Movie Recommendation

Shengjie Li, Junlin Lu

Rutgers University

April 24, 2020

Project Background and Goal

- Background
  A recommender system is a system that seeks to predict the "rating" or "preference" a user would give to an item [1].
  Recommender systems are used in a variety of areas, e.g., playlist generator for music and video services, product recommendation for online shopping, content recommendation for news services.
- Goal
  To build a movie recommender system and predict users' ratings to recommended movies.

Dataset

MovieLens Latest Datasets-small[1]

A public dataset provided by MovieLens, which contains $100,000$ ratings and $3,600$ tag applications applied to $9,000$ movies by more than 600 users

The dataset contains 4 csv files:

- `ratings.csv`: (userId, movieId, rating, timestamp)
- `movies.csv`: (movieId, title, genres)
- `tags.csv`: (userId, movieId, tag, timestamp)
- `links.csv`: (movieId, imdbId, tmdbId)

[1] https://grouplens.org/datasets/movielens/

Dataset

The first 5 rows of each file are shown in the following tables:

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

(a) `ratings.csv`

userId	movieId	tag	timestamp
2	60756	funny	1445714994
2	60756	Highly quotable	1445714996
2	60756	will ferrell	1445714992
2	89774	Boxing story	1445715207
2	89774	MMA	1445715200

(b) `tags.csv`

movieId	title	genres	timestamp
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1445714994
2	Jumanji (1995)	Adventure|Children|Fantasy	1445714996
3	Grumpier Old Men (1995)	Comedy|Romance	1445714992
4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1445715207
5	Father of the Bride Part II (1995)	Comedy	1445715200

(c) `movies.csv`

Preprocessing - Bias Removal

Ratings contain bias!
To formalize, we have:

$$r_{xi} = b_x + b_i + r'_{xi}$$

- r_{xi}: user x's rating of movie i
- $b_x = \overline{r_x} - \overline{r}$: rating deviation of user x ((avg. rating of user x) – (overall mean movie rating))
- $b_i = \overline{r_i} - \overline{r}$: the rating deviation of movie i
- r'_{xi}: real unbiased rating.

$$r'_{xi} = r_{xi} - b_x - b_i$$
$$r'_{xi} = r_{xi} + 2 * \overline{r} - \overline{r_i} - \overline{r_x}$$

Preprocessing - Time Correction

Timestamp: seconds since 12:00AM of January 1, 1970 UTC

$$\downarrow$$

Timestamp: weeks since the first review of each movie

$$t = \frac{t - t_{first}}{3600 \times 24 \times 7}$$

Preprocessing - Time Correction

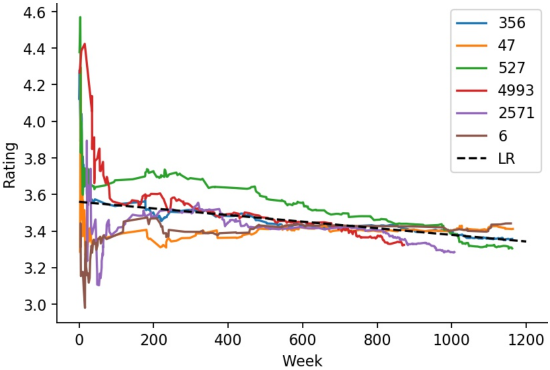

Figure: Ratings of some movies over time

$$r = kt + b$$

$$k = \frac{\sum_{t,r}(t - \bar{t})(r - \bar{r})}{\sum_r (r - \bar{r})^2} = -4.5739 \times 10^{-20}, b = \bar{r} - k\bar{t} = 3.5596$$

Preprocessing - Time Correction

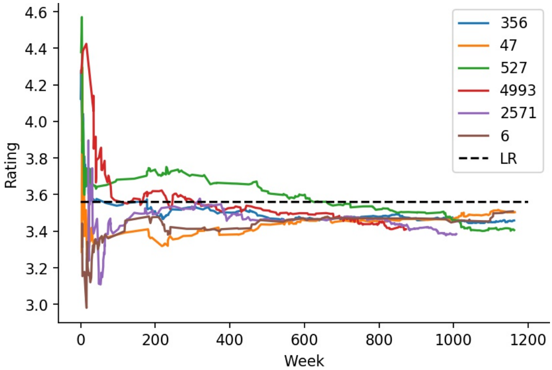

Figure: Ratings of some movies over time

$$r' = r - kt$$

Preprocessing - Dataset Splitting

5-fold cross validation:

1. Split the dataset into 5 equal sized parts
2. Of these 5 parts, a single part is retained as the testing dataset, and the remaining 4 parts are used as the training dataset. (We make sure each training dataset has all movies and users)
3. Repeat this process for 5 times

	# of rows in train set	# of rows in test set	# of users in test set	# of movies in test set
$fold_1$	82729	18107	609	3991
$fold_2$	82729	18107	608	3950
$fold_3$	82729	18107	609	3985
$fold_4$	82730	18106	608	3946
$fold_5$	82730	18106	610	3963

Table: Some statistics information of our dataset

Content-based Model

Main idea: recommend movies to user x that are similar to previous movies rated highly by x.

- Item profile: ‘title’, ‘genres’ from `movies.csv`, ‘tag’ from `tags.csv`
  TF-IDF:

$$TF_{ij} = log(f_{ij} + 1)$$
$$IDF_i = log\frac{N}{n_i}$$
$$TF\text{-}IDF_{ij} = TF_{ij} \times IDF_i$$

f_{ij}: frequency of feature i in movie j
n_i: the number of movies that have feature i
N: the total number of movies.

Content-based Model

- User profile: 'title', 'genres' from `movies.csv`, 'tag' from `tags.csv`
  1. Weighted average of all rated item profiles
  2. Average of item profiles of the top 20 rated movies
- Movie recommendation:
  Use user profile x to calculate cosine similarity score against all movies, and then find k movies with the highest cosine similarity scores:

$$\underset{i}{\operatorname{argmax}} \, cos(x, i) = \underset{i}{\operatorname{argmax}} \frac{x \cdot i}{\|x\| \cdot \|i\|}$$

- Rating prediction:
  For rating prediction given user x and movie i, we find the most similar 10 movies to movie i that user x has watched and rated, and let the average rating of this 10 movies be the predicted rating for movie i.

User-Based Collaborative Filtering

For target user:

- Find a set of similar users
- From this set, Find the movies that these users rated and recommend the ones with the highest rating

Use cosine similarity to calculate user similarities:

$$Sim_{uv} = \frac{|M(u) \cap M(v)|}{\sqrt{|M(u)||M(v)|}}$$

Where *M(u)* represents the set of movies user *u* has watched.

Item-Based Collaborative Filtering

For each movie target user watched:

- Find similar movies based on their cosine similarity
- For each movie target user didn't watch, rate them based on the weighted sum
- Recommend top k movies with the highest rating

Movie similarity:

$$Sim_{mn} = \frac{|V(m) \cdot V(n)|}{\sqrt{|V(m)||V(n)|}}$$

Where V is the rating vector of a movie.
Predicted rating for a movie:

$$r(m) = \sum_{n \subset K} r(n) \times Sim_{uv}$$

Where K is the set of top k similar movies.

Dimensional reduction:

- Remove noisy features
- Find hidden correlations between users and movies

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$$

- U: User feature matrix, represents the users' interest
- Σ: Diagonal matrix of singular values
- V^T: Movie feature matrix, represents each feature's relevance to each movie

After dimensional reduction, we take the dot product of U, Σ and V^T to predict the rating and recommend the movies with the highest rating.

Results and Conclusions

Model	Precision	Recall	F-measure	NDCG	MAE	RMSE
CB	0.035	0.012	0.018	0.014	0.689	0.899
UserCF	0.240	0.081	0.121	0.675	**0.631**	**0.821**
ItemCF	0.232	0.073	0.121	0.668	0.745	0.849
LF	**0.264**	**0.089**	**0.133**	**0.719**	1.688	1.959

Table: Average Performance of Each Model

Conclusions:

- The latent factor model performs best in the movie recommendation task
- The user-based collaborative filtering model performs best in the rating prediction task
- The content-based model doesn't have a competitive performance in the movie recommendation task, but performs good in the rating prediction task.

Future Work

Future works:

- Use more data
- Deal with missing value more effectively in Latent Factor Model
- Build a hybrid model
- Extract more features for the content-based model

References

[1] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*, pages 1–35. Springer, Boston, MA, USA, 10 2010.

Thank You