BPR
Bayesian Personalized Ranking from Implicit Feedback

Haejune Lee
DSAIL Winter Internship

Contents

Overview

Overview

BPR-OPT

Implicit feedback 이용해 학습

Personalized Ranking 직접 구하는 방법 제시

Model parameter를 직접 조정

LearnBPR

BPR-OPT 적용한 learning algorithm

Motivation

Implicit feedback

Explicit feedback

많은 연구가 이뤄짐

데이터를 구하기 어려움

Implicit feedback

Automatically tracked

Problem

Positive observation만 가능

Non-observed는 Real negative feedback과 missing value의 혼합

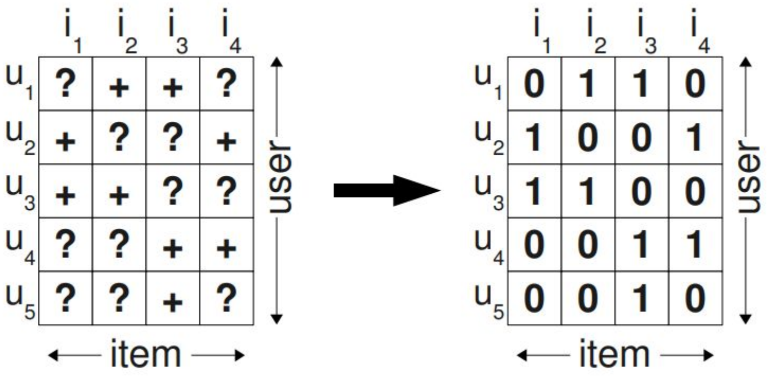
1) Missing value 무시

+, - 구분 불가, ML모델이 학습할 수 없음

2) Negative 취급

Item별 personalized score 예측

모두 Negative로 학습하는 문제 → Regularization



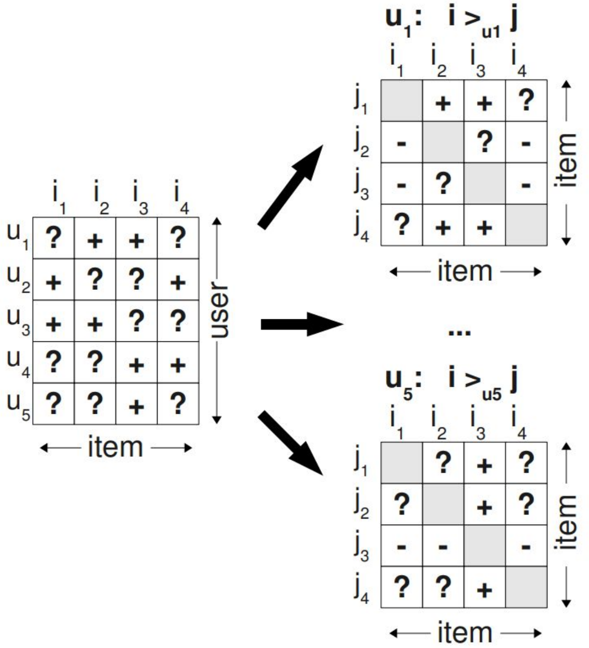
Pairwise preference

Pair level comparison

Observed와 non-observed간의 선호도 비교

각 유저에 대한 item pair의 preference

$(u, i) \in S$ is preferable $>_u$ than $(U \times I) \backslash S$



Pairwise preference

$(u, i) \in S$ is preferable $>_u$ than $(U \times I) \setminus S$

$$\forall i, j \in I : i \neq j \Rightarrow i >_u j \vee j >_u i \qquad \textit{(totality)}$$
$$\forall i, j \in I : i >_u j \wedge j >_u i \Rightarrow i = j \qquad \textit{(antisymmetry)}$$
$$\forall i, j, k \in I : i >_u j \wedge j >_u k \Rightarrow i >_u k \qquad \textit{(transitivity)}$$

