Bayesian Personalized Ranking from Implicit Feedback

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme

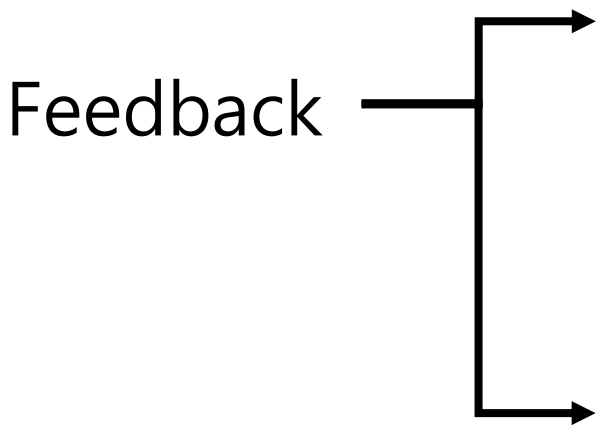
KAIST/BTM & ISysE/20190552

장산하

INDEX

1. Introduction

1. Introduction



Explicit Feedback

- 명시적 피드백
- 별점, 좋아요 or 싫어요 등
- 소수의 데이터

Implicit Feedback

- 암시적 피드백
- 클릭, 사이트 조회 시간, 구매 여부 등
- 다수의 데이터 + 수집 용이

Implicit Feedback = Observed + Non-Observed

= Positive + (Negative + Missing Value)

ex) 구매 내역 + (무관심 + 미래의 구매 예정)

1. Introduction

Implicit Feedback을 이용한 아이템 추천 기존 Model들

: Matrix Factorization, k-Nearest-Neighbor …

But 위의 방법들은 User가 Item을 고를 확률을 계산하는데 초점을 맞춤

So, Ranking을 직접 최적화하기 위해 이번 BRT-Opt를 제시

+ Negative & Missing Value를 구분하면서 Ranking Optimization
+ MF와 adaptive kNN에 적용

2. Formalization

2. Formalization

U: Set of all Users

I: Set of all items

$S \subseteq U \times I$: Implicit Feedback

I_u^+: User u가 Implicit Feedback을 준 Set of Items, $\{i \in I : (u, i) \in S\}$

U_i^+: Item i에 Implicit Feedback을 준 Set of users, $\{u \in U : (u, i) \in S\}$

2. Formalization

$>_u$: User u의 모든 item에 대한 선호 순위, $>_u \subset I^2$

- $i >_u j$ 는 User u가 item i를 j보다 더 선호함을 의미
- I^2인 이유는 Item pair에 대한 비교이기 때문

$>_u$가 만족해야 하는 특성 3가지

$$\forall i, j \in I : i \neq j \Rightarrow i >_u j \vee j >_u i \qquad \text{(totality)}$$

$$\forall i, j \in I : i >_u j \wedge j >_u i \Rightarrow i = j \qquad \text{(antisymmetry)}$$

$$\forall i, j, k \in I : i >_u j \wedge j >_u k \Rightarrow i >_u k \qquad \text{(transitivity)}$$

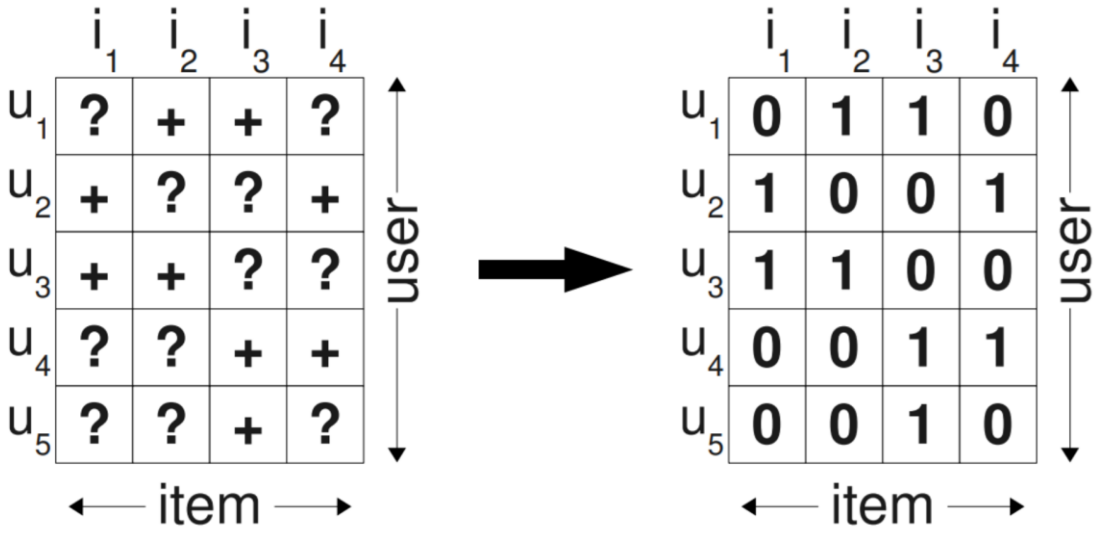
$\wedge: and$
$\vee: or$
$\setminus: minus$

