SoRec : Social Recommendation using Probabilistic Matrix Factorization
&
SoReg : Recommender Systems with Social Regularization

김지완

Contents

Introduction

Content Based Filtering / Collaborative Filtering

- Content Based Filtering

  Using Items' Features to Recommend Items Similar to What User Likes

  No Need for Other Users' Information / Good At Cold Start User

  Need for Domain Knowledge for Feature Engineering

- Collaborative Filtering

  Using Similarity Between Users/Items to Recommend Items Similar to What User Likes

  No Need Items Feature Information / Generally Good Performance

  Poor Recommendation on Cold Start User

  Memory Based (Neighborhood Based) / Model Based (Latent Factor Based)

Underlying Assumption : Active User will prefer those items which the similar users prefer

→ Widely Employed in Large, Famous Commercial Systems like Amazon, Netflix

Introduction

Inherent Weakness of Collaborative Filtering

1. Sparsity of User - Item Matrix
    Memory Based Models Fails to find similar users

2. Cold Start Problem
    Memeory → Similarity - Cosine / Pearson Correlation Coeffiecient
    Model → Not Updated Latent Vector

3. In Reality, People turn to friends for Movie, Music, and Book Recommendation
    As People are affected by Company they keep,
    Using Only User-Item Matrix is not unrealistic

Traditional Recommender System Assume IID (Independent and Identically Distributed)
→ Ignore Social Interaction Between Users

Introduction

Trust-Aware Recommendation [SoRec]

Traditional Recommender System's Assumption

→ Active User will prefer those items which the similar users prefer

Additional Assumpotion for Social Network

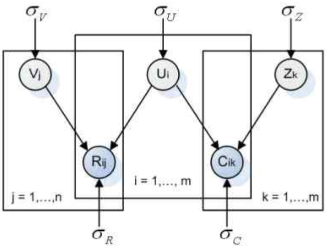
→ Trust Relations can be employed to enhance traditional recommender systems

Fusing User's Social Network Graph with User-Item Rating Matrix for More Accurate and Personalized Recommendation → Social Recommendation

Introduction

SoRec's Probabilistic Factor Analysis

Connect Social Network Structure and the User-Item Rating Matrix Through Shared User Latent Feature Space Based on Probabilistic Factor Analysis



In Shorts, Probabilsitic Matrix Factorization with Graph Matrix and User-Item Matrix

1. Good at Cold Start : Good Performance on users that have few ratings or even none at all

2. Large Data Applicable : Linear Scale with the number of Observations

Introduction

Social Recommendation in Real Sense [SoReg]

Social Recommendation is different from Trust Relationships

1. Trust-Aware Recommender Systems cannot represent the concept of “Social Recommendation” like Social Friend Networks

2. Trust-Aware recommender systems are based on the Assumption that Users have similar tastes with other users they trust
    → the tastes of one user’s friends may vary significantly

3. To provide more proactive and personalized recommendation results to online users

Related Work

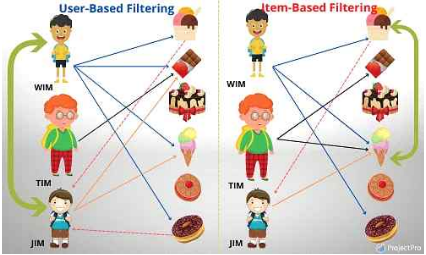
Traditional Recommender Systems

Memory Based (Neighborhood Based)

Widely adopted in Commercial Domain

User Based : Recommend Items based on Other Similar Users

Item Based : Recommend Items based on Other Similar Items



→ Similarity between Users/Items is Important !