$$I_u^+ := \{i \in I : (u, i) \in S\}$$
$$U_i^+ := \{u \in U : (u, i) \in S\}$$

BPR-OPT

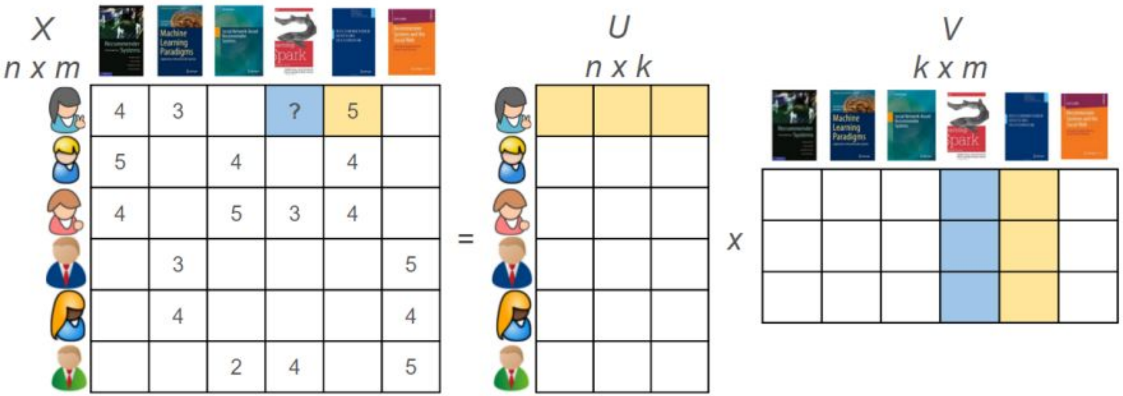
Objective

Increase posterior probability

정확한 ranking을 매기는 것 = $p(\Theta | >_u)$의 최대화

주어진 user의 선호도 학습 데이터를 가장 잘 만족하는 Θ를 찾는 것

e.g. MF에서 feature factor를 찾는 것



Posterior probability

$p(\Theta| >_u) \propto p(>_u |\Theta)p(\Theta)$ Bayes theorem

$p(>_u |\Theta)$ = $\prod_{u \in U} p(>_u |\Theta)$ user independent

= $\prod_{(u,i,j) \in U \times I} p(i >_u j|\Theta)^{\delta((u,i,j) \in D_s)} \cdot (1 - p(i >_u j|\Theta))^{\delta((u,i,j) \notin D_s)}$ item independent

= $\prod_{(u,i,j) \in D_s} p(i >_u j|\Theta)$ totality, antisymmetry

D_s = training data

$$\delta(b) := \begin{cases} 1 & \text{if } b \text{ is true,} \\ 0 & \text{else} \end{cases}$$

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta))$$

σ는 $\sigma(x) := \frac{1}{1+e^{-x}}$ 인 logistic sigmoid 함수

$\hat{x}_{uij}(\Theta)$: user, item사이 관계인 Θ의 실제 값 함수

이후에는 $\hat{x}_{uij}$로 간략화

Prior probability

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

Σ_Θ : variance-covariance matrix, = $\lambda_\Theta I$ 라고 간소화

$$\begin{aligned}
p(\Theta) &= \frac{1}{\sqrt{(2\pi)^d |\Sigma_\Theta|}} \exp\left(-\frac{1}{2}(\Theta-\mu)^T \Sigma_\Theta^{-1} (\Theta-\mu)\right) \\
&= (2\pi\lambda_\Theta)^{-\frac{d}{2}} \exp\left(-\frac{1}{2\lambda_\Theta}\Theta^T\Theta\right) \\
&= (2\pi\lambda_\Theta)^{-\frac{d}{2}} \exp\left(-\frac{1}{2\lambda_\Theta}||\Theta||^2\right)
\end{aligned}$$

BPR-OPT

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

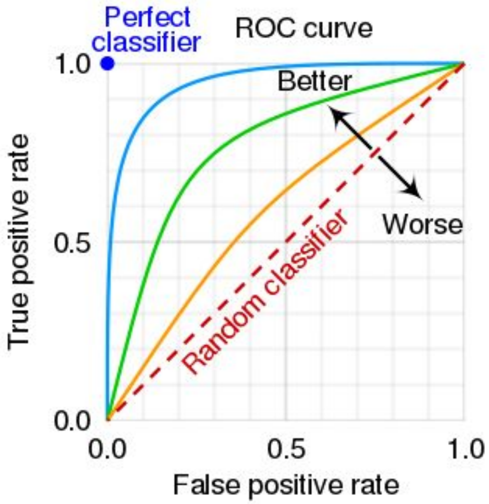
λ_Θ는 model regularization parameter

AUC Optimization

ROC curve

True positive의 비율: Classifier model의 성능 나타냄

AUC(Area Under the Curve): ROC 곡선 아래 영역의 비율



AUC Optimization

AUC per user : True positive / 전체 item pair

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

avg AUC

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

$$= \sum_{(u,i,j) \in D_S} z_u \, \delta(\hat{x}_{uij} > 0) \qquad z_u = \frac{1}{|U|\,|I_u^+|\,|I \setminus I_u^+|}$$