3. Personalized Ranking

3. Personalized Ranking

Implicit Feedback = Observed + Non-Observed

Non-Observed 데이터를 무시하면, Model은 학습할 수 있는게 없다.



- 기존의 Model은 Non-Observed Data를 모두 Negative Feedback(0)으로 분류 및 예측
- 결국 Ranking을 예측할 수 없게 됨
  - Regularization 같은 Overfitting 방지 전략 덕분에 그나마 랭킹 예측 가능
- **즉, 새로운 데이터 Set 및 가정 필요**

3. Personalized Ranking

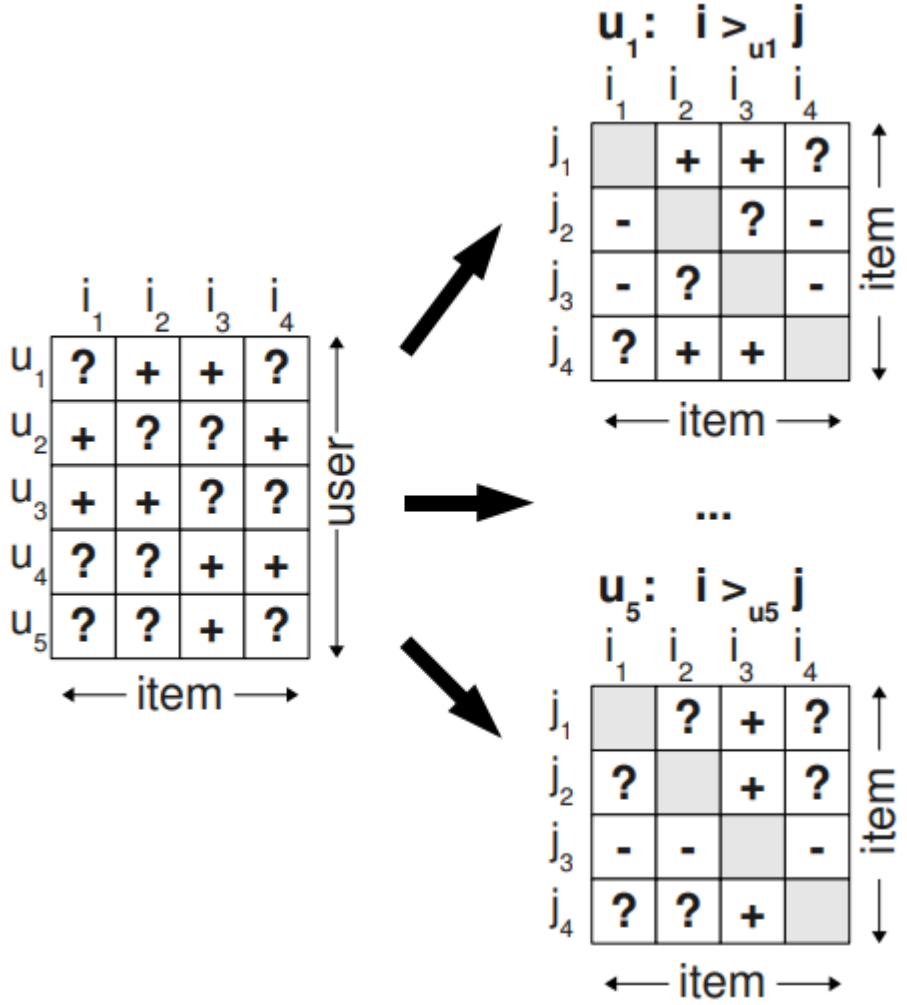
- 단일 item에 점수를 매기는 방식 대신 Item Pair에 랭크를 매겨 최적화

