Final Report. Movie Recommender System. Assignment 2. PMLDL

Makar Shevchenko
B20-DS-01
Innopolis University
Innopolis, Russia
m.shevchenko@innopolis.university

I. Introduction

Recommendation systems are used in modern content platform to suggest new content to users that they would probably like. In 1998 authors of such a platform MovieLens collected and published movie reviews dataset MovieLens 100K [1]. It then then become a benchmark for recommendation systems. In this work I explored, preprocessed the dataset and applied it for training and evaluation of matrix factorization based recommendation model LightFM.

II. Data analysis

The dataset consist of three main and several supporting files. The main files are u.data containing 100,000 movie reviews, u.user containing demographic information about 943 users, and u.item containing information about 1682 movies. Each user in the dataset have at least 20 reviews. The supporting files are u.info containing information about count of the users, movies, and reviews; u.genre listing movie genres; u.occupation listing user occupations; two scripts for generating review data subsets; and 5 subsets themselves.

A. Data Exploration

Figs. 1, 5, and 2 show a few rows from u.data, u.item, and u.user, respectively. The u.item contain movie name, release date, IMDb url, genre, and everywhere empty video_release_date. The u.user data include age, gender, occupation, and zip code. I derived state column from zip code to better understand the locations.

	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116

Fig. 1. A few rows from u.data

Figs. 3, 6, and 4 illustrate statistics of numberic columns from u.data, u.item, and u.user, respectively. Interestly, medium rating is greater than the exact middle of the review borders three. The youngest user is seven years old, the oldest - 73, the median is 34.

	user_id	age	gender	occupation	zip_code	state
0	1	24	M	technician	85711	AZ
1	2	53	F	other	94043	CA
2	3	23	M	writer	32067	FL

Fig. 2. A few rows from u.user

	user_id	item_id	rating	timestamp
count	100000.00000	100000.000000	100000.000000	1.000000e+05
mean	462.48475	425.530130	3.529860	8.835289e+08
std	266.61442	330.798356	1.125674	5.343856e+06
min	1.00000	1.000000	1.000000	8.747247e+08
25%	254.00000	175.000000	3.000000	8.794487e+08
50%	447.00000	322.000000	4.000000	8.828269e+08
75%	682.00000	631.000000	4.000000	8.882600e+08
max	943.00000	1682.000000	5.000000	8.932866e+08

Fig. 3. Statistics of numberic columns of u.data

After analyzing statistics analysed distributions. From the subfigures in Fig. 7 I made the following observations. The count of reviewed movies per user vary, peaking in 550 of movies and dropping till 30. The distribution is similar for counts of users reviewed per movie, peaking 550 users per movie and dropping till 1 review. The most common ratings

	user_id	age
count	943.000000	943.000000
mean	472.000000	34.051962
std	272.364951	12.192740
min	1.000000	7.000000
25%	236.500000	25.000000
50%	472.000000	31.000000
75%	707.500000	43.000000
max	943.000000	73.000000

Fig. 4. Statistics of numberic columns of u.user

	movie_id	movie_title	release_date	video_release_date	IMDb_URL	unknown	Action	Adventure	Animation	Children's	...	Film-Noir	Horror
0	1	Toy Story (1995)	1995-01-01	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	0
1	2	GoldenEye (1995)	1995-01-01	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	...	0	0
2	3	Four Rooms (1995)	1995-01-01	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	...	0	0

Fig. 5. A few rows from u.item

	movie_id	video_release_date	unknown	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	...
count	1682.000000	0.0	1682.000000	1682.000000	1682.000000	1682.000000	1682.000000	1682.000000	1682.000000	1682.000000	...
mean	841.500000	NaN	0.001189	0.149227	0.080262	0.024970	0.072533	0.300238	0.064804	0.029727	...
std	485.695893	NaN	0.034473	0.356418	0.271779	0.156081	0.259445	0.458498	0.246253	0.169882	...
min	1.000000	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
25%	421.250000	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
50%	841.500000	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
75%	1261.750000	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	...
max	1682.000000	NaN	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...