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

BPR-OPT

$$\sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2$$

LearnBPR

Gradient descent

BPR-OPT gradient

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

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

Gradient descent

O(|S||I|)의 모든 training triple(u,i,j) 사용

Skewness problem: 각 i가 모든 j와 비교

$$\Theta \leftarrow \Theta - \alpha \frac{\partial \text{BPR-O}\textsc{pt}}{\partial \Theta}$$

Stochastic Gradient descent

Training 순서에 따라 poor convergence

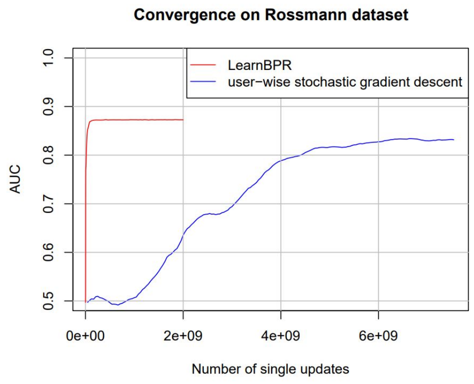
$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \Theta \right)$$

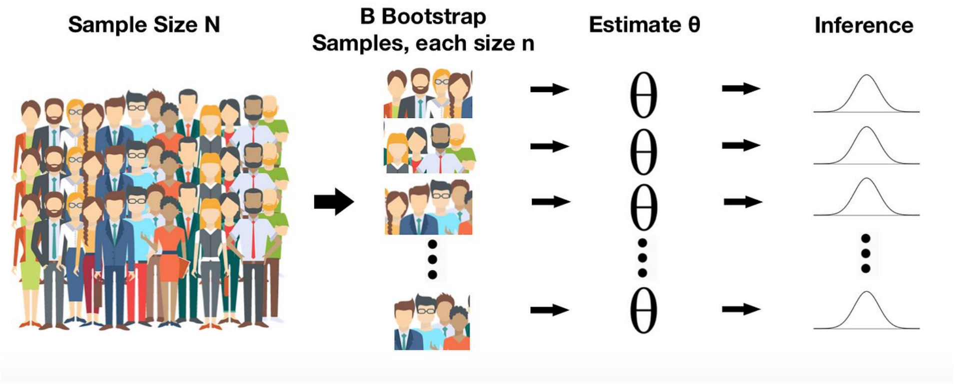
Bootstrap sampling

Randomly chooses triples

Earlystopping 가능함

하나의 triple마다 업데이트 실행





Application

Decomposing estimator

Prediction on single item

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

Matrix factorization

$$\hat{X} := WH^t$$

X는 $W : |U| \times k$와 $H : |I| \times k$로 근사됨

Model parameter

$$\Theta = (W, H)$$

$$\hat{x}_{ui} = \langle w_u, h_i \rangle = \sum_{f=1}^{k} w_{uf} \cdot h_{if}$$

w, h는 각각 user와 item의 feature vector

SVD

overfitting issue

LearnBPR

model parameter에 대한 x hat의 gradient

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

Adaptive kNN

item 기반 kNN

$$\hat{x}_{ui} = \sum_{l \in I_u^+ \wedge l \neq i} c_{il}$$

아이템 간의 유사도

Cosine similarity

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

Using C directly as a model parameter

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

Related works

WR-MF

