Collaborative Filtering for Implicit Feedback Datasets

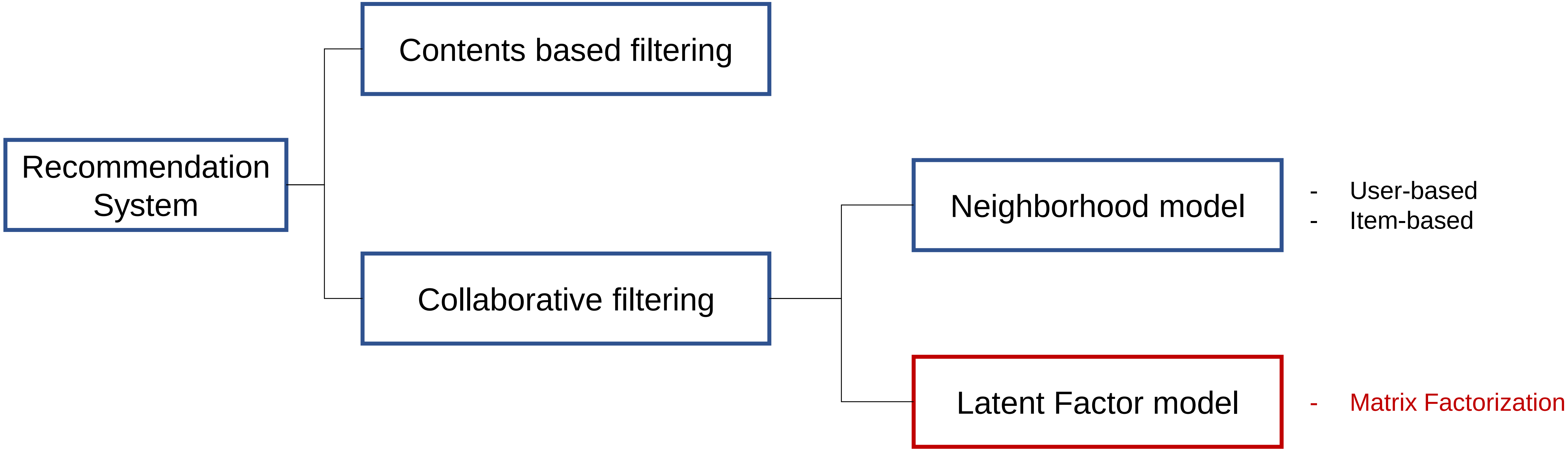
Hu, Yifan, Yehuda Koren, and Chris Volinsky. (2008)