- 3가지 가정

  I. **User는 관측된 item을 관측되지 않은 모든 item보다 더 선호**

  II. 관측된 item들 사이의 선호도 추론할 수 없음

  III. 관측되지 않은 item들 사이의 선호도 또한, 추론할 수 없음



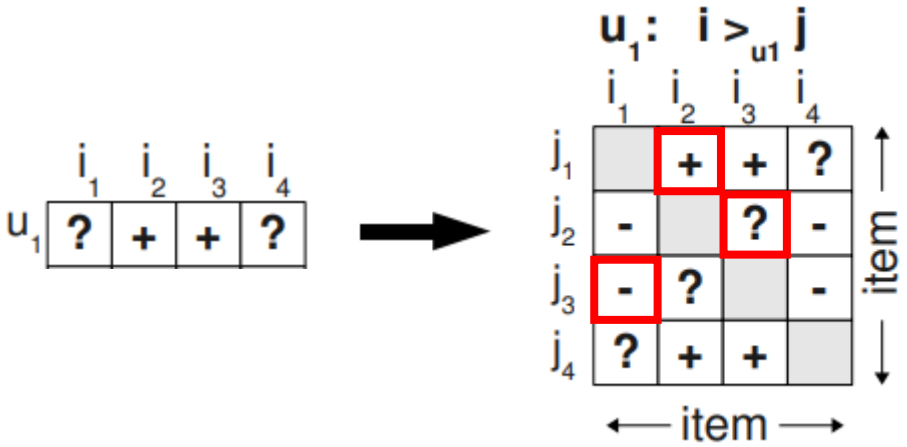
3. Personalized Ranking

New training data set, $D_s: U \times I \times I$

$$D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\}$$

$\wedge: and$
$\vee: or$
$\setminus: minus$

- $(u, i, j) \in D_s$ 는 User u가 i를 j보다 더 선호함을 의미한다.



Item Pair 접근이 좋은 이유

1. Pos, Neg, Missing Value 모두 포함된 Data
2. Training Data가 $>_u$ 의 **부분집합**이기에 오로지 Ranking 예측을 위한 데이터

4. BPR(Bayesian Personalized Ranking)

4. BPR

Bayes' rule:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

$$= \frac{p(y|\theta)p(\theta)}{\int_\theta p(y|\theta)p(\theta)d\theta} \qquad \left(\because p(y) = \int_\theta p(y|\theta)p(\theta)d\theta\right)$$

$$\propto p(y|\theta)p(\theta)$$

Posterior $\propto$ Likelihood X Prior

$p(\theta)$: Prior - subjective belief about θ

$p(y|\theta)$: Likelihood – observation (data) regarding θ

$p(\theta|y)$: Posterior - Updated belief about θ with the data

4. BPR Likelihood

$$p(\Theta|>_u) \propto p(>_u|\Theta)\, p(\Theta)$$ Posterior인 $p(\Theta|>_u)$를 Maximize하는 Θ를 찾아야 함

가정

1. **각 User는 서로 독립적이다.**
2. **각 Item Pair의 Order 또한 서로 독립적이다.**

이때 $>_u$를 $(u,i,j) \in D_s$로 생각한다면

- $i >_u j$, $j >_u i$의 2가지 경우만 존재하기에 베르누이 분포를 따른다고 표현 가능
- 위의 가정에 의해 iid이므로 아래와 같이 표현 가능하다.

$$\prod_{u \in U} p(>_u|\Theta) = \prod_{(u,i,j) \in U \times I \times I} p(i >_u j|\Theta)^{\delta((u,i,j) \in D_S)} \cdot (1 - p(i >_u j|\Theta))^{\delta((u,j,i) \notin D_S)} \qquad \delta(b) := \begin{cases} 1 & \text{if } b \text{ is true,} \\ 0 & \text{else} \end{cases}$$

4. BPR Likelihood

Simplified를 위한 성질

- $>_u$의 totality와 antisymmetry
- 어떤 (i,j)는 다른 모든 Pair와 독립적

$$\prod_{u \in U} p(>_u | \Theta) = \prod_{(u,i,j) \in D_S} p(i >_u j | \Theta)$$

하지만, $>_u$ 의 transitivity가 충족되어야 하기에 $\hat{x}_{uij}(\Theta)$의 sigmoid 값으로 정의

$$p(i >_u j | \Theta) := \sigma(\hat{x}_{uij}(\Theta)) \qquad \sigma(x) := \frac{1}{1 + e^{-x}}$$

$\hat{x}_{uij}(\Theta)$: User u와 item i,j의 관계를 의미하는 model parameter vector Θ 에 대한 real-valued function

4. BPR Prior

$$p(\Theta) \sim N(0, \Sigma_\Theta)$$

이때 Σ_Θ를 $\lambda_\Theta I$로 두면, Unknown hyperparameters를 줄일 수 있다.