Fig. 6. Statistics of numberic columns of u.item

are 4, 3, and 5. 5 is given as triple more times as 1. The time of the reviews has almost uniform distribution with some drop and peaks at 20% of periods.

Subfigures of Fig. 8 show that the data is biased towards movies of 1990-2000 as opposite to ¡1990 and not present for ¿2000 [2]. This is because the dataset was collected in 1997-1998. Due to the bias the usage of release date in modelling might be unpredictable. Also the bias is towards movies of genre drama and comedy as opposite for Film-Noir and Fantasy. This might cause underfitting of embeddings of last genres.

Fig. 9 suggest that distribution of movies among genres persist throughout time [3]. The only parameter that change - number of released movies in general.

From Fig. 10 I found that the data is biased towards: people in age of 20-30 as opposite to ¡20 and ¿50; males as opposite to females; students as opposite to doctors; users from CA state as opposite to AE state. This may lead to biased modelling.

After explorative data analysis I checked the quality of the data. I counted the percentage of null values in columns of u.data, u.item, and u.user. I found that video_release_date in u.item has all the values as nulls. Also I noticed that release_date and IMDb_URL have few null values. Fortunately, I did not used these columns for modelling, and avoided cleaning. The rest of the data were normal. No dublicate rows.

B. Data Preprocessing

In data preprocessing step I performed data selection, construction, and formatting.

In selection stage I removed timestamps of the reviews as my model do not use time features. Among columns of U Item I decided that only genre columns apropriate feature for such small amount of data. I removed the rest feature columns according to the following reasoning. The release date and year columns looks noisy as the data is biased towards releases of shord time frame of 1990-1997. IMDb url is not understandable for the models. Video release date is empty. Movie titles are too abstract for the model, so it is unlikely that the meaning of the words in it relate to user rating. From users data I removed the zip code as I already derived informative data about state. The rest age, gender, occupation, and state columns I preserve as user features.

I did not perform data cleaning as the columns I pick were normal.

As part of data construction step I divided age to different groups as people of different ages has different behavior. The groups are 14-, 14-30, 30-50, 50+.

Finally, I encoded categorical features of my data and stored preprocessed data in data/interim directory for my model.

III. Model Implementation

To decide what model to use I first discovered available options. I asked ChatGPT for classes of recommendataional models and it suggested me Collaborative Filtering, Content-Based Filtering, Matrix Factorization, Deep Learning-Based Recommender Systems, and Association Rule Mining. As I already have an experience with deep learning-based models such as language ones for POS-tagging, I decided to try something new for me - Matrix Factorization. I found existing implementation of that algorithm in LightFM Python library [4].

The idea of the matrix factorization is to learn embeddings for users and items in a shared latent space. This allows to