Presented by HYOJUN KIM

CONTENTS

01 Backgrounds

- Previous work

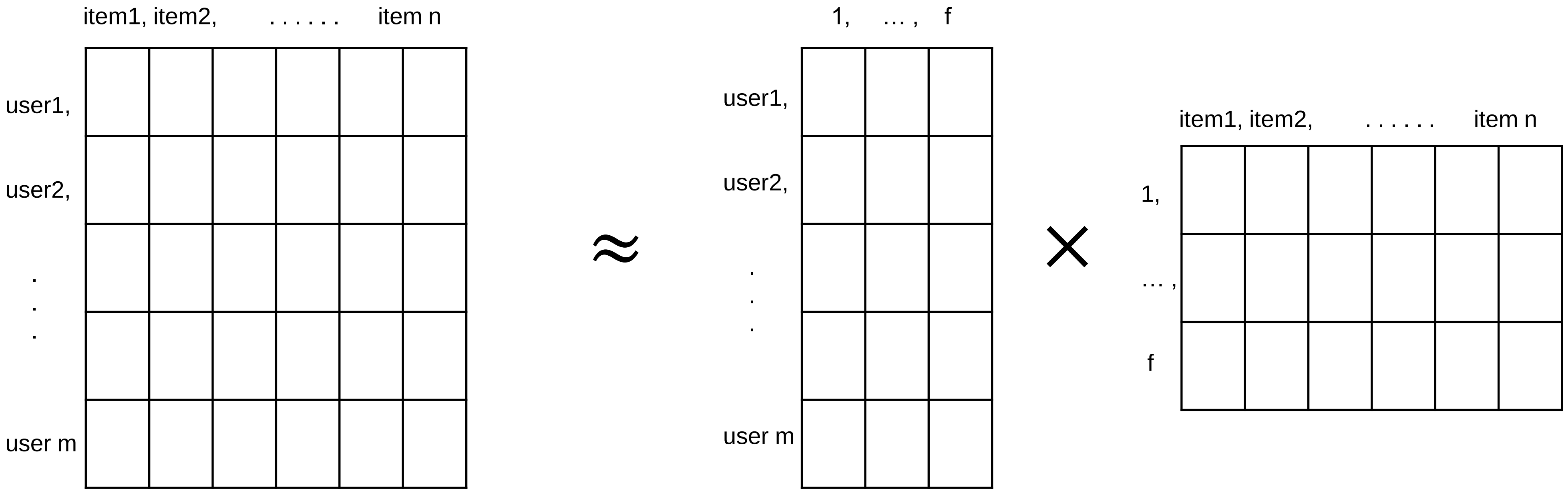


01 Backgrounds

- Previous work

- Matrix Factorization

$$R = U * V^T$$



User-Item Matrix
(m X n)

User Latent Matrix
(m X F)

Item Latent Matrix
(F X n)

01 Backgrounds

- Previous work

- Cost Function

$$J = ||R - UV^T||^2 + \lambda(||U||^2 + ||V||^2)$$

- λ : control the extent of regularization

- Learning Algorithms

1) SGD (Stochastic Gradient Descent)

$$min\ J = \frac{1}{2}\sum_{(i,j)\in S} e_{ij}^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2$$

$$= \frac{1}{2}\sum_{(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}v_{js}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2$$

➔

Ⅰ

$$\frac{\partial J}{\partial u_{iq}} = \sum_{(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}v_{js}\right)\left(-v_{jq}\right) + \lambda u_{iq}$$

$$= \sum_{(i,j)\in S}\left(e_{ij}\right)\left(-v_{jq}\right) + \lambda u_{iq}$$

$$\frac{\partial J}{\partial v_{jq}} = \sum_{(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}v_{js}\right)\left(-u_{iq}\right) + \lambda v_{jq}$$

$$= \sum_{(i,j)\in S}\left(e_{ij}\right)\left(-u_{iq}\right) + \lambda v_{jq}$$

Ⅱ

$$u_{iq} = u_{iq} - \alpha\{\sum_{(i,j)\in S}\left(e_{ij}\right)\left(-v_{jq}\right) + \lambda u_{iq}\}$$

$$v_{jq} = v_{jq} - \alpha\{\sum_{(i,j)\in S}\left(e_{ij}\right)\left(-u_{iq}\right) + \lambda v_{jq}\}$$

- Previous work

- Learning Algorithms

1) SGD (Stochastic Gradient Descent)

① Random Initialize

Ex)

?	3	2
5	1	2
4	2	1

User-Item rating matrix

0.576	1.453
-0.199	-1.218
2.730	0.480

User Latent(U)

×

0.367	-1.108	1.459
-0.339	0.897	0.453

Item Latent(V^T)

=

-0.282	0.666	1.498
0.340	-0.872	-0.842
0.838	-2.593	4.201

$U * V^T$

01 Backgrounds

- Previous work

0.576	1.453
-0.199	-1.218
2.730	0.480

×

0.367	-1.108	1.459
-0.339	0.897	0.453

User Latent(U) Item Latent(V^T)

- Learning Algorithms

1) SGD (Stochastic Gradient Descent)

② Gradient Descent

- $learning\ rate$: 0.05
- λ : 0.01

?	3	2
5	1	2
4	2	1

User-Item rating matrix

-0.282	0.666	1.498
0.340	-0.872	-0.842
0.838	-2.593	4.201

$U * V^T$

Ⅰ

$$e_{12}: 3 - 0.666 = 2.334$$

$$\frac{\partial J}{\partial u_1}: -2.334 * [-1.108, 0.897] + 0.01 * [0.576, 1.453]$$
$$= [2.591, -2.079]$$

$$\frac{\partial J}{\partial v_2}: -2.334 * [-1.108, 0.897] + 0.01 * [0.576, 1.453]$$
$$= [-1.3544, -3.3828]$$