λ_Θ: Model Specific Regularization Parameter

$$N(\Theta|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(\Theta - \mu)^T \Sigma^{-1}(\Theta - \mu))$$

$$\propto exp(-\frac{1}{2}\Theta^T(\frac{1}{\lambda_\Theta}I)\Theta)$$

$$= exp(-\frac{1}{2\lambda_\Theta}\Theta^T\Theta) \simeq exp(-\lambda_\Theta\|\Theta\|^2)$$

?

4. BPR Posterior

$$
\begin{aligned}
\text{BPR-OPT} &:= \ln p(\Theta| >_u) \\
&= \ln p(>_u |\Theta)\, p(\Theta) \\
&= \ln \prod_{(u,i,j) \in D_S} \sigma(\hat{x}_{uij})\, p(\Theta) \\
&= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
&= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2
\end{aligned}
$$

4. BPR AUC

$$\text{AUC}(u) := \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{uij} > 0)$$

?

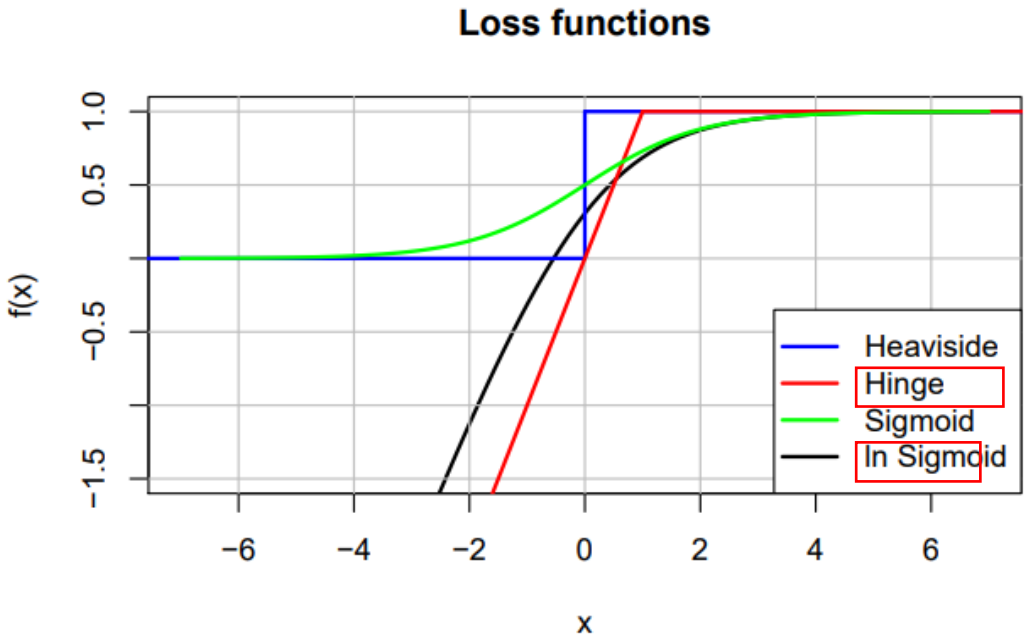
$$\delta(x > 0) = H(x) := \begin{cases} 1, & x > 0 \\ 0, & \text{else} \end{cases}$$

Heaviside

$$\text{AUC} := \frac{1}{|U|} \sum_{u \in U} AUC(u)$$

$$= \sum_{(u,i,j) \in D_S} z_u \, \delta(\hat{x}_{uij} > 0)$$

$$z_u = \frac{1}{|U||I_u^+||I \setminus I_u^+|}$$



- 미분이 불가능한 $H(x)$ 함수를 주로 $\sigma(x)$함수로 대체
- 이 논문에서는 $\ln \sigma(x)$를 사용

정규화 상수인 z_u를 제외하더라도, Loss Function의 살짝 차이만 있음

즉, BPR-Opt와 AUC는 매우 유사

4. BPR Optimizing

BRT-Opt 가 미분 가능하니 Gradient Descent를 쓰는 것이 좋은 방안

$$\frac{\partial \text{BPR-OPT}}{\partial \Theta} = \sum_{(u,i,j) \in D_S} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \frac{\partial}{\partial \Theta} ||\Theta||^2 \qquad \Theta \leftarrow \Theta - \alpha \frac{\partial \text{BPR-OPT}}{\partial \Theta}$$

$$\propto \sum_{(u,i,j) \in D_S} \frac{-e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_\Theta \Theta \qquad \Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \Theta \right)$$

