# user - item matrix prediction
users_batch = users_batch.long()
items_batch = items_batch.long()
val = val.float().to(device)
preds = model.forward(users_batch, items_batch)
loss = nn.MSELoss(reduction='sum')(preds, val)
```

08 Implementation

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

- **Loss Function second part**
  - social_preds : dot product of U and Z
  - social_val : cij
  - loss : lambda_C * MSE loss

```python
# user - user2 social network
sim_user = []
social_val = []
users_batch_2 = []

for current_user in (users_batch):
  if len(put_user1_set[int(current_user)]) > 0:
    sim_user += random.sample(put_user1_set[int(current_user)], 1)
    d_minus = len(put_user2_set[sim_user[-1]])
    d_plus = len(put_user1_set[int(current_user)])
    new_val = math.sqrt(d_minus / (d_plus + d_minus + 0.0))
    social_val.append(new_val)
    users_batch_2.append(current_user)
```

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

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

```python
sim_user = torch.tensor(sim_user).long().to(device)
users_batch_2 = torch.tensor(users_batch_2).long().to(device)
social_val = torch.tensor(social_val).float().to(device)
social_preds = model.social_forward(users_batch_2, sim_user)
loss += self.lambda_C * nn.MSELoss(reduction='sum')(social_preds, social_val)
```

08 Implementation

- **Loss Function third part**
  - lambda_UVZ : default 0.001
  - get Frobenius Norm (L2 norm) of every latent factor U, V, Z

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

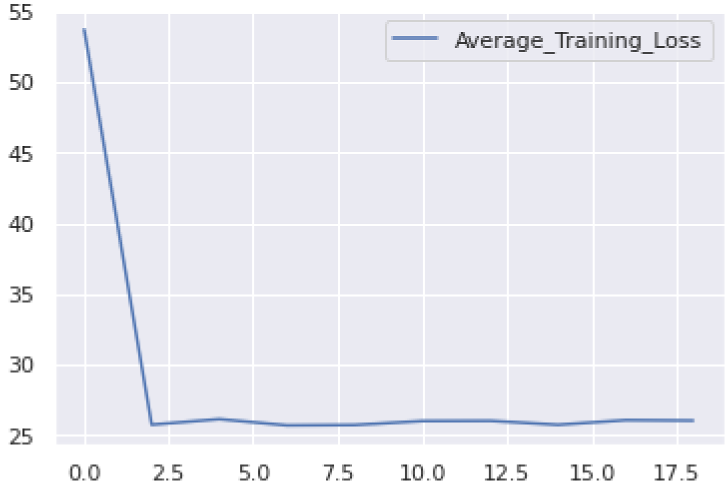
```
lambda_C=10, lambda_UVZ=0.001,
```

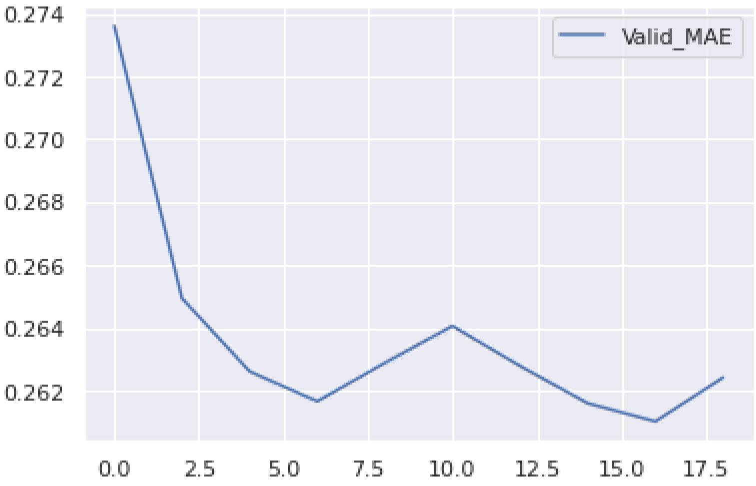
```
# Lastly, every weight norm
loss += self.lambda_UVZ * (self.item_embeddings.weight.norm() +
                           self.user_embeddings.weight.norm() +
                           self.social_embeddings.weight.norm())
```

```
torch.norm(input, p='fro',
```

08 Implementation

▪ SoRec Train loss and MAE





Thank You