Ⅱ

Update User Latent : [0.576, 1.453] $- learning\ rate * [2.591, -2.079]$

= [0.446, 1.557]

Update Item Latent : [-1.108, 0.897] $- learning\ rate * [-1.3544, -3.3828]$

= [−1.040, 1.066]

01 Backgrounds

- Previous work

0.446	1.557
-0.199	-1.218
2.730	0.480

×

0.367	-1.040	1.459
-0.339	1.0663	0.453

User Latent(U)

Item Latent(V^T)

- Learning Algorithms

1) SGD (Stochastic Gradient Descent)

③ All rating update

?	3	2
5	1	2
4	2	1

User-Item rating matrix

➡

-0.364	1.196	1.356
0.340	-1.092	-0.842
0.839	-2.327	4.201

Ⅰ

Ⅱ

…

➡ All rating update (epoch : 1)

01 Backgrounds

- Previous work

- Learning Algorithms

2) ALS (Alternating Least Squares)

: user-factor 나 item-factor 중 하나를 고정시키고,
다른 하나를 최적화 시키는 방법

$$J = ||R - UV^T||^2 \xrightarrow{\text{fix } U} J = ||R - U\beta||^2$$

$$= ||Y - X\beta||^2$$

least square regression

01 Backgrounds

- Previous work

- Learning Algorithms

2) ALS (Alternating Least Squares)

- Basic Cost function

$$J = ||Y - X\beta||^2$$

$$\sum e^2 = e^T \cdot e = (Y - X\beta)^T(Y - X\beta)$$

$$= (Y^T - \beta^T X^T)(Y - X\beta)$$

$$= (Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta)$$

$$= (Y^T Y - 2\beta^T X^T Y + \beta^T X^T X\beta)$$

$$\frac{\sigma \sum e^2}{\sigma\beta} = -2X^T Y + 2X^T X\beta = 0$$

$$\beta^* = (X^T X)^{-1} X^T \mathrm{Y}$$

- Previous work

- Learning Algorithms

2) ALS (Alternating Least Squares)

- Basic Cost function + Regularization

$$J = ||Y - X\beta||^2 + \lambda||\beta||^2$$

$$\sum e^2 = e^T \cdot e = (Y - X\beta)^T (Y - X\beta)$$

$$= (Y^T - \beta^T X^T)(Y - X\beta)$$

$$= (Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta)$$

$$= (Y^T Y - 2\,\beta^T X^T Y + \beta^T X^T X\beta) + \lambda||\beta||^2$$

$$\frac{\sigma \sum e^2}{\sigma\beta} = -2X^T Y + 2X^T X\beta + 2\lambda\beta = 0$$

$$(X^T X + \lambda I)\beta = X^T Y$$

$$\beta^* = (X^T X + \lambda I)^{-1} X^T Y$$

01 Backgrounds

- Previous work

- Learning Algorithms

2) ALS (Alternating Least Squares)

$$J(u_i) = ||R_i - u_i V^T||^2 + \lambda ||u_i||^2$$

$$J(v_i) = ||R_i - U v_j^T||^2 + \lambda ||v_j||^2$$

$$u_i = (V^T V + \lambda I)^{-1} V^T R_{i.}$$

$$v_j = (U^T U + \lambda I)^{-1} U^T R_{.j}$$

01 Backgrounds

- Previous work

- Learning Algorithms

2) ALS (Alternating Least Squares)

① Random Initialize

Ex)

0	3	2
5	1	2
4	2	1

→

0.576	1.453
-0.199	-1.218
2.730	0.480

User Latent(U)

×

0.367	-1.108	1.459
-0.339	0.897	0.453

Item Latent(V^T)

=

-0.282	0.666	1.498
0.340	-0.872	-0.842
0.838	-2.593	4.201

$U * V^T$

01 Backgrounds

- Previous work

$$u_i = (V^T V + \lambda I)^{-1} V^T R_{i.}$$

- Learning Algorithms

2) ALS (Alternating Least Squares)

② Fix Item Latent

Ex)

0	3	2
5	1	2
4	2	1

$u_1 = (V^T V + \lambda I)^{-1} V^T R_{1.}$ = [0.315, 3.297]

