Github link - https://github.com/drindustrial/Collaborative-Filtering

Collaborative-Filtering

1. Introduction

Recommender systems appeared on the Internet a long time ago, about 20 years ago. However, the real rise in this area happened about 5-10 years ago when the Netflix Prize competition took place. Netflix then did not rent digital copies, but sent out VHS tapes and DVDs. It was very important for them to improve the quality of the recommendations. The better Netflix recommends movies to its users, the more movies they rent. Accordingly, the company's profit grows as well. In 2006, they launched the Netflix Prize competition. They made available the collected data: about 100 million ratings on a five-point scale, indicating the IDs of the users who put them down. The participants of the competition had to predict as best as possible what rating a particular user would give to a certain film. The prediction quality was measured using the RMSE metric. Netflix already had an algorithm that predicted user ratings with a quality of 0.9514 based on the RMSE metric. The task was to improve the prediction by at least 10% - to 0.8563. The winner was promised a prize of $ 1,000,000. The competition lasted approximately three years. In the first year, the quality improved by 7%, then everything slowed down a little. However, at the end, two teams with a difference of 20 minutes sent their solutions, each of which passed the threshold of 10%. The late team (like many others who participated in the competition) were left with nothing, but the competition itself greatly spurred development in this area.

There are two main types of recommender systems. Content-based and Collaborative Filtering. In this work was used Collaborative Filtering.

2. Methodology

There are many methods for collaborative filtering; in this work was used Matrix Factorization and Deep Learning Based Method.

Matrix Factorization

Matrix factorization algorithms work by decomposing the matrix of user interaction with an element into the product of two lower-dimensional rectangular matrices.

Deep Learning Based Method

A typical architecture contains two embedding layers, a hidden line layer (s) and an output layer. These two embedding - user embedding and item embedding. These two embeddings are combined before going through a series of fully connected hidden layers and an output layer.

3. Data

Data represented by the table with three columns (user id, item id, user rate). For the matrix factorization this data transforms into the spars matrix n x m, where n is number of users and m is the number of items.

4. Matrix Factorization

For the result of this method we should obtain two lower-dimensional rectangular matrices $P \in R^{n\times k}$ and $Q \in R^{m\times k}$, where n is number of users and m is the number of items and k is latent dimension. Multiplying p_i by q_j we will obtain predicted rating for item j from the user i.

For obtaining P and Q was used gradiant decent algorithm to minimize this equation $(P \times Q^T - R)^2 + \alpha * \left(\|P\|_2 + \|Q\|_2\right) \rightarrow \min\limits_{P,Q}$

The result on test dataset after 250 iteration of optimizing algorithm is 0.766 MSE with 15 latent dimension.

5. Neural Collaborative Filtering (NCF)

Architecture of NN:

```
Embedding(6744, 20)
Embedding(118697, 20)
Linear(in_features=40, out_features=80, bias=True)
Linear(in_features=80, out_features=40, bias=True)
Linear(in_features=40, out_features=1, bias=True)
```

First embedding for users and second for items then outputs of those layers concatenate and passing through few fully connected layers with dropout and activation function Relu.

After 20 iterations of training this net was obtained 0.58 MSE on train dataset and 0.72 on test dataset.

6. Conclusion

From the results of each model, we can make these assumptions:

Sparse matrices works fast and consume a little memory for matrix factorization. Matrix factorization need less memory then and computes faster than Deep Learning Based Method but have bigger MSE loss. If we want to implement collaborative filtering and choose between those two methods, we should take into consideration our resources and that MSE loss is enough for our task, with bigger resources we can use Deep Learning Based Method and if we fine with lower MSE Matrix factorization method could be enough.