Related Work

Traditional Recommender Systems – Similarity Function

PCC Algorithm

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

VSS Algorithm

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

Generally PCC achieve higher performance since it considers the differences of user rating style

Related Work

Traditional Recommender Systems

Model Based (Latent Factor Based)

There are many model-based approach : Clustering Model, Aspect Model, Latent Factor Model

$$R \approx U^T V,$$

Finding Low Dimensional Factors U, V

Proposed Many Variants with Additional Condition on U, V

- Low-Rank Matrix Factorization
- Singular Value Decomposition
- Constraining Norm of U, V
- Probabilistic Semantic Analysis

However, All of above are IID Condition without Considering Social Network

Related Work

Trust-aware Recommender Systems [SoRec]

Collaborative Filitering Process is informed by the reputation of users which is computed by propagating trust

→ Increase Coverage while not reducing Accuracy

Previous Models Before SoRec are not Trust-aware in the real sense

→ Only use Social Information heuristically on Generating Recommendation

In SoRec,
Trust Network and User-item Network simultaneously and seamlessly

Related Work

Social Recommender Systems [SoReg]

“Social Recommender Systems” is Using Social Freinds Network to Imporve Recommender Systems

Many Related Works including SoRec Study Social Recommendation Problem

→ They all have some Disadvantages
→ Not a “Social Recommender System” in Real Sense

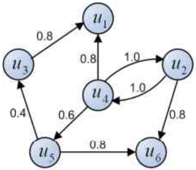
- Only exploring Similar Users to generate Recommendations
- Utilizes Only Trust Information in the experimental analysis
- Using Social Information heuristically on Generating Recommendation

Social Recommendation Framework

SoRec – Toy Example

R : User-Item Rating Matrix
C : Social Network Matrix (Weighted Adjacency Matrix)

→ Matrix Factorization Simultaneously with R ~ UV / C ~ UZ with Shared U



(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

Social Recommendation Framework

SoRec – Matrix Factorzation

m : Users

n : Items

l : Latent Dimension

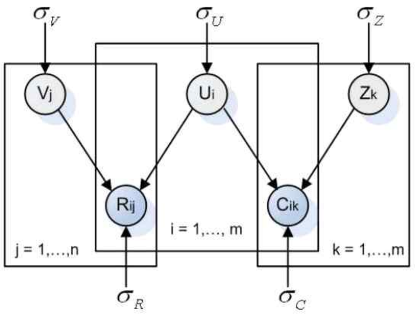
U : Shared User Latent Feature (l by m)

V : Item Latent Feature (l by n)

Z : Factor Latent Feature for Social (l by m)

R : User – Item Rating Matrix (m by n)

C : Socail Network Matrix (m by m)



Social Recommendation Framework

SoRec – Matrix Factorzation

Bayes' Rule

$$P(\theta|D) \propto P(\theta)\ P(D|\theta)$$

Posterior Prior Likelihood

- Posterior : The Probability of θ, given Observed D
- Prior : Initial Probabilty of θ
- Likelihood : The Probability of D, given θ

Social Recommendation Framework

SoRec – Matrix Factorzation

Bayes' Rule

Social Network Matrix

Posterior

$$p(U, Z|C, \sigma_C^2, \sigma_U^2, \sigma_Z^2) \propto \underbrace{p(C|U, Z, \sigma_C^2)}_{\text{Likelihood}} \underbrace{p(U|\sigma_U^2)p(Z|\sigma_Z^2)}_{\text{Prior}}$$

User-Item Matrix

Posterior

$$p(U, V|R, \sigma_R^2, \sigma_U^2, \sigma_V^2) \propto \underbrace{p(R|U, V, \sigma_R^2)}_{\text{Likelihood}} \underbrace{p(U|\sigma_U^2)p(V|\sigma_V^2)}_{\text{Prior}}$$

Social Recommendation Framework

SoRec – Matrix Factorzation

Bayes’ Rule

Prior

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

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

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

Social Recommendation Framework

SoRec - Matrix Factorzation

Bayes' Rule

Likelihood

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

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

Social Recommendation Framework

SoRec

Social Network Matrix Factorization

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

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

Posterior

$$\begin{aligned} p(U, Z|C, \sigma_C^2, \sigma_U^2, \sigma_Z^2) \\ \propto \quad & p(C|U, Z, \sigma_C^2) p(U|\sigma_U^2) p(Z|\sigma_Z^2) \end{aligned}$$

