UNSW SYDNEY

COMP9417

Machine Learning and Data Mining

Assignment

Topic: 3.4

Movie recommender system using collaborative filtering

Joel Lawrence (z3331029)
Deepansh Deepansh (z5199370)

August 2019

Introduction

The goal of this project is to build a recommender system that answers 2 questions:

1. Whether a user will enjoy a movie or not?
2. Whether there are similar movies to the one selected by the user or are there users with similar taste?

To do this, we examine the MovieLens Dataset which contains 100,000 movie ratings from users across a giant sample of movies. Various methods were tested to determine an overall "best" approach. An examination of these method was carried out and models were fine-tuned according to these methods. Errors were compared according to our predicted result vs the test set. For research purposes, various classifier was used to predict ratings of movies by a particular user on a particular set of features. These classifiers were then compared on the basis of their error rates with and without k-fold validation.

Importantly, a predicted rating should aim to not only be accurate, but also be able to effectively determine enjoyment or not given the incredibly