Movie Recommendation System

Group 11

Chenjun Tang

Zhimin Sun

Ziqihong Yue

Introduction

- Inspired by online streaming platforms like Netflix, Amazon Prime Video, Youtube, etc. Those recommendation services provide recommendation based on user preference and browse history.

- We used three different algorithms to develop a movie recommendation system:
  - User-Based Collaborative Filtering - find popular movies among similar users
  - Latent Factor Model - recommend using matrix factorization
  - PersonalRank Algorithm - recommend using graphic representation of user-movie relationship

MovieLens 1M Dataset

- Over 1 million ratings from 6000 users on 4000 movies

- Ratings are made on 5-star scale

- Each user has at least 20 ratings

UserID	MovieID	Rating	Timestamp
1	1193	5	9.78E+08
1	661	3	9.78E+08
1	914	3	9.78E+08
1	3408	4	9.78E+08
1	2355	5	9.79E+08
1	1197	3	9.78E+08
1	1287	5	9.78E+08
1	2804	5	9.78E+08
1	594	4	9.78E+08
1	919	4	9.78E+08
1	595	5	9.79E+08

User-Based Colaborative Filtering

- Find the cosine similarity of users to the target user U using.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Predict the rating of the movies not rated by the target user.

$$s(u, i) = \bar{r_u} + \frac{\sum_{v \in V}(r_{vi} - \bar{r_v}) * w_{uv}}{\sum_{v \in V} w_{uv}}$$

Evaluation of UserCF

- Use part of the data as input.
- Generate a list of recommended movies.
- Compare the results with the unused data.

```
Start for user 341 ,  1 /10
Accuracy:  0.2
Cost time: 18.807611
Start for user 5888 ,  2 /10
Accuracy:  0.75
Cost time: 22.797583
Start for user 2824 ,  3 /10
Accuracy:  0.75
Cost time: 19.792316
Start for user 2214 ,  4 /10
Accuracy:  0.4
Cost time: 18.361912
Start for user 914 ,  5 /10
Accuracy:  0.3
Cost time: 17.759914
Start for user 1029 ,  6 /10
Accuracy:  0.5
Cost time: 20.297516
```

Latent Factor Model (LFM) - Introduction

- Latent factors are hidden factors unseen in the data set.

- Construct user-item matrix by reading ratings data. Value 1 - interested / Value 0 - uninterested.

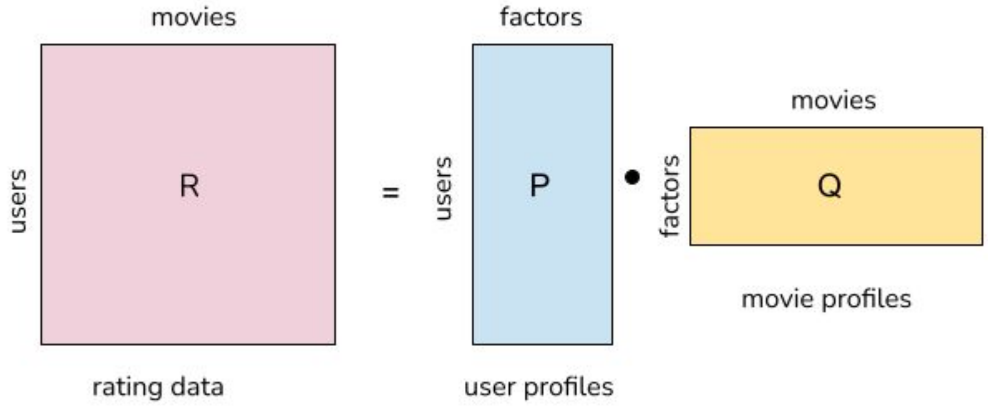
- Using matrix factorization to make recommendation.

Latent Factor Model (LFM) - Matrix Factorization

‘Decompose’ rating matrix into two product of two lower dimension matrices

- user profiles $m \times k$
- movie profiles $n \times k$

k is one of our hyper-parameters, which represents the amount of latent factors we’re using to estimate the ratings matrix.



Latent Factor Model (LFM) - Optimization

- $$loss = \sum_{(u,i)\in D} (R_{ui} - \hat{R}_{ui})^2 = \sum_{(u,i)\in D} (R_{ui} - \sum_{k=1}^{K} P_{uk} \cdot Q_{ik})^2 + \lambda||P_u||^2 + \lambda||Q_i||^2$$

- Optimize matrices P and Q by using Stochastic Gradient Descent, introducing two more hyper-parameters: learning rate and epoch.

- In each epoch, iterate through every known rating in our original $m \times n$ matrix.

- Then, get a error by subtracting the original rating value by the dot product of the original rating's user's row in P and its item's column in Q.