Likelihood Prior

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

Social Recommendation Framework

SoRec

User-Item Rating Matrix Factorzation

Posterior

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

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

Likelihood Prior

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

$$\times \quad \prod_{i=1}^{m} \mathcal{N}(U_i | 0, \sigma_U^2 \mathbf{I}) \times \prod_{j=1}^{n} \mathcal{N}(V_j | 0, \sigma_V^2 \mathbf{I}).$$

Social Recommendation Framework

SoRec

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

For Convenience of Calculation, Log for Posterior Probability

Maximizing Log Posterior for Social Recommenation

Social Recommendation Framework

SoRec

$$\mathcal{L}(R, C, U, V, Z) =$$

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

$$+\frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2, \qquad (9)$$

Maximizing Log Posterior for Social Recommenation

→ Same With Minimizing Above (9) Equation

Social Recommendation Framework

SoRec

$$\frac{\partial \mathcal{L}}{\partial U_i} = \sum_{j=1}^{n} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})V_j$$

$$+ \lambda_C \sum_{k=1}^{m} I_{ik}^{C} g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^{*})Z_k + \lambda_U U_i,$$

$$\frac{\partial \mathcal{L}}{\partial V_j} = \sum_{i=1}^{m} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})U_i + \lambda_V V_j,$$

$$\frac{\partial \mathcal{L}}{\partial Z_k} = \lambda_C \sum_{i=1}^{m} I_{ik}^{C} g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^{*})U_i + \lambda_Z Z_k, (10)$$

Social Recommendation Framework

SoReg - Adding Social Regularization Term

Original Models

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

Social Regularization Model

$$\min_{U,V} \mathcal{L}_1(R,U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - U_i^T V_j)^2$$

Regularization Term

$$+ \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2, \quad (5)$$

Social Recommendation Framework

SoReg - Adding Social Regularization Term

Model 1 : Average-based Regularization

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

Model 2 : Individual-based Regularization

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

Social Recommendation Framework

SoReg

Model 1 : Average-based Regularization

From Intution that we will consult lots of our friends for valuable suggestions

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

User’s Taste (Latent Vector) should be close to the average tastes of all friends

Social Recommendation Framework

SoReg

Model 1 : Average-based Regularization

Among all of these friends, some friends may have similar tastes with this user, while some other friends may have totally different tastes
Not Just Arithmethic Mean → Weighted Mean

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

Social Recommendation Framework

SoReg

Model 2 : Individual-based Regularization

Model 1's approach is insensitive to those users whose friends have diverse tastes

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

it indirectly models the propagation of tastes

Social Recommendation Framework

SoReg - Adding Social Regularization Term

Model 1 : Average-based Regularization

$$\frac{\partial \mathcal{L}_1}{\partial U_i} = \sum_{j=1}^{n} I_{ij}(U_i^T V_j - R_{ij})V_j + \lambda_1 U_i$$
$$+ \alpha(U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)})$$
$$+ \alpha \sum_{g\in\mathcal{F}^-(i)} \frac{-Sim(i,g)(U_g - \frac{\sum_{f\in\mathcal{F}^+(g)} Sim(g,f)\times U_f}{\sum_{f\in\mathcal{F}^+(g)} Sim(g,f)})}{\sum_{f\in\mathcal{F}^+(g)} Sim(g,f)},$$
$$\frac{\partial \mathcal{L}_1}{\partial V_j} = \sum_{i=1}^{m} I_{ij}(U_i^T V_j - R_{ij})U_i + \lambda_2 V_j. \qquad (9)$$

Model 2 : Individual-based Regularization

$$\frac{\partial \mathcal{L}_2}{\partial U_i} = \sum_{j=1}^{n} I_{ij}(U_i^T V_j - R_{ij})V_j + \lambda_1 U_i$$
$$+ \beta \sum_{f\in\mathcal{F}^+(i)} Sim(i,f)(U_i - U_f)$$
$$+ \beta \sum_{g\in\mathcal{F}^-(i)} Sim(i,g)(U_i - U_g),$$
$$\frac{\partial \mathcal{L}_2}{\partial V_j} = \sum_{i=1}^{m} I_{ij}(U_i^T V_j - R_{ij})U_i + \lambda_2 V_j. \qquad (12)$$