Full Gradient Descent - "Correct" direction but, convergence is slow

- Data가 skewed 하기에 상대적으로 적은 item i의 변화에 민감 → 더 작은 learning rate 설정
- Gradient가 많이 다르기 때문에 Regularization이 어렵다.

Stochastic Gradient Descent

- Observed인 Item i와 Non-Observed인 Item j의 Skewness 어느정도 해소
- 기본적인 SGD를 사용하면, Training pair의 순서가 중요하다 → 주로 User별 or Item별

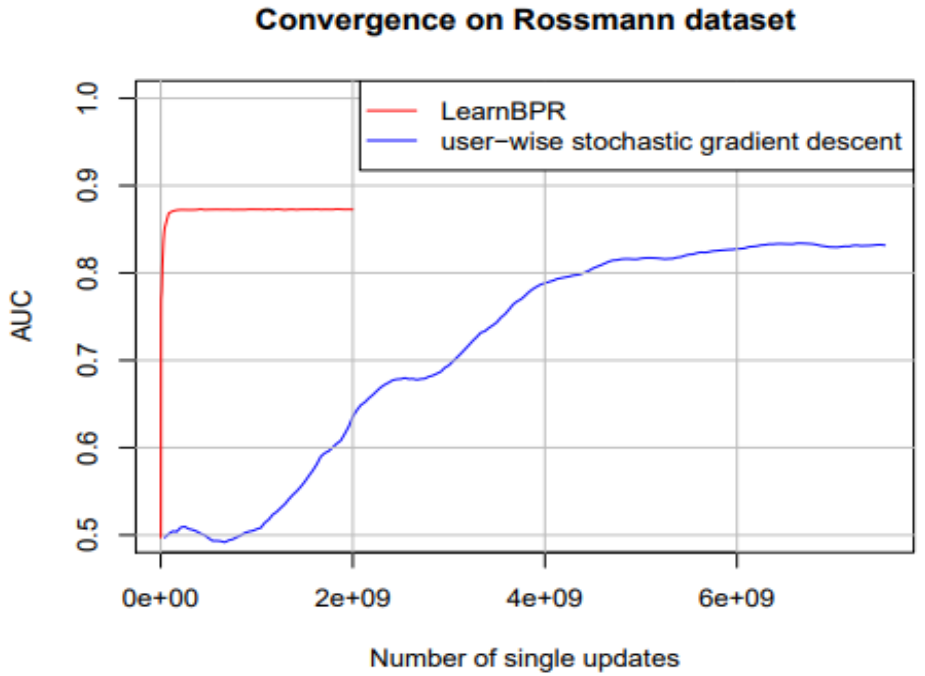
4. BPR Optimizing

하지만, SGD 방식 또한 같은 User-Item Pair에 대해 많은 연속적 업데이트를 가지게 됨

즉, 하나의 (u, i) Pair의 경우에 수 많은 j가 존재한다는 의미

SGD with Bootstrap sampling: (Random으로 Triples (u, i, j) 선택하여 학습)

procedure LEARNBPR(D_S, Θ)
  initialize Θ
  repeat
    draw (u, i, j) from D_S
    $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$
  until convergence
  return $\hat{\Theta}$
end procedure



4. BPR Learning models

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj}$$

이렇게 분해를 하면, $\hat{x}_{ul}$을 예측하는 기존 Collaborate Filtering Model에 적용 가능

하지만 모델은 동일하여도, 다른 optimization criterion인 것을 유의

특히, BRT-Opt는 $\hat{x}_{ul}$인 Score 하나를 예측하는 것이 아니라 $\hat{x}_{ui} - \hat{x}_{uj}$의 차이를 분류