Weighted Regularized Matrix Factorization

error function내 weight줘서 positive feedback 영향 증대

Implicit feedback으로 item prediction

Optimization criterion: SVD + regularization

Pros

$$O(iter(|S|k^2 + k^3(|I| + |U|)))$$

Cons

One item based prediction

MMMF

Maximum Margin Matrix Factorization

BPR 적용한 MF와 유사한 optimization criterion

Can be applied to implicit feedback

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

Cons

학습 느림: Learning method aims to sparse explicit feedback

Only applicable to MF

Evaluation

Methodology

Leave one out

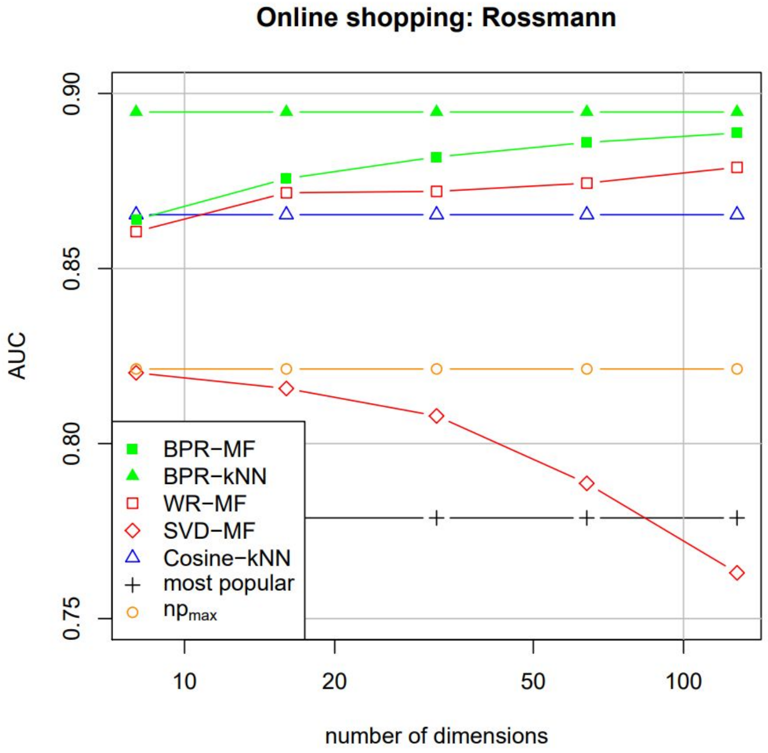
각 user에 대해 S에서 하나의 (u,i) 제거, 예측

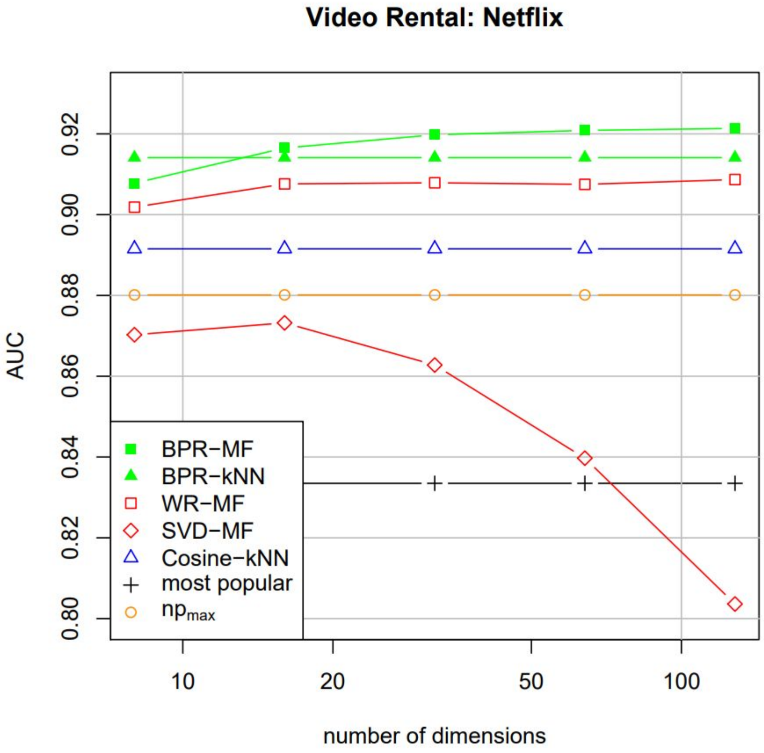
Evaluation criterion: avg AUC

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

E(u)는 각 user의 evaluation pair (i,j)

Result


Online shopping: Rossmann
AUC
0.90
0.85
0.80
0.75
10
20
50
100
number of dimensions
BPR−MF
BPR−kNN
WR−MF
SVD−MF
Cosine−kNN
most popular
npmax


Video Rental: Netflix
AUC
0.92
0.90
0.88
0.86
0.84
0.82
0.80
10
20
50
100
number of dimensions
BPR−MF
BPR−kNN
WR−MF
SVD−MF
Cosine−kNN
most popular
npmax

Conclusion

Importance of optimization criterion

Model choice 뿐 아니라 opt criterion도 중요

Dataset selection

Explicit feedback을 변환하여 사용함 : rating하는 user behavior를 implicit으로 간주

Implementation

Implementation

Transform explicit into implicit

Same as evaluation on paper

Implicit

Use implicit feedback dataset

Explicit to implicit

MovieLens 100K Dataset

MovieLens 100K movie ratings. Stable benchmark dataset. 100,000 ratings from 1000 users on 1700 movies. I
4/1998.