$u_2 = (V^T V + \lambda I)^{-1} V^T R_{2.}$ = [1.112, 0.543]

$u_3 = (V^T V + \lambda I)^{-1} V^T R_{3.}$ = [0.323, 0.915]

0.576	1.453
-0.199	-1.218
2.730	0.480

Original User Latent

0.315	3.297
1.112	0.543
0.323	0.915

Updated User Latent

$$v_j = (U^T U + \lambda I)^{-1} U^T R_{.j}$$

- Learning Algorithms

2) ALS (Alternating Least Squares)

③ Fix User Latent

Ex)

0	3	2
5	1	2
4	2	1

$v_1 = (U^T U + \lambda I)^{-1} V^T R_{1.}$ = [1.974, -2.317]

$v_2 = (U^T U + \lambda I)^{-1} V^T R_{2.}$ = [0.699, 0.634]

$v_3 = (U^T U + \lambda I)^{-1} V^T R_{3.}$ = [0.456, -0.036]

0.367	-1.108	1.459
-0.339	0.897	0.453

Original Item Latent

1.974	0.699	0.456
-2.317	0.634	-0.036

Updated Item Latent

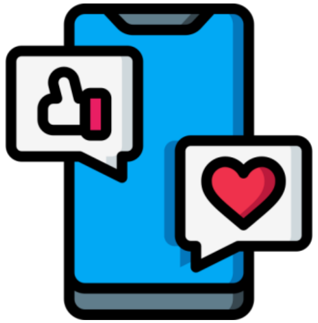
All update
(epoch : 1)

02 Introduction

- Datasets for Recommendation

- Explicit feedback

  - Not easily collected



Sparse Matrix

4	?	3	?	?	?
?	1	?	?	2	2
?	4	?	1	?	?

1~ 5 ratings

02 Introduction

- Datasets for Recommendation

- Implicit Feedback

  - Can be easily collected

Dense Matrix

2	0	3.4	0	0	0
0	1	1.2	3	2	0
0	4.1	0	1	0	0

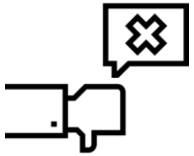
Watching Time

02 Introduction

- Implicit Feedback's prime characteristics

A. No Negative feedback
B. Inherently noisy
C. Numerical value of implicit feedback indicates confidence
D. Require appropriate Evaluation measures

Can infer which items they probably like



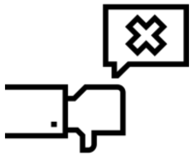
Hard to infer which items a user did not like

- Watching Time : 0
  ➔ They really did not like
  ➔ They don't know that show
  ➔ They are not available to watch it

02 Introduction

- Implicit Feedback’s prime characteristics

A. No Negative feedback
B. Inherently noisy
C. Numerical value of implicit feedback indicates confidence
D. Require appropriate Evaluation measures



Only guess their preference and true motives

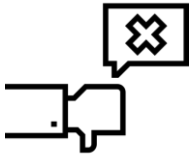
- Buying Something
  ➔ For gift

- Watching TV on a particular channel a particular time
  ➔ Be asleep

02 Introduction

- Implicit Feedback's prime characteristics

A. No Negative feedback
B. Inherently noisy
C. Numerical value of implicit feedback indicates confidence
D. Require appropriate Evaluation measures



A larger value is not indicating a higher preference

- One time event is caused by various reasons

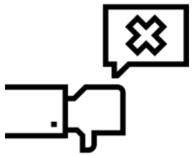
However, the numerical value of the feedback is definitely useful

- A recurring event is more likely to reflect the user opinion

02 Introduction

- Implicit Feedback’s prime characteristics

A. No Negative feedback
B. Inherently noisy
C. Numerical value of implicit feedback indicates confidence
D. Require appropriate Evaluation measures



Have to take into account availability of the item

- Unclear how to evaluate a show that has been watched more than once
- How to compare two shows that are on at the same time

03 Our model

- **Notation**

r_{ui} Implicit Feedback Datasets
- observations for user actions
- value 0 : no action