Social Recommendation Framework

SoReg Vs SoRec

SoRec

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

SoReg

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

Complexity Analysis

SoReg Vs SoRec

SoRec

$$Objective\ Function : O(p_R l + p_c l)$$

$$\frac{\partial L}{\partial U} : O(p_R l + p_c l)$$

$$\frac{\partial L}{\partial V} : O(p_R l)$$

$$\frac{\partial L}{\partial Z} : O(p_c l)$$

$$\mathbf{Total} : \mathbf{O(p_R l + p_c l)}$$

SoReg

$$Objective\ Function : O(p_R l + p_c l)$$

$$\frac{\partial L}{\partial U} : O(p_R l + p_c l)$$

$$\frac{\partial L}{\partial V} : O(p_R l)$$

$$\mathbf{Total} : \mathbf{O(p_R l + p_c l)}$$

p : Number of Observation

Experimental Analysis

Datasets - Epinions

	SoRec	SoReg	Implementation
	User Item Matrix		
User	40,163	51,670	49,289
Item	139,529	83,509	139,738
Ratings	664,824	631,064	664,824
Sparsity	0.011%	0.014%	0.010%
	Social Network		
User	40,163	51,670	49,289
Edges	487,183	511,799	487,183

Epinions have Trust / Distrust -> Only Use Trust

Experimental Analysis

Datasets – Epinions

MAE (Mean Absolute Error)

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

RMSE (Root Mean Square Error)

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

Experimental Analysis

Datasets – Epinions

Training Data 80%

	MMMF	**PMF**	**SoRec**	**SoReg**
MSE	1.0275	1.0182	0.9240	0.8443

SoReg Results

Dataset	Training	Metrics	UserMean	ItemMean	NMF	PMF	RSTE	$SR1_{vss}$	$SR1_{pcc}$	$SR2_{vss}$	$SR2_{pcc}$
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%				

SoRec Results

Training Data	Dimensionality = 10			
	MMMF	PMF	CPMF	SoRec
99%	0.9916	0.9885	0.9746	**0.8932**
80%	1.0275	1.0182	0.9923	**0.9240**
50%	1.1012	1.0857	1.0632	**0.9751**
20%	1.2413	1.2276	1.1864	**1.0944**

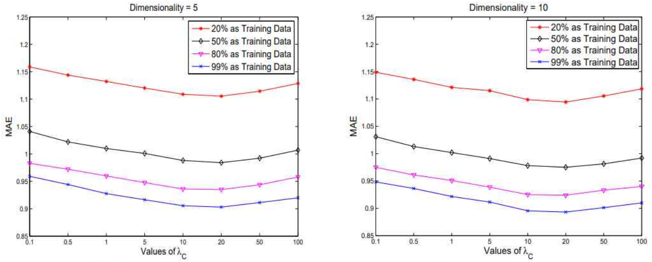
Experimental Analysis

Impact of Parameter λ_c

$\lambda_c = 0$, then Only use Matrix Factorization

$\lambda_c = \text{inf}$, then Only use Social Information

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



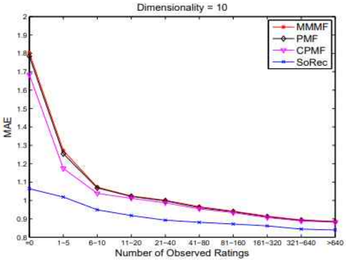
(a) Dimensionality=5 (b) Dimensionality=10

Figure 4: Impact of Parameter λ_C

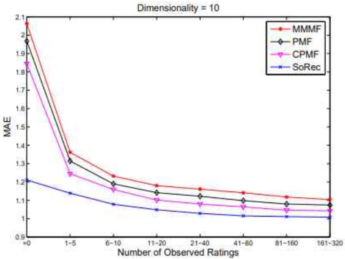
Experimental Analysis

Performance on Different Users

Good Performance when users only supply a few ratings or even have no rating records