- README.txt
- ml-100k.zip (size: 5 MB, checksum)
- Index of unzipped files

Permalink: https://grouplens.org/datasets/movielens/100k/

Data structure


```
# matrix로 변환
df_mat = df.pivot_table(index = 'user', columns = 'item', values = 'rating', fill_value=0)
print(df_mat)

X_hat = df_mat.to_numpy()
print(X_hat)
```

```
item  874724727  874724754  874724781  874724843  874724905  874724937  ₩
user
1             0          0          0          0          0          0
2             0          0          0          0          0          0
3             0          0          0          0          0          0
4             0          0          0          0          0          0
5             0          0          0          0          0          0
...         ...        ...        ...        ...        ...        ...
939           0          0          0          0          0          0
940           0          0          0          0          0          0
941           0          0          0          0          0          0
942           0          0          0          0          0          0
943           0          0          0          0          0          0
```

Parameters

```
class BPR_MF():

  def __init__(self, data, epochs = 200000, learning_rate = 0.01, feat_dim = 30, lambd = 0.01, patience = 10000):
    self.data = data
    self.epochs = epochs
    self.learning_rate = learning_rate
    self.feat_dim = feat_dim
    self.lambd = lambd
    self.patience = patience
```

Gradient calculation

```
# logistic sigmoid part
x_uij = np.dot(w_u, h_i) - np.dot(w_u, h_j)
exp = np.exp(-x_uij) / (1 + np.exp(-x_uij))

# gradient derivatives
grad_u = exp * (h_i-h_j) + self.lambd * w_u
grad_i = exp * w_u + self.lambd * h_i
grad_j = exp * (-w_u) + self.lambd * h_j

# stochastic gradient descent
W[u,:] = W[u,:] + self.learning_rate * grad_u
H[:,i] = H[:,i] + self.learning_rate * grad_i
H[:,j] = H[:,j] + self.learning_rate * grad_j
```

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

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

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

AUC calculation

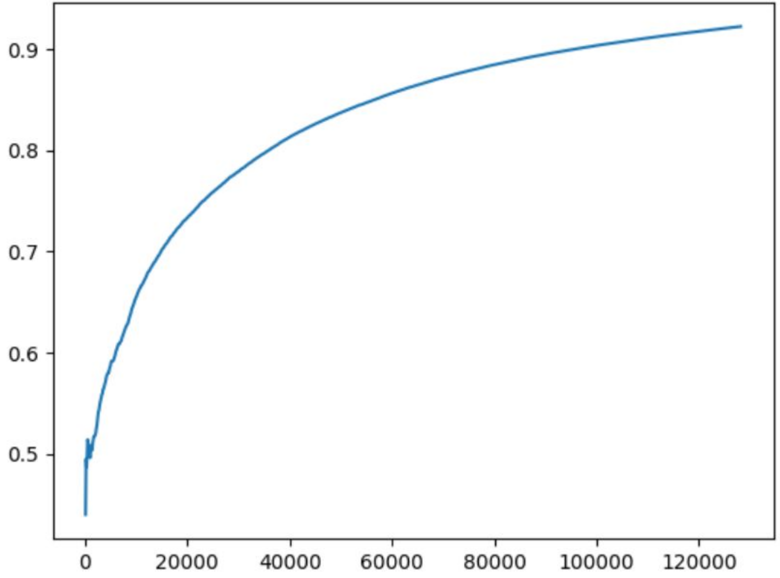

```
### AUC calculation for each 100000 epochs ###
# 평가할때마다 O(|U|||+||/|+|)만큼씩 소요됨..
"""AUC = 0
if iter % 1000 == 0:
  for user in range(self.data.shape[0]):
    z_u_inv = len(np.where(self.data[user] == 1)) * len(np.where(self.data[user] == 1))
    sum_delta = 0
    for i in np.where(self.data[user] == 1)[0]:
      for j in np.where(self.data[user] == 0)[0]:
        auc_w_u = W[user, :]
        auc_h_i = H[:, i]
        auc_h_j = H[:, j]
        auc_x_uij = np.dot(w_u, h_i) - np.dot(w_u, h_j)
        if auc_x_uij > 0:
          sum_delta += 1
    AUC += sum_delta / z_u_inv
  avgAUC = AUC / self.data.shape[0]
  print("epoch:", iter, "/ avgAUC:", avgAUC)"""
```