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

- Need to have difference confidence levels

$$c_{ui} = 1 + \alpha r_{ui}$$

$$c_{ui} = 1 + \alpha \log(1 + r_{ui}/\epsilon).$$

- Have some minimal confidence in p_{ui} for every user-item pair
- Confidence in $p_{ui} = 1$ increases accordingly
- α = 40 was found to produce good results

03 Our model

- **Cost function**

$$\min_{x_\star, y_\star} \sum_{u,i} \underline{c_{ui}} (\underline{p_{ui} - x_u^T y_i})^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

- **Learning Algorithm**

In Implicit Datasets, $m * n$ ➔ a few billion

1) SGD : parameter number (n*k + m*k)
2) ALS : cost function becomes convex
   ➔ Its global minimum can be readily computed

03 Our model

- **Learning Algorithms on Implicit Datasets**

\- Confidence + Basic Cost function + Regularization

$$J = C||Y - X\beta||^2 + \lambda||\beta||^2$$

$$= ||\sqrt{C}Y - \sqrt{C}X\beta||^2 + \lambda||\beta||^2$$

$$\sum e^2 = e^T \cdot e = (\sqrt{C}Y - \sqrt{C}X\beta)^T(\sqrt{C}Y - \sqrt{C}X\beta)$$

$$= (Y^T\sqrt{C} - \beta^T X^T\sqrt{C})(\sqrt{C}Y - \sqrt{C}X\beta)$$

$$= (Y^T CY - Y^T CX\beta - \beta^T X^T CY + \beta^T X^T CX\beta)$$

$$= (Y^T CY - 2\beta^T X^T CY + \beta^T X^T CX\beta) + \lambda||\beta||^2$$

$$\frac{\sigma \sum e^2}{\sigma\beta} = -2X^T CY + 2X^T CX\beta + 2\lambda\beta = 0$$

$$(X^T CX + \lambda I)\beta = X^T CY$$

$$\beta^* = (X^T CX + \lambda I)^{-1} X^T CY$$

03 Our model

- **Analytic expression by differentiation**

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i)$$

- **Explaining Recommendations**

$$\hat{p}_{ui} = y_i^T x_u = y_i^T W^u Y^T C^u p(u)$$

Define

$W^u =$ $f \times f$ matrix $(Y^T C^u Y + \lambda I)^{-1}$: a weighting Matrix associated with user u

$s_{ij}^u = y_i^T W^u y_j$: weighted similarity between items i and j from u's viewpoint

$$\hat{p}_{ui} = \sum_{j: r_{uj} > 0} \underline{s_{ij}^u} \, \underline{c_{uj}}$$

04 Experiments

- Datasets

- **Data description**

  - digital TV service
    - Size : 300,000 set top box
    - Features : channels, timestamp
    - Unique programs: 17,000

  - Training set
    - r_{ui} : watching minutes during a 4-week
    - # of non-zero : 32,000,000

  - Test set
    - r_{ui}^{t} : all channel tune events during the single week following a 4-week
    - Remove “easy” predictions (had been watched by that user during the training period)
    - # of non-zero : 2,000,000

- Datasets

- **Data Preprocessing**
  - Log scaling
    - $c_{ui} = 1 + \alpha \log(1 + r_{ui}/\epsilon).$
    - The tendency to watch the same programs repeatedly
  - Down Weight
    - Watch the single channel for many hour ➔ less expected to reflect real preference
    - $\frac{e^{-(at-b)}}{1+e^{-(at-b)}}$: assign t-th show down weight (a=2, b=6)

04 Experiments

- Datasets

- **Evaluation methodology**

  - A good model should be able to rank relevant items towards the top of the list

  - $rank_{ui}$: percentile-ranking of program i for user u
    - $rank_{ui} = 0\%$: program i is predicted to be the most desirable for user u
    - $rank_{ui} = 100\%$: program i is predicted to be the least desirable for user u

  - $$\overline{rank} = \frac{\sum_{u,i} r_{ui}^t rank_{ui}}{\sum_{u,i} r_{ui}^t}$$

Ex) $r_{u1}^t = 2, r_{u2}^t = 5$

Model 1 $rank_{u1} = 1\%,\ rank_{u2} = 90\%$ ➔ $\overline{rank} = \frac{2*0.01+5*0.9}{2+5} = 0.64\%$

