

[F23] PMLDL | Assignment 2 Report

Introduction



At first, I have started to observe the given data. There were lots of variables like movie genres, user data, ratings, timestamps, and etc. I thought, do we really need to train the model on all of the data? Maybe there is another much simpler and more accurate way to do it.

As a main reference for my solution I will be using paper by **Han et. al.** [1] and Pytorch code [2], where they have implemented **Global and Local Kernels for Recommender Systems**. I am using this approach because their solution is lightweight, the model is very simple, does not have many layers, and gives good predictions. I have added many comments and edited bunch of code in their solution, so it would be more understandable for me and others.

Secondly, the model does not include side information such as user's gender, age, and etc. It only takes the ratings for movies of user.

Lastly, I have wrote my model which does not predict ratings for movies, but instead outputs top K movies based on the given ratings.

Data analysis



For the data analysis part, it was a simple task because PMLDL team provided information on the MovieLens dataset in the task description. Also, I did my personal data exploration like observing structure of each .base and .test alongside other csv-style files.

From the code by Han *et. al.*, they use two matrices constructed from **u1.base** and **u1.test** files:

- 1) 2D Matrix of 1682 movies each one rated by 943 users
- 2) Masks for the previous matrix because not all movies were rated by all users, so we have zeros in many rows.

Overall, they have 80000 + 20000 ratings for validation.

For my model, I use **ua.base** and **ua.test** for training, **ub.base** and **ub.test** for validation.

- 1) I got rid of masks for ratings because I will not predict the ratings, thus, I don't need them.
- 2) I have added to **y_train** and **y_test** only those movies which have rating > 3. Thus, Model will learn to predict movies, which the user will definitely like.

Model Implementation



Han *et. al.* consider matrix completion problem in their paper, i.e. they map higher dimensional space to low dimensions.

They Idea is to use 3x3 sized kernel on the input to achieve the best performance. They imply that *“focusing on more local features with smaller kernel size might be more effective for extracting generalizable patterns over the whole data matrix.”*

My model is an MLP with the softmax layer. I have tried ReLU and TanH activations, where first shown the best performance.

Model Advantages and Disadvantages

Han *et. al.* model

Advantages

- Lightweight.
- Does not use side information.
- Can produce quality recommendations based on small amount of inputs from user.

Disadvantages

- Original paper was implemented in **Tensorflow** and **Keras** and has shown better performance than solution in **Pytorch** that I use.

My model


Advantages

- Lightweight.
- Does not use side information.

Disadvantages

- Relies on too many input information, thus cannot produce good recommendation with low number of ratings.


Training Process



Local kernel model was trained on 500 epochs each time passing to it single matrix of shape (1682, 943) using early stopping with 5 consecutive rounds. The global kernel was trained on 1000 epochs with the same early stopping technique.

My model was trained on 1000 epochs with early stopping. The difference in training is that each time I am passing to a model (1, 1682) vector, which is ratings of movies of single user, and train model on them. After all 943 users' ratings', I take the average cross entropy loss and backpropagate it to model.

Evaluation



Inputs to both model are user's movie ratings. The more the better.

Han *et. al.* use **NDCG** and **RMSE** metrics to measure the predicted ratings. Both NDCG and RMSE were less than 1. which is a good result in general for predicting ratings. When the user passes ratings, model outputs almost the ratings for them, which is a sign that it understands the input-output relation. However, on inference, I exclude movies that were already watched by user to not recommend the same movie again.

As for my model, I use only **Cross Entropy Loss** because my task is to predict indices of movies, rather than their ratings. During validations, the best score was 42,67... which is not a promising result. In the next section, I provide example of input and output for both models.

Results

Han *et. al.* model

```
Input two integeres: index of movie and it's rating (1 to 5) separated by a single space.
To stop writing, press ENTER.
0 5
1 5
2 5
3 5
4 1

1. Twelve Monkeys (1995)
2. Babe (1995)
3. Dead Man Walking (1995)
4. Richard III (1995)
5. Seven (Se7en) (1995)
6. Usual Suspects, The (1995)
7. Mighty Aphrodite (1995)
8. Postino, Il (1994)
9. Mr. Holland's Opus (1995)
10. French Twist (Gazon maudit) (1995)
```

My model

```
Input two integeres: index of movie and it's rating (1 to 5) separated by a single space.
To stop writing, press ENTER.
0 5
1 5
2 5
3 5
4 1

1. Liar Liar (1997)
2. Star Wars (1977)
3. L.A. Confidential (1997)
4. Titanic (1997)
5. Saint, The (1997)
6. Schindler's List (1993)
7. English Patient, The (1996)
8. Twelve Monkeys (1995)
9. Godfather, The (1972)
10. Clear and Present Danger (1994)
```

Han *et. al.* model

```
Input two integeres: index of movie and it's rating (1 to 5) separated by a single space.
To stop writing, press ENTER.
0 5
1 4
2 5
3 4
4 1
5 1
6 5
7 5
8 5
9 4
10 3
11 3
12 4
13 5
14 5
15 5
16 1
17 1
18 5
19 1
20 2

1. Taxi Driver (1976)
2. Rumble in the Bronx (1995)
3. Birdcage, The (1996)
4. Brothers McMullen, The (1995)
5. Bad Boys (1995)
6. Apollo 13 (1995)
7. Batman Forever (1995)
8. Belle de jour (1967)
9. Crimson Tide (1995)
10. Crumb (1994)
```

My model

```
Input two integeres: index of movie and it's rating (1 to 5) separated by a single space.
To stop writing, press ENTER.
0 5
1 4
2 5
3 4
4 1
5 1
6 5
7 5
8 5
9 4
10 3
11 3
12 4
13 5
14 5
15 5
16 1
17 1
18 5
19 1
20 2

1. Star Wars (1977)
2. Liar Liar (1997)
3. Titanic (1997)
4. L.A. Confidential (1997)
5. English Patient, The (1996)
6. Saint, The (1997)
7. Schindler's List (1993)
8. Twelve Monkeys (1995)
9. Jerry Maguire (1996)
10. Contact (1997)
```



In the first example, for the same 5 ratings, both models produce different results, except for the one movie - **Twelve Monkeys**, which is present in both lists.

As we see, classification model is very sensitive to input, so it requires quite a lot of ratings to make predictions, while *et. al.* is much more scalable and produces better recommendations even for low amount of data.

References

[1] <https://arxiv.org/pdf/2108.12184v1.pdf>

[2] <https://github.com/fleanend/TorchGlocalK>