4. BPR Matrix Factorization

$X = U \times I$를 목표로 $\hat{X} = WH^T$를 추정

즉, Parameter Θ는 (W, H)

$$W = U \times k\,, \qquad H^T = k \times I$$

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj} = \sum_{f=1}^{k} w_{uf} \cdot h_{if} - \sum_{f=1}^{k} w_{uf} \cdot h_{jf}$$

w_u는 User u를 설명하는 Feature vector, h_i는 Item i를 설명하는 Feature vector

$$\frac{\partial}{\partial \theta} \hat{x}_{uij} = \begin{cases} (h_{if} - h_{jf}) & \text{if } \theta = w_{uf}, \\ w_{uf} & \text{if } \theta = h_{if}, \\ -w_{uf} & \text{if } \theta = h_{jf}, \\ 0 & \text{else} \end{cases}$$

Regularization Constants

User Feature에 대한 1개

Item Feature에 대한 Regulation 2개

4. BPR Adaptive kNN

Parameter Θ는 C : $I \times I$

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj} = \sum_{l \in I_u^+ \wedge l \neq i} c_{il} - \sum_{l \in I_u^+ \wedge l \neq j} c_{jl}$$

$\wedge: and$
$\vee: or$
$\setminus: minus$

$$c_{i,j}^{\text{cosine}} := \frac{|U_i^+ \cap U_j^+|}{\sqrt{|U_i^+| \cdot |U_j^+|}}$$

c_{il}은 Item i와 Item j의 코사인 유사도를 의미

$$\frac{\partial}{\partial \theta} \hat{x}_{uij} = \begin{cases} +1 & \text{if } \theta \in \{c_{il}, c_{li}\} \wedge l \in I_u^+ \wedge l \neq i, \\ -1 & \text{if } \theta \in \{c_{jl}, c_{lj}\} \wedge l \in I_u^+ \wedge l \neq j, \\ 0 & \text{else} \end{cases}$$

Regularization Constants

c_{il} 업데이트를 위한 1개

c_{jl} 업데이트를 위한 1개

5. Relation to other methods

5. Relation to other methods

Weighted Regularized Matrix Factorization(WR-MF)

$$\sum_{u \in U} \sum_{i \in I} c_{ui}(\langle w_u, h_i \rangle - 1)^2 + \lambda ||W||_f^2 + \lambda ||H||_f^2$$

c_{ul}는 Parameter가 아닌 정해진 값

- WR-MF는 Square-Loss를 최소화하는 SVD 사용
- Overfitting 방지 Regularization
- Positive Feedback의 영향을 증가시키기 위한 가중치 사용

하지만, Item Pair가 아닌 단일 Item에 대한 최적화 진행

5. Relation to other methods

Maximum Margin Matrix Factorization(MMMF)

$$\sum_{(u,i,j)\in D_s} \max(0, 1-\langle w_u, h_i - h_j \rangle)+\lambda_w ||W||_f^2+\lambda_h ||H||_f^2$$

- Explicit Feedback을 위해 설계(For sparse dataset)
- Error Function을 제외하면 BPR-Opt와 매우 유사
- 특정 Matrix Factorization에만 적용 가능

즉, Implicit Feedback 데이터에는 적절하지 않다.

6. Evaluation & Conclusion

6. Evaluation

Datasets

- Rossmann: 10,000 users, 4000 items에 대한 426,612건의 구매 기록
  - User가 다음에 사고 싶어하는 개인화된 list를 예측 목표
- Netflix: 10,000 users, 5000 item에 대한 565,738 Rating 기록
  - User가 다음에 별점을 주고 싶어하는 개인화된 list를 예측 목표
  - Rating Score를 제거함으로 기록만 남게 만듬