Latent Factor Model (LFM) - Prediction

- Get prediction by P dot product Q, and output the top 10 results for each user.

- Prediction format: (movie id, preference).

```
LFM - predict
(3590, 0.9926766793235611)
(3135, 0.9916098569533052)
(2924, 0.9914715224564732)
(2802, 0.9913050796683407)
(2370, 0.9908468029755513)
(2875, 0.9907864108630289)
(3549, 0.989553195859362)
(3602, 0.9891085642659722)
(3394, 0.988979206005912)
(2758, 0.9888948879079363)
```

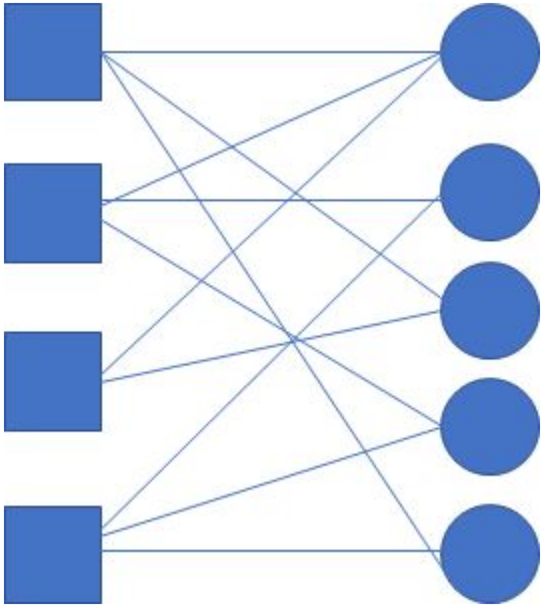
Latent Factor Model (LFM) - Evaluation

- Evaluation metrics:
  - Absolute error (AE) = true rating - prediction rating generated by matrics P and Q
  - Precision = TP/(TP+FP)
  - Recall = TP/(TP+FN)
- Part of our evaluation result; Randomly pick 10 users at each evaluation

```
Evaluation started
--------------------Evaluation----Absolute Error---------
userID: 4593, AE: 2.0735
userID: 1817, AE: 1.7777
userID: 2037, AE: 2.8433
userID: 1234, AE: 4.0462
userID: 2799, AE: 3.6447
userID: 1378, AE: 2.9804
userID: 2331, AE: 4.9819
userID:  605, AE: 2.6308
userID: 1757, AE: 2.3505
userID: 1206, AE: 3.4598
userID: 1206, average AE: 3.0789
```

```
--------------------Evaluation---precision and recall--------------------
userID:  633, precision: 0.7778
userID:  633, recall   : 0.8750
userID: 1962, precision: 0.9239
userID: 1962, recall   : 0.8019
userID: 3500, precision: 0.8462
userID: 3500, recall   : 0.9167
userID: 4762, precision: 1.0000
userID: 4762, recall   : 0.9286
userID: 1042, precision: 0.5000
userID: 1042, recall   : 0.8889
userID: 4335, precision: 0.9032
userID: 4335, recall   : 0.8750
userID: 3040, precision: 0.4286
userID: 3040, recall   : 0.7500
userID: 3945, precision: 0.9091
userID: 3945, recall   : 0.9434
userID: 3404, precision: 1.0000
userID: 3404, recall   : 1.0000
userID: 5486, precision: 0.8571
userID: 5486, recall   : 0.7500
average precision:   0.8146
average recall:      0.8729
--------------------------------------------------------------------------
Evaluation ended
```

PersonalRank Algorithm - Intro



· A recommendation algorithm based on graph

· The graph is a representation of the relationships between users and items

PersonalRank Algorithm

· Initiate a random walk starting from the user who wants recommendations

· Proceed to the next available node with probability α and retreat with probability 1-α

· The probability of the random walk ending at each node will converse

PersonalRank Algorithm - Evaluation

· Use a partial data as input for recommendation

· Compare the results with the unused data

Ziqihong Yue

Proof of Concept

Movies Recommendation

Upload a file containing your preference of movies and see recommendations based on your review.

Model Selection

PersonalRank Model

Upload File Below

Choose File input.Txt

See Recommendations

Comparison

	User-based Colaborative Filtering	Latent Factor Model	PersonalRank
Average prediction running time	19s	3s	15s
Drawbacks	• Slow on huge datasets • Limited ability to expand on the users' existing interests. • Unstable if user give too many negative ratings	• Rating matrix is a sparse matrix • Training is time-consuming (About 6 hours for 5 iterations) • Hard to recommend for new users	• Graph initializing and converging are slow • Does not differentiate between top ratings with average ratings • Negative ratings are not efficiently used

Thanks for listening!