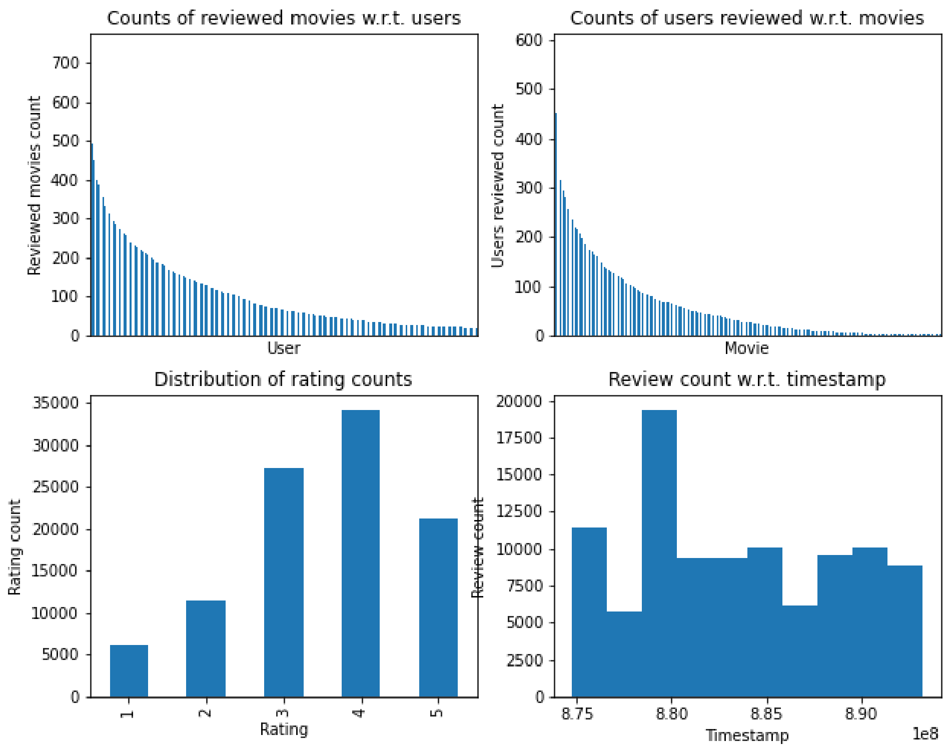

Fig. 7. Four visualizations of u.data: Counts of reviewed movies w.r.t. users, Counts of users reviewed w.r.t. movies, Distribution of rating counts, and Review count w.r.t. timestamp

estimate whether the interaction between user and a product would be positive. Algorithm derive embeddings from patterns in data, i.e. similar user scores for two movies say that they have similar meaning and same reaction from new users. The algorithm estimates the embeddings with stochastic gradient descent [5] Algorithm allows to use feature vectors of the users and vectors to improve model fitting.

IV. Model Advantages and Disadvantages

The advantages of LightFM model are the following. As a hybrid latent representation recommender the model excels in handling both collaborative and content-based information through feature-based embeddings. The model employs the WARP loss, suitable for precision@k optimization when only positive interactions are present.

The limitations include the absence of built-in support for predicting new users without retraining, potential over-reliance on provided features leading to underfitting, and a recommendation process that may need repeating for new users. Moreover, a lack of user/item features might result in underfitting when solely relying on features.

Overall, LightFM offers a versatile solution for movie recommendations, striking a balance between collaborative and content-based approaches, despite the highlighted limitations.

V. Training Process

To prepare the data for the model I used build-in tool for preparing the data: computing user-item interaction matrix and user/item-features matrices. I counted the interaction as positive if a user gave score 5, otherwise interaction does not exist. I splitted the data into train and test sets with proportion of 80% to 20%, respectively. Experimentally I found the set of hyperparameters giving the best metrics. The hyperparameters are: 40 - dimensionality of the latent space, 10 training epochs. The loss used is Weighted Approximate-Rank Pairwise as the LightFM authors recommend it when only positive intractions are used, as in my case. The loss "maximises the rank of positive examples by repeatedly sampling negative examples until rank violating one is found."

VI. Evaluation

I evaluated my model on the random 20% part of the reviews dataset. The model give the following results:

- ROC AUC: 0.896 on the test set and 0.911 on the train set. It is close to 1. This means that the model is good in distinguishing positive and negative classes;
- Reciprocal rank: 0.160 on the test set and 0.441 on the train set. This means that the model usually guess the recommendation right only on the sixth position;