이 데이터 셋에서 User마다 한 개씩 User-Item Pair를 선택하여 Train set으로 지정

남은 데이터는 Test set으로 사용하며, 이 과정을 10번 반복

6. Evaluation

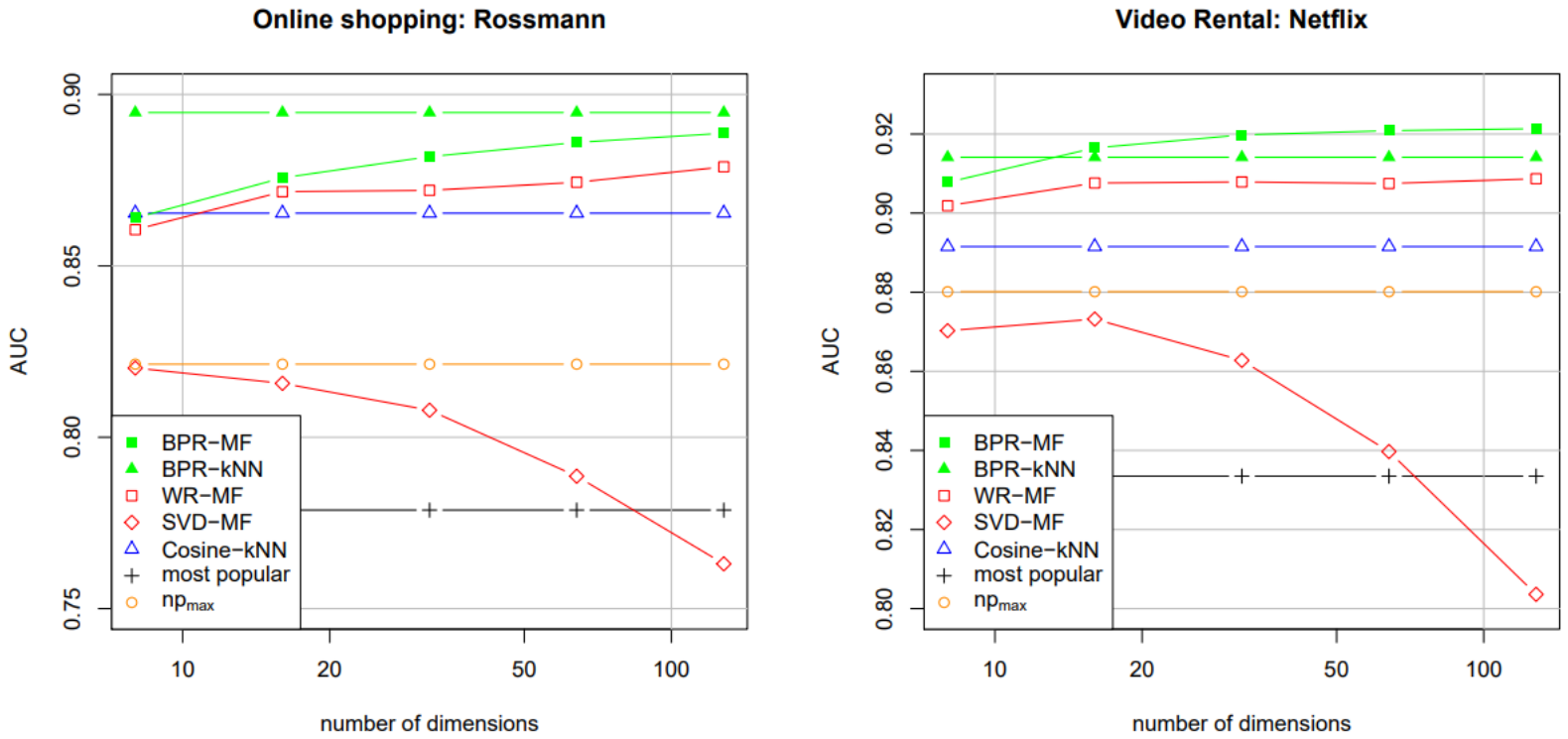
Models:

- MF: SVD-MF, WR-MF, BPR-MF
- kNN: Cosine-kNN, BPR-kNN
- 각 Item마다 User 독립적 가중치를 주는 Most popular 방법
- 개인화되지 않은 Raking 방식에서 AUC의 이론적 상한선

$$\mathrm{AUC} = \frac{1}{|U|} \sum_{u} \frac{1}{|E(u)|} \sum_{(i,j) \in E(u)} \delta(\hat{x}_{ui} > \hat{x}_{uj})$$

$$E(u) := \{(i,j) | (u,i) \in S_{\text{test}} \wedge (u,j) \notin (S_{\text{test}} \cup S_{\text{train}})\}$$

6. Evaluation



Results

- BPR 방법이 다른 모델보다 우수한 성능
- 동일한 모델이지만(MF와 WR-MF) 최적화 방법에 따른 예측 성능 차이를 보임
- 개인화되지 않은 Ranking 방법의 이론적 상한선인 np_{max}를 가장 단순한 Cosine-kNN이 상회하는 성능