99% as Training Data



20% as Training Data

CONCLUSIONS AND FUTURE WORK

1. In SoRec, Distrust Information isn’t investigated
   whether the distrust information is useful to increase the prediction quality
   how to incorporate it

2. In SoRec, Social Network MF ignore Network Diffusion

3. In SoReg, Need More Effective Algorithm to find Suitable Group of Friends

Implementation

1. SoRec and SoReg Both used Mapping function f, g, h

$$f(x) = \frac{x-1}{R_{max}-1}, h(x) = \frac{x+1}{2}$$

$$g(x) = \frac{1}{1+e^{-x}}$$

2. SoReg Used Learning Rate = 0.05

3. SoReg used Thresholds to low similiarity

Implementation

Data Loading - User Item Rating Matrix

```python
with open("data/epinion/ratings_data.txt") as rd :
    n_user = 0
    n_item = 0
    lines = rd.readlines()
    np.random.shuffle(lines)
    train_size = int(len(lines) * 0.5)

    train_data, train_row, train_col = [], [], []
    test_data, test_row, test_col = [], [], []
    train_data_real, test_data_real = [], []

    for i, line in enumerate(lines) :
        user, item, rating = line.split(" ")

        if i < train_size :
            if (int(user) > user_max) or (int(item) > item_max) :
                continue
            train_data.append((int(rating)-1) / 4)
            train_data_real.append(int(rating))
            train_row.append(int(user) - 1)
            train_col.append(int(item) - 1)
        else :
            if (int(user) > user_max) or (int(item) > item_max) :
                continue
            test_data.append((int(rating)-1) / 4)
            test_data_real.append(int(rating))
            test_row.append(int(user) - 1)
            test_col.append(int(item) - 1)

train_R = sps.csr_matrix((train_data, (train_row, train_col)), shape = (user_max, item_max), dtype = 'float64').todok()
test_R = sps.csr_matrix((test_data, (test_row, test_col)), shape = (user_max, item_max), dtype = 'float64').todok()
```

Implementation

Data Loading - Social Network Matrix

```
with open("./data/epinion/trust_data.txt") as td :

    lines = td.readlines()
    data, row, col = [], [], []

    for line in lines :
        user1, user2, _ = line.split(" ")
        if (int(user1) > user_max) or (int(user2) > user_max) :
            continue
        data.append(1)
        row.append(int(user1) - 1)
        col.append(int(user2) - 1)

C = sps.coo_matrix((data, (row, col)), shape = (user_max, user_max), dtype = 'float64')
indegree = C.sum(axis = 0)
outdegree = C.sum(axis = 1)
C_star = copy.deepcopy(C)

for k in range(C_star.data.shape[0]) :
    i = C_star.row[k]
    j = C_star.col[k]
    C_star.data[k] = np.sqrt(indegree[0, j] / (indegree[0, j] + outdegree[i, 0]))

C_star = sps.dok_matrix(C_star)
```

Implementation

SoRec - Loss Function

$$\mathcal{L}(R, C, U, V, Z) =$$

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

$$+\frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2, \qquad (9)$$

```
rating_loss = 0.
for rating, (user_index, item_index) in zip(train_R.values(), train_R.keys()):
    rating_loss += (rating - g(np.dot(U[:, user_index], V[:, item_index])))**2 / 2
```

```
network_loss = 0.
for trust, (user_1, user_2) in zip(C_star.values(), C_star.keys()) :
    network_loss += lambda_C * (trust - g(np.dot(U[:, user_1], Z[:, user_2]))) ** 2 / 2
```

```
norm_loss = 0.
norm_loss += lambda_U * np.linalg.norm(U) / 2
norm_loss += lambda_V * np.linalg.norm(V) / 2
norm_loss += lambda_Z * np.linalg.norm(Z) / 2
```

Implementation

SoRec - Gradients

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

```
for rating, (user_index, item_index) in zip(train_R.values(), train_R.keys()):
    u = U[:, user_index]
    v = V[:, item_index]

    U_grads[:, user_index] += (g2(np.dot(u, v)) * (g(np.dot(u, v)) - rating) * v)
    V_grads[:, item_index] += (g2(np.dot(u, v)) * (g(np.dot(u, v)) - rating) * u)
```