Model 2 $rank_{u1} = 90\%,\ rank_{u2} = 1\%$ ➔ $\overline{rank} = \frac{2*0.9+2*0.01}{2+5} = 0.26\%$

04 Experiments

- Results

- **More Experiments (10~200 factors)**

1) $$\min_{x_\star, y_\star} \sum_{u,i} (r_{ui} - x_u^T y_i)^2 + \lambda_1 \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

   - 100 factor: $\overline{rank}$: 13.4%

2) $$\min_{x_\star, y_\star} \sum_{u,i} (p_{ui} - x_u^T y_i)^2 + \lambda_2 \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

   - 100 factor: $\overline{rank}$: 10.49%

3) $$\min_{x_\star, y_\star} \sum_{u,i} c_{ui} (p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

   - 100 factor: $\overline{rank}$: 8.56%

- **Results on Rank**

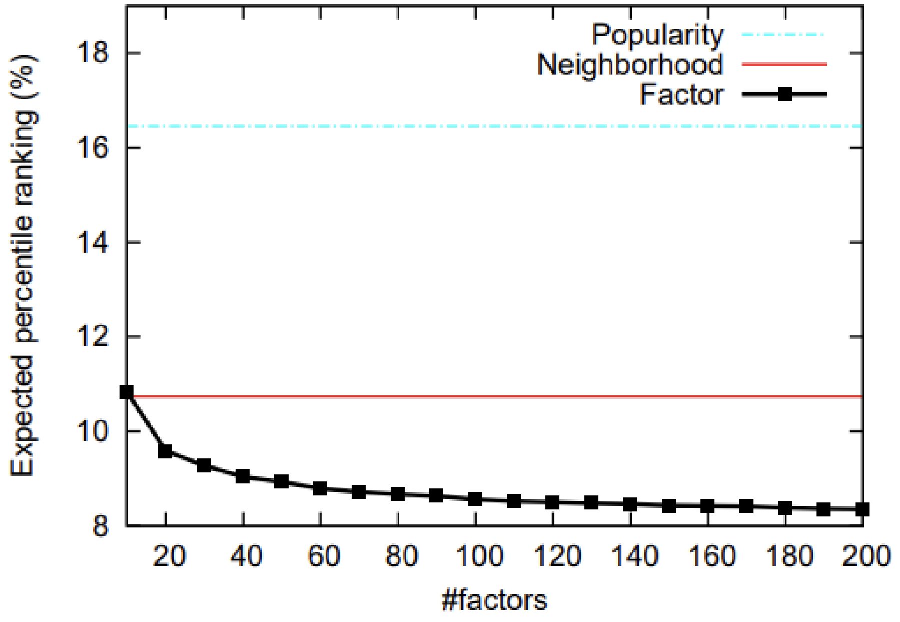
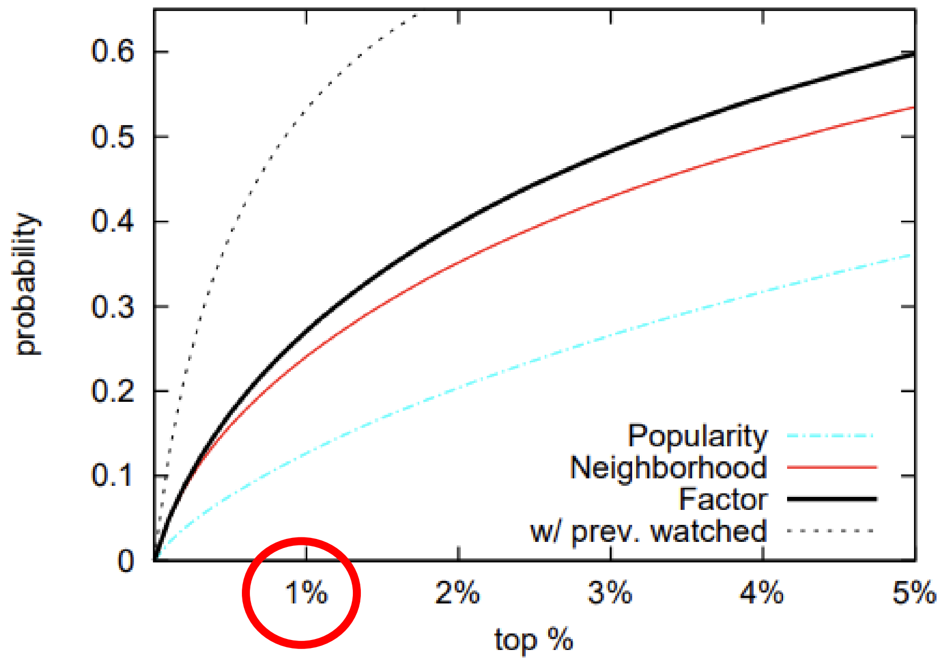


