RECOMMENDER SYSTEM

Atom Group: Jifu Zhao (jzhao59), Jinsheng Wang (jwang278)

Nuclear, Plasma, and Radiological Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801, USA

1. INTRODUCTION

In this project, our goal is to build a recommender system based on the MovieLens 1M Dataset. MovieLens 1M dataset contains 1,000,209 anonymous rating of approximately 3,900 movies made my 6,040 MovieLens users who joined MovieLens in 2000. These movies can be categorized into 18 different categories. In this project, based on some well developed Python package, we compared the performance of different algorithms and finally choose 2 models as our final models. More details will be described in the following sections.

2. PRE-PROCESSING

The dataset has three files: movies.dat, ratings.data and users.dat. Among these three files, the most useful is ratings.data. Since the algorithm is pretty straight forward, there is little pre-processing steps. In this project, we compared the implementation and performance of Python and R, finally we decide to use Python. The main package we used is called surprise, which is is Python scikit package for building and analyzing recommerder systems.

3. METHODS

Surprise has many algorithms, such as baseline algorithms, neighbourhood methods, matrix factorization-based (SVD, PMF, SVD++, NMF) and so on. In this project we first studied the performance of 10 different algorithms (NormalPredictor, BaselineOnly, KNNBasic, KNNWithMeans, KNNBaseline, SVD, SVDpp, NMF, SlopeOne, CoClustering). Through 5 folder cross validation, we finally choose two best performed algorithm: KNN with Means and SVD algorithms.

3.1. KNN with Means Algorithm

KNN with Means algorithm is a basic collaborative filtering algorithm. It takes into account the mean ratings of each user. Two important factors for this algorithm is whether or not it is user-based and the measurement of the similarity. Using cross validation, we choose the mean squared difference (msd) similarity and use item-based mode.

3.2. SVD Algorithm

The SVD algorithm is popularized by Simon Funk during the Netflix Prize. In Surprise, there are in fact two algorithms based on SVD: SVD and SVD++. Although SVD++ performs a little bit better than naive SVD algorithm, it takes too much time to run the code. So, in this project, we only implement the SVD algorithm.

4. CODE DESCRIPTION

All of our code is contained in the file named mymain.py. It will first read the training and test data. After build the KNN and SVD model, it will

make predictions on the test dataset and save the prediction into local csv files.

The code runs very fast. As tested, the total running time is around 3 minutes.

5. RESULTS

In this project, we use the Root-mean-square-error (RMSE) as the metric to measure the performance of different algorithms. A summary of the result of 5 folder cross validation is shown in Table 1.

Table 1. Summary of Cross Validation

RMSE/Model	KNNWithMeans	SVD
Fold 1	0.8852	0.8617
Fold 2	0.8862	0.8627
Fold 3	0.8873	0.8627
Fold 4	0.8866	0.8625
Fold 5	0.8830	0.8603
Average	0.8857	0.8620

Note: in our cross-validation, the data is randomly split into 5 folders. But for actual test, there will be some users or movies shown only in test dataset. In that case, the performance is expected to be a little bit worse.

The above code is tested on our laptop, the average running time is about 3 minutes. The corresponding computer system is: MacBook Pro, 2.6GHz Intel Core i7, 16 GB memory.

Acknowledgement

The authors would like to thank Xichen Huang for his tutorial notebook on Piazza.