Implementation

SoRec - Gradients

$$\frac{\partial \mathcal{L}}{\partial U_i} = \sum_{j=1}^{n} I_{ij}^R g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})V_j$$

$$+ \boxed{\lambda_C \sum_{k=1}^{m} I_{ik}^C g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^*)Z_k} + \lambda_U U_i,$$

$$\frac{\partial \mathcal{L}}{\partial V_j} = \sum_{i=1}^{m} I_{ij}^R g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})U_i + \lambda_V V_j,$$

$$\frac{\partial \mathcal{L}}{\partial Z_k} = \boxed{\lambda_C \sum_{i=1}^{m} I_{ik}^C g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^*)U_i} + \lambda_Z Z_k, (10)$$

```
for trust, (user_1, user_2) in zip(C_star.values(), C_star.keys()) :
    u1 = U[:, user_1]
    u2 = Z[:, user_2]

    U_grads[:, user_1] += lambda_C * g2(np.dot(u1, u2)) * (g(np.dot(u1, u2)) - trust) * u2
    Z_grads[:, user_2] += lambda_C * g2(np.dot(u1, u2)) * (g(np.dot(u1, u2)) - trust) * u1
```

Implementation

SoRec - Gradients

$$\frac{\partial \mathcal{L}}{\partial U_i} = \sum_{j=1}^{n} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})V_j$$

$$+ \lambda_C \sum_{k=1}^{m} I_{ik}^{C} g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^{*})Z_k + \boxed{\lambda_U U_i}$$

$$\frac{\partial \mathcal{L}}{\partial V_j} = \sum_{i=1}^{m} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})U_i + \lambda_V V_j,$$

$$\frac{\partial \mathcal{L}}{\partial Z_k} = \lambda_C \sum_{i=1}^{m} I_{ik}^{C} g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^{*})U_i + \boxed{\lambda_Z Z_k} \quad (10)$$

```
U_grads += lambda_U * U
V_grads += lambda_V * V
Z_grads += lambda_Z * Z
```

Implementation

SoReg
Model 2 – Loss Function

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

```
rating_loss = 0.
for rating, (user_index, item_index) in zip(train_R.values(), train_R.keys()):
    rating_loss += (rating - g(np.dot(U[:, user_index], V[:, item_index])))**2 / 2

network_loss = 0.

for user_idx in (range(n_user)) :
    u_vec = U[:, user_idx]

    for out_user_idx in outdegree[user_idx].keys() :
        sim_ij = get_similarity(user_idx, out_user_idx)

        network_loss += beta * sim_ij * np.linalg.norm(u_vec - U[:, out_user_idx]) / 2

norm_loss = 0.
norm_loss += lambda_1 * np.linalg.norm(U) / 2
norm_loss += lambda_2 * np.linalg.norm(V) / 2
```

Implementation

SoReg

Model 2 – Gradients

$$
\begin{aligned}
\frac{\partial \mathcal{L}_2}{\partial U_i} &= \boxed{\sum_{j=1}^{n} I_{ij}(U_i^T V_j - R_{ij}) V_j} + \lambda_1 U_i \\
&+ \beta \sum_{f \in \mathcal{F}^+(i)} Sim(i,f)(U_i - U_f) \\
&+ \beta \sum_{g \in \mathcal{F}^-(i)} Sim(i,g)(U_i - U_g), \\
\frac{\partial \mathcal{L}_2}{\partial V_j} &= \boxed{\sum_{i=1}^{m} I_{ij}(U_i^T V_j - R_{ij}) U_i} + \lambda_2 V_j. \qquad (12)
\end{aligned}
$$

```
for rating, (user_index, item_index) in (zip(train_R.values(), train_R.keys())) :
    u = U[:, user_index]
    v = V[:, item_index]

    U_grads[:, user_index] += g2(np.dot(u, v)) * (g(np.dot(u, v)) - rating) * v
    V_grads[:, item_index] += g2(np.dot(u, v)) * (g(np.dot(u, v)) - rating) * u
```