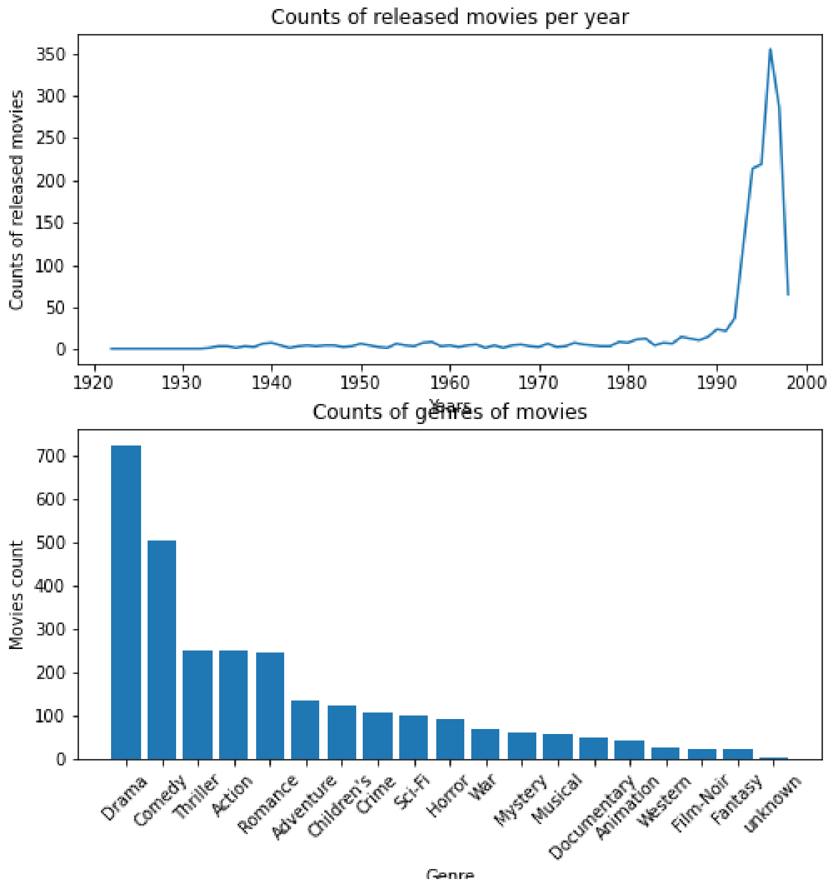

Fig. 8. Two visualizations of u.item: Counts of released movies per year and Counts of genres of movies

- Precision at k (from 1 to 10) is shown in left subfigure of Fig. 11. The meric is 0.06 at k=1, and 0.05 at k=10; It is ok with only 0.01 of true samples for a user. This means that model give right answer only in 1 out of 20 recommendations;
- Recall at k (from 1 to 10) is shown in right subfigure of Fig. 11. The meric is 0.01 at k=1, and 0.11 at k=10.

The metrics are defined as follows:

- ROC AUC metric for a model: the probability that a randomly chosen positive example has a higher score than a randomly chosen negative example;
- Reciprocal rank metric: 1 / the rank of the highest ranked positive example;
- Precision at k metric for a model: the fraction of known positives in the first k positions of the ranked list of results;
- Recall at k metric for a model: the number of positive items in the first k positions of the ranked list of results divided by the number of positive items in the test period.

An example of prediction is shown in Fig. **??**. Some of the recommended movies match known positives: Star Wars and Contact. This means that model trained embeddings correctly: the estimated scores match interactions from train data.

VII. Results

In this work I applied Matrix Factorization algorithm to recommend movies to user based on their score history to other movies. I used MovieLens 100K dataset and LightFM implementation of the method. I explored the data and find insights regarding demography of movie watchers, rating behaviour, and popularity of movie genres. I preprocessed the data: selected relevant features of user, movies and scores, formatted them and passed to the model for training. As a result, I got Reciprocal rank = 0.160, ROC AUC = 0.896, recall = 0.06 at k=1 and 0.05 at k=10, and precision = 0.01 at k=1 and 0.1 at k=10. A sample prediction is shown. Strengths and weaknesses of the model are discussed.

References

[1] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, p. 1–19, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1145/2827872

