Air Liquide Project

Peiran Chen

March 2019

1 Problems

Develop user-based, item-based neighborhood methods and matrix factorization algorithms for collaborative filtering using one of the MovieLens data set. Benchmark the performance of each algorithm.

2 Data sets

MovieLens Latest Datasets Small: 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. Last updated 9/2018. Separate the dataset into training set and test set by 70%/30% rule.

3 Algorithms

3.1 User-based kNN algorithm

First read rating file from dataset. Adjust the rating by average rating of this user across all his rating on movies he has enjoyed. Then, calculate weighting matrix between two users. Each entry in weighting matrix in calculated as the cosine correlation between all the movies these two users all enjoyed and rated before. With weighting matrix measuring similarity between any two users, make prediction based on weighted-average of other users' mean-adjusted rating on this specific movie. In the test part, I apply very simple rule to quantify the performance of this algorithm. User's rating has been divided into positive and negative on threshold 2.5. I calculate 'Precision' and 'Recall' based on whether my algorithm could predict positive or negative correctly. I introduce the idea of k-nearest neighbor by assuming user only use at most k movies to calculate weighting matrix and k users to make prediction.

3.2 Item-based kNN algorithm

First read rating file from dataset. Adjust the rating by average rating of this movie across all its rating by users who has enjoyed it. Then, calculate weighting matrix between two movies. Each entry in weighting matrix in calculated as

the cosine correlation between all the rating pairs they received from same user. With weighting matrix measuring similarity between any two movies, make prediction based on weighted-average of other movies' mean-adjusted rating on this specific user. In the test part, I apply very simple rule to quantify the performance of this algorithm. User's rating has been divided into positive and negative on threshold 2.5. I calculate 'Precision' and 'Recall' based on whether my algorithm could predict positive or negative correctly. I introduce the idea of k-nearest neighbor by assuming user only use at most k movies to calculate weighting matrix and k users to make prediction.

3.3 Matrix Factorization algorithm

In this part, I use machine learning package from scikit-surprise. I choose SVD and Non-negative factorization algorithm.

4 Results

The result is calculated by personal desktop without parallel computing. The CPU is Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz. I only use Numba jit to accelerate the computing process.

4.1 User-based kNN algorithm

k	**Precision**	**Recall**	**Comp-cost/min**
2	87.51 %	95.71 %	4.03
5	87.68 %	93.43 %	8.24
10	100.00 %	100.00 %	20.95

4.2 Item-based kNN algorithm

k	**Precision**	**Recall**	**Comp-cost/min**
2	88.28 %	92.18 %	7.41
5	87.67 %	91.57 %	19.46
10	100.00 %	100.00 %	50.59

4.3 Matrix Factorization algorithm

RMSE of SVD is 0.8738. RMSE of Non-negative matrix factorization is 0.9200.

5 Conclusion

In this exercise, I choose three types of methods. Based on MovieLens Latest Datasets Small, algorithm based on matrix factorization is much faster than

kNN algorithm.

6 Next Steps

6.1 Accelerating computing

Evaluating weight matrix and predicting people's rating can be parallelized with MPI, OpenMP or CUDA.