Implementation

SoReg
Model 2 – Gradients

$$\frac{\partial \mathcal{L}_2}{\partial U_i} = \sum_{j=1}^{n} I_{ij}(U_i^T V_j - R_{ij})V_j + \lambda_1 U_i + \boxed{\beta \sum_{f \in \mathcal{F}^+(i)} Sim(i,f)(U_i - U_f)} + \boxed{\beta \sum_{g \in \mathcal{F}^-(i)} Sim(i,g)(U_i - U_g)},$$

$$\frac{\partial \mathcal{L}_2}{\partial V_j} = \sum_{i=1}^{m} I_{ij}(U_i^T V_j - R_{ij})U_i + \lambda_2 V_j. \quad (12)$$

```
for user_idx in (range(n_user)) :
    u = U[:, user_idx]

    for out_user_idx in outdegree[user_idx].keys() :

        U_grads[:, user_idx] += beta * get_similarity(user_idx, out_user_idx) * (u - U[:, out_user_idx])

    for in_user_idx in indegree[user_idx].keys() :

        U_grads[:, user_idx] += beta * get_similarity(user_idx, in_user_idx) * (u - U[:, in_user_idx])
```

Implementation

SoReg

Model 2 – Gradients

$$
\begin{aligned}
\frac{\partial \mathcal{L}_2}{\partial U_i} &= \sum_{j=1}^{n} I_{ij}(U_i^T V_j - R_{ij})V_j + \boxed{\lambda_1 U_i} \\
&+ \beta \sum_{f \in \mathcal{F}^+(i)} Sim(i,f)(U_i - U_f) \\
&+ \beta \sum_{g \in \mathcal{F}^-(i)} Sim(i,g)(U_i - U_g), \\
\frac{\partial \mathcal{L}_2}{\partial V_j} &= \sum_{i=1}^{m} I_{ij}(U_i^T V_j - R_{ij})U_i + \boxed{\lambda_2 V_j.}
\end{aligned}
\qquad (12)
$$

```
U_grads += lambda_1 * U
V_grads += lambda_2 * V
```

Implementation

SoReg

Model 1 – Loss Function

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

```
rating_loss = 0.
for rating, (user_index, item_index) in zip(train_R.values(), train_R.keys()):
    rating_loss += (rating - g(np.dot(U[:, user_index], V[:, item_index])))**2 / 2

network_loss = 0.
for user_idx in range(n_user) :
    u_vec = U[:, user_idx]
    similar_vec = np.zeros_like(U[:, user_idx])
    total_sim = 0.0
    for out_user_idx in outdegree[user_idx] :
        sim_ij = get_similarity(user_idx, out_user_idx)

        similar_vec += sim_ij * U[:, out_user_idx]
        total_sim += sim_ij

    if total_sim != 0.0 :
        similar_vec /= total_sim
    network_loss += alpha * np.linalg.norm(u_vec - similar_vec) / 2

norm_loss = 0.
norm_loss += lambda_1 * np.linalg.norm(U) / 2
norm_loss += lambda_2 * np.linalg.norm(V) / 2
```