6. Conclusion

개인화된 순위에 대한 Generic Optimization Criterion인 BPR-Opt 제시

BPR-Opt에 관한 일반적인 학습 알고리즘은 Learn-BPR 제안

BPR-Opt를 다른 MF, kNN에 적용시킴으로 우수한 성능을 입증

결국, 예측의 질은 모델뿐만 아니라 Optimization Criterion에도 영향을 받는 것을 의미

7. Implementation

7. Implementation

```
df = pd.read_csv("ratings.csv")

df = data_pre(df)

pair_mat = pair_mat(df)

test_list, train_mat = test_train_split(pair_mat)

epochs, aucs = run(train_mat)

aucs[0]= 0

plt.plot(epochs, aucs)
plt.show()
```

userId	movieId	rating	timestamp
1	1	4.0	964982703
1	3	4.0	964981247
1	6	4.0	964982224
1	47	5.0	964983815
1	50	5.0	964982931
...	...	...	...
610	166534	4.0	1493848402
610	168248	5.0	1493850091
610	168250	5.0	1494273047
610	168252	5.0	1493846352
610	170875	3.0	1493846415

7. Implementation

```
def data_pre(df):
    df = df[['userId','movieId','rating']]

    User = pd.DataFrame(df[['userId']].value_counts())
    User.reset_index(inplace = True)
    User = User[User[0] > 9]

    Item = pd.DataFrame(df[['movieId']].value_counts())
    Item.reset_index(inplace = True)
    Item = Item[Item[0] > 9]

    final_user = User['userId'].unique()
    final_item = Item['movieId'].unique()

    df = df[df['movieId'].isin(final_item)]
    df = df[df['userId'].isin(final_user)]
    df = df.reset_index(drop = True)

    return df
```

- Item에 rating을 10번 이상 준 User 선택
- User 10명 이상 rating을 한 Item 선택

7. Implementation

```
def pair_mat(df):
    user_set = sorted(df['userId'].unique())
    item_set = sorted(df['movieId'].unique())

    n_users = len(user_set)
    n_items = len(item_set)

    pair_mat = np.zeros((n_users, n_items))
    pair_mat = pd.DataFrame(pair_mat, index = user_set, columns = item_set)

    for i in range(len(df)):
        pair_mat.loc[df.loc[i,'userId']][df.loc[i,'movieId']] = 1

    return pair_mat
```

- User × Item Matrix 생성
- User별 Rating한 Item을 골라 1로 바꿈

7. Implementation

```python
def test_train_split(pair_mat):
    test_list = []
    train_mat = pair_mat.copy()

    for i in range(train_mat.shape[0]):
        temp = train_mat.iloc[i,:]
        test_idx = np.random.choice(np.where(temp == 1)[0])
        train_mat.iloc[i,test_idx] = 0
        test_list.append((i,test_idx))

    return test_list, train_mat
```

?

- Pair_mat에서 User별 한 개씩 Feedback을 삭제한 Train_mat 생성(1 → 0)
- 삭제한 (User, Item)의 Index를 모음

```python
def bootstrap(train_mat):
    u = np.random.choice(range(train_mat.shape[0]))
    temp = train_mat.iloc[u,:]
    i = np.random.choice(np.where(temp == 1)[0])
    j = np.random.choice(np.where(temp == 0)[0])
    return (u,i,j)
```

- SGD에 사용할 Random Triple $(\boldsymbol{u},\boldsymbol{i},\boldsymbol{j})$ 선택

7. Implementation

```python
def run(train_mat):
    n_dim, epoch, learning_rate, ld = 20, 500000, 0.01, 0.0001

    user_mat = np.random.random((train_mat.shape[0], n_dim))
    item_mat = np.random.random((train_mat.shape[1], n_dim))

    pred, aucs, epochs = [], [], []

    for epoch in range(epoch):
        u, i, j = bootstrap(train_mat)

        W_u, H_i ,H_j  = user_mat[u, : ], item_mat[i, : ], item_mat[j, : ]

        x_uij = np.dot(W_u, H_i) - np.dot(W_u, H_j) # x_ui - x_uj

        grad = np.exp(-x_uij) / (1 + np.exp(-x_uij))

        user_mat[u, : ] = user_mat[u, : ] + learning_rate * (grad * (H_i - H_j) + ld * (W_u))
        item_mat[i, : ] = item_mat[i, : ] + learning_rate * (grad * W_u + ld * (H_i))
        item_mat[j, : ] = item_mat[j, : ] + learning_rate * (-(grad * W_u) + ld * (H_j))

        pred.append(x_uij)

        if epoch % 10000 == 0 :
            auc = np.where(np.array(pred) > 0, 1, 0).mean()
            epochs.append(epoch)
            aucs.append(auc)

    return epochs, aucs
```

- SGD를 통해 최적화
- AUC 변화 확인

7. Implementation