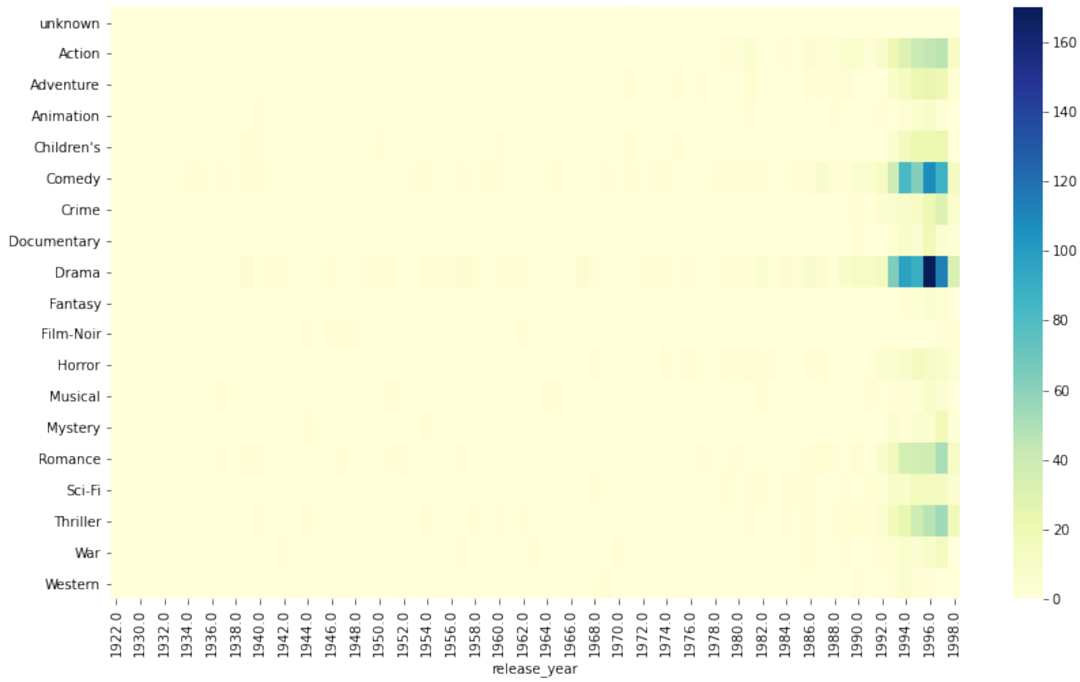

Fig. 9. Heatmap of user movie genres w.r.t. release year

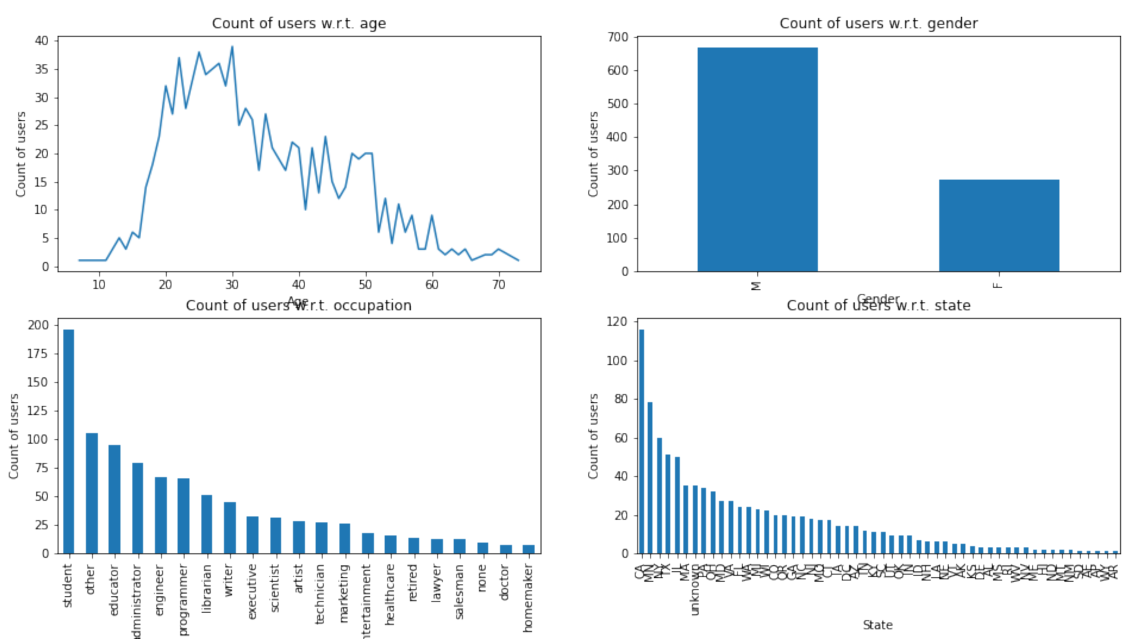

Fig. 10. Four visualizations of u.user: Count of users w.r.t. age, Count of users w.r.t. gender, Count of users w.r.t. occupation, Count of users w.r.t. state