Implementation

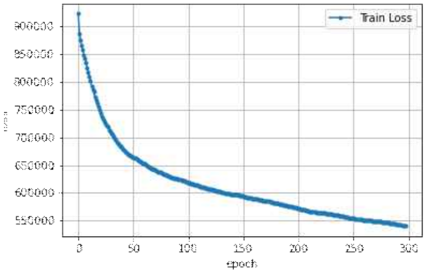
SoReg
Model 1 - Gradients

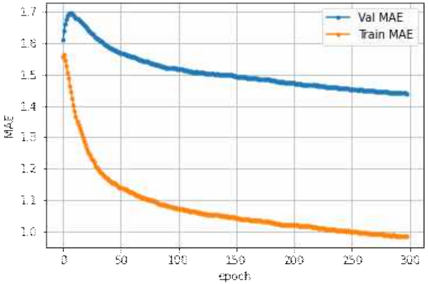
```python
for user_idx in range(n_user) :
    u_vec = U[:, user_idx]
    sim_ig_vec = np.zeros_like(u_vec)

    for in_user_idx in indegree[user_idx].keys() :
        sim_ig = get_similarity(user_idx, in_user_idx)
        if sim_ig == 0 :
            continue
        u_g = U[:, in_user_idx]
        sim_gf_sum = 0.0
        sim_gf_uf = np.zeros_like(u_g)

        for out_user_idx in outdegree[in_user_idx].keys() :
            sim_gf = get_similarity(in_user_idx, out_user_idx)

            sim_gf_sum += sim_gf
            sim_gf_uf += sim_gf * U[:, out_user_idx]

        if sim_gf_sum == 0 :
            continue
        sim_ig_vec += -sim_ig * (u_g - sim_gf_uf / sim_gf_sum)

    U_grads[:, user_idx] += alpha * sim_ig_vec / sim_gf_sum
```

$$\frac{\partial \mathcal{L}_1}{\partial U_i} = \sum_{j=1}^{n} I_{ij}(U_i^T V_j - R_{ij})V_j + \lambda_1 U_i + \alpha\left(U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)}\right) + \boxed{\alpha \sum_{g\in\mathcal{F}^-(i)} \frac{-Sim(i,g)\left(U_g - \frac{\sum_{f\in\mathcal{F}^+(g)} Sim(g,f)\times U_f}{\sum_{f\in\mathcal{F}^+(g)} Sim(g,f)}\right)}{\sum_{f\in\mathcal{F}^+(g)} Sim(g,f)}},$$

$$\frac{\partial \mathcal{L}_1}{\partial V_j} = \sum_{i=1}^{m} I_{ij}(U_i^T V_j - R_{ij})U_i + \lambda_2 V_j. \tag{9}$$

Implementation

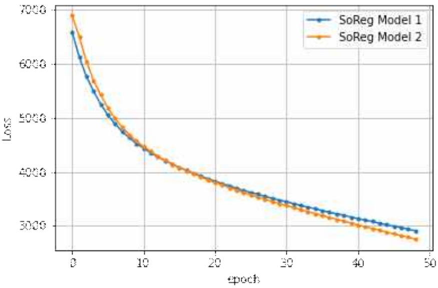


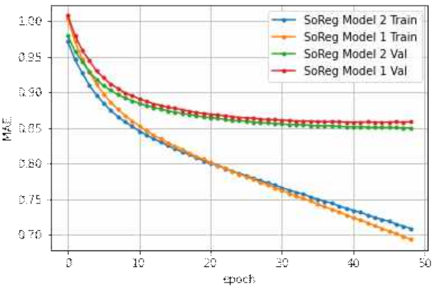


Training Data 50%
Latent Dimension : 10

Learing Speed is Slow / Much Higher than Paper MAE

Implementation

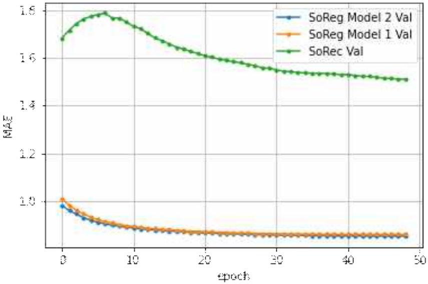




Training Data 50%
Latent Dimension : 10

Due to User Similarity Calculation, Small Size Data Used
User : 10000
Item : 30000

Implementation



1 Epoch Time → SoRec : 11s / SoReg Model 1 : 15s / SoReg Model 2 : 12s

Due to User Similarity Calculation, Small Size Data Used
User : 10000
Item : 30000

Implementation

1. 학습 시간이 오래걸리는 부분들을 찾아내기 어려웠음
   - SoReg 모델에서 User Similarity 구하는 과정
   - SoReg 모델에서 Indegree / Outdegree 구하는 과정

   → 미리 구한 뒤, 사전 형태로 저장

2. Gradient Exploding 하는 경우가 SoRec, SoReg 에서 모두 자주 발생
   - Learning Rate 설정
   - Gradient 계산 과정에서 Overflow 체크