Figure 1. Comparing factor model with popularity ranking and neighborhood model.

04 Experiments

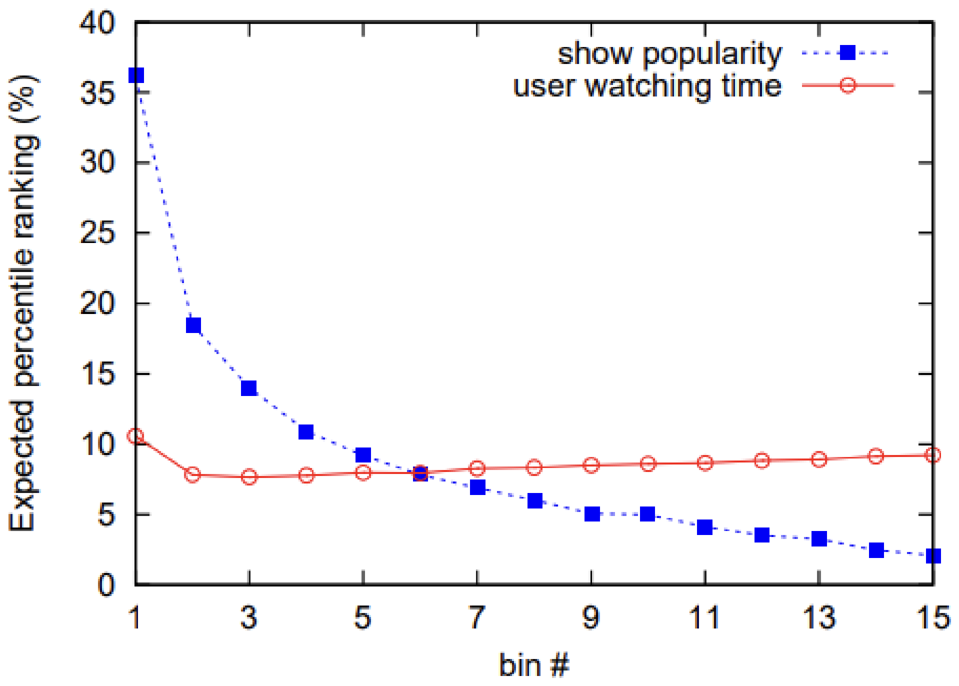
- Results

- **Results on Probability**



- Our model : Top 1% is about 27% of the watched time
  - Previously watched show in Test data
    ➔ high predictive accuracy (But, not useful)

- **Results on Performance**



- Easier to predict popular programs
- Not do much better for heavy watchers
  - heterogeneous accounts

04 Experiments

- Results

- **Utility of our recommendation explanations**

So You Think You Can Dance	Spider-Man	Life In The E.R.
Hell's Kitchen Access Hollywood Judge Judy Moment of Truth Don't Forget the Lyrics	Batman: The Series Superman: The Series Pinky and The Brain Power Rangers The Legend of Tarzan	Adoption Stories Deliver Me Baby Diaries I Lost It! Bringing Home Baby
Total Rec = 36%	Total Rec = 40%	Total Rec = 35%

- Previous Matrix Factorization can't explain Recommendation
- Top 5 shows only explain between 35%~40% of the recommendation

05 Conclusion

- ALS algorithm for implicit feedback is effective with two modifications
  - Accounting for all possible user-item interactions
  - Using the concept of confidence, preference in the cost function

- The algebraic construct of the user and item factors can be leveraged for providing explanations for the recommendations