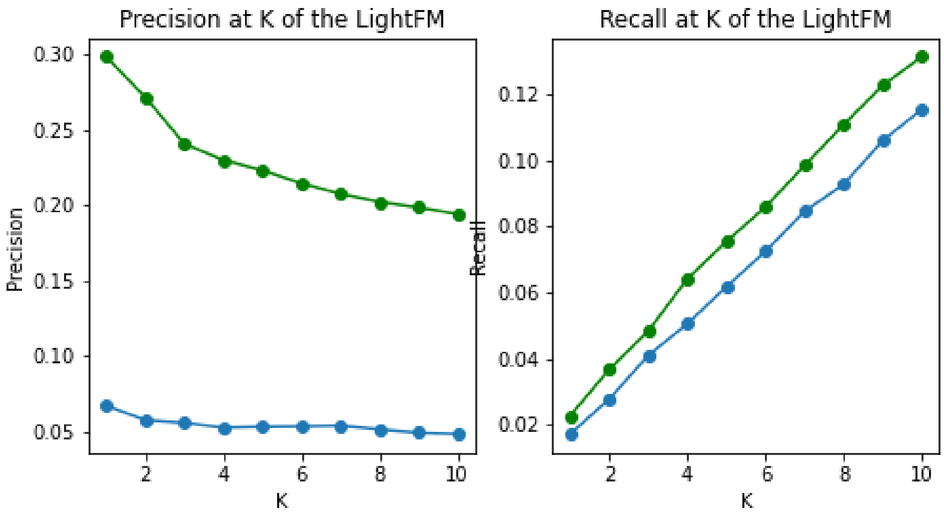

Fig. 11. Metrics of my LightFM on train and test set. On the left subfigure: Precision at k. On the right subfigure: Recall at k. K is from one up to ten.

```
User 4
        Known positives:
                Star Wars (1977)
                Contact (1997)
                Liar Liar (1997)
                Air Force One (1997)
                In & Out (1997)
                Ulee's Gold (1997)
                Lost Highway (1997)
                Cop Land (1997)
                Desperate Measures (1998)
                Wedding Singer, The (1998)
        Recommended:
                Star Wars (1977)
                Raiders of the Lost Ark (1981)
                Usual Suspects, The (1995)
                Godfather, The (1972)
                Return of the Jedi (1983)
                Terminator 2: Judgment Day (1991)
                Fargo (1996)
                Pulp Fiction (1994)
                Empire Strikes Back, The (1980)
                Contact (1997)
```

Fig. 12. Sample prediction of my LightFM

[2] A. CHAUHAN. (2021) Eda & recommendation model on movielens-100k. [Online]. Available: https://www.kaggle.com/code/amar09/eda-recommendation-model-on-movielens-100k

[3] TRISHNAS. (2020) Movielens-100k exploratory data analysis. [Online]. Available: https://www.kaggle.com/code/trishna8/movielens-100k-exploratory-data-analysis/notebook

[4] M. Kula, "Metadata embeddings for user and item cold-start recommendations," in *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, ser. CEUR Workshop Proceedings, T. Bogers and M. Koolen, Eds., vol. 1448. CEUR-WS.org, 2015, pp. 14–21. [Online]. Available: http://ceur-ws.org/Vol-1448/paper4.pdf

[5] Li, Fei-Fei & Li, Yunzhu, and Gao, Ruohan. (2023) Cs231n convolutional neural networks for visual recognition. [Online]. Available: https://cs231n.github.io/optimization-1/