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

AUC calculation


```
# 해당 iteration의 user에서만 AUC 계산
z_u_inv = len(np.where(self.data[u] == 1)) * len(np.where(self.data[u] == 1))
if x_uij > 0:
  sum_AUC += 1
avg_AUC = sum_AUC / iter
if iter % 100 == 0:
  AUC.append(avg_AUC)
  num_epochs.append(iter)
  print("epoch:",iter,"/ AUC:", avg_AUC)
if iter % 1000 == 0:
  print("patience:", self.patience)
```

AUC calculation




```
patience: 1
epoch: 128100 / AUC: 0.921943793911007
epoch: 128200 / AUC: 0.922004680187207
```

Implicit

RETAILROCKET · UPDATED A YEAR AGO

503 New Notebook

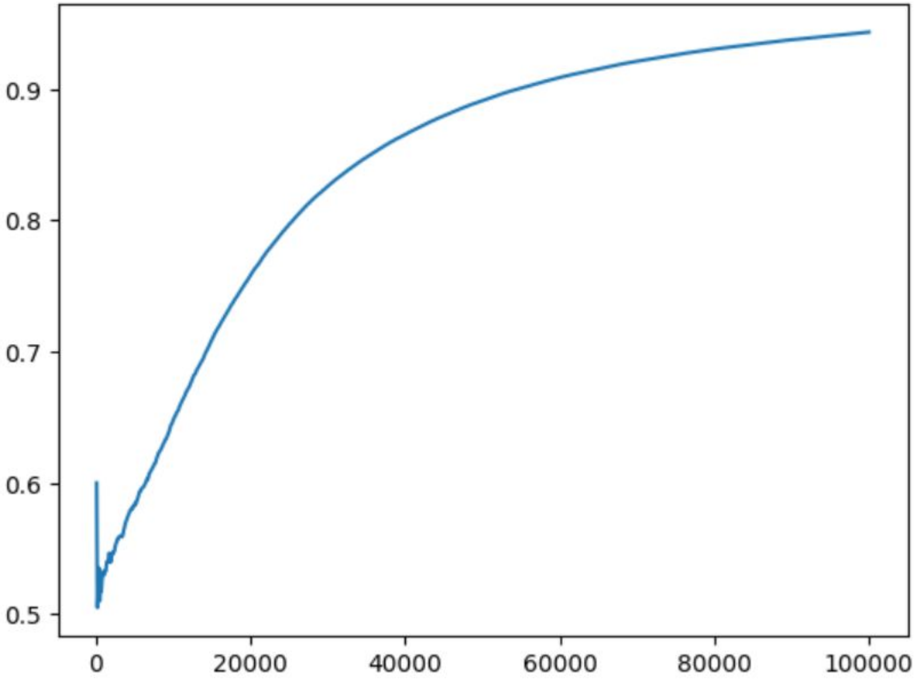
Retailrocket recommender system dataset

Ecommerce data: web events, item properties (with texts), category tree

Content

The behaviour data, i.e. events like clicks, add to carts, transactions, represent interactions that were collected over a period of 4.5 months. A visitor can make three types of events, namely “view”, “addtocart” or “transaction”. In total there are 2 756 101 events including 2 664 312 views, 69 332 add to carts and 22 457

Implicit




```
epoch: 99700 / AUC: 0.9434002006018054
epoch: 99800 / AUC: 0.9434569138276553
epoch: 99900 / AUC: 0.9435035035035035
epoch: 100000 / AUC: 0.94356
```

Result

Explicit feedback을 변환한 것보다 학습이 빠름

성능 또한 더 높음

Limitation & Feedback

AUC calculation

논문에 나온 대로 AUC를 계산하기에는 너무 많은 시간이 소요됨

간략화 AUC를 토대로 성능 개선을 보이나, 논리가 부족함

성능이 너무 높게 나옴

Performance justification

Explicit to implicit과 Implicit feedback dataset을 학습한 결과를 비교했음

Dataset 자체의 영향이 있을 수 있기 때문에 정당성 떨어짐

다른 Method(SVD-MF 등)과 추가적으로 비교해야함

Thank you