- Changing the model by taking advantage of the characteristics of the data may affect the performance improvement

06 Discussion

A. No Negative feedback ➔ By setting a minimum threshold on r_{ui}

B. Inherently noisy ➔ Log scaling, Down Weight

C. Numerical value of implicit feedback ➔ Confidence

D. Require appropriate Evaluation measures ➔ $rank_{ui}$ (Not RMSE)

06 Discussion

- Computational Bottleneck (paper p.4)

$$x_u = (\underline{Y^T C^u} Y + \lambda I)^{-1} Y^T C^u p(u)$$

→ Require time $O(f^2 n)$

- A significant speedup

$$Y^T C^u Y = \underline{Y^T Y} + \underline{Y^T (C^u - I) Y}$$

➔ Only non_zero elements

```
self._V_t = np.transpose(self._V)
self._V_t_V = self._V_t.dot(self._V)
self.optimize_User_latent()

self._U_t = np.transpose(self._U)
self._U_t_U = self._U_t.dot(self._U)
self.optimize_Item_latent()
```

- My Implementation On Explicit Datasets
  - https://github.com/rlagywns0213/2021_Summer_Internship/tree/main/RecSys/OCCF
  - Implicit Datasets?
  - TO DO : only non_zero elements

06 Discussion

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

```
bottleneck problem
기존 loss: 40.24770828391712
time to compute rank : 0.1506
Iteration: 1, loss = 0.742698, test_rank = 0.3872, time : 27.4694
time to compute rank : 0.1655
Iteration: 2, loss = 0.614921, test_rank = 0.3640, time : 30.5546
time to compute rank : 0.1556
Iteration: 3, loss = 0.591714, test_rank = 0.3725, time : 30.3122
time to compute rank : 0.1596
Iteration: 4, loss = 0.582945, test_rank = 0.3714, time : 30.3857
time to compute rank : 0.1536
Iteration: 5, loss = 0.578381, test_rank = 0.3761, time : 29.8251
time to compute rank : 0.1606
Iteration: 6, loss = 0.575621, test_rank = 0.3827, time : 30.3143
time to compute rank : 0.1566
Iteration: 7, loss = 0.573788, test_rank = 0.3770, time : 29.6165
time to compute rank : 0.1606
Iteration: 8, loss = 0.572478, test_rank = 0.3836, time : 33.0068
time to compute rank : 0.1536
Iteration: 9, loss = 0.571488, test_rank = 0.3806, time : 34.3357
time to compute rank : 0.1745
Iteration: 10, loss = 0.570710, test_rank = 0.3811, time : 34.3407
time to compute rank : 0.1616
Iteration: 11, loss = 0.570084, test_rank = 0.3835, time : 33.5701
time to compute rank : 0.1626
```

$$Y^T C^u Y = Y^T Y + Y^T (C^u - I) Y$$

With all elements

```
Reduce bottleneck problem
기존 loss: 40.24770828391712
time to compute rank : 0.1695
Iteration: 1, loss = 0.742698, test_rank = 0.3872, time : 79.3030
time to compute rank : 0.1656
Iteration: 2, loss = 0.614921, test_rank = 0.3640, time : 78.9446
time to compute rank : 0.1676
Iteration: 3, loss = 0.591714, test_rank = 0.3725, time : 82.0225
time to compute rank : 0.1965
Iteration: 4, loss = 0.582945, test_rank = 0.3714, time : 85.8256
time to compute rank : 0.1685
Iteration: 5, loss = 0.578381, test_rank = 0.3761, time : 92.4284
time to compute rank : 0.1695
Iteration: 6, loss = 0.575621, test_rank = 0.3827, time : 87.0184
time to compute rank : 0.1765
Iteration: 7, loss = 0.573788, test_rank = 0.3770, time : 89.3223
time to compute rank : 0.1685
Iteration: 8, loss = 0.572478, test_rank = 0.3836, time : 89.2158
time to compute rank : 0.1626
Iteration: 9, loss = 0.571488, test_rank = 0.3806, time : 100.4290
time to compute rank : 0.1685
Iteration: 10, loss = 0.570710, test_rank = 0.3811, time : 93.9412
time to compute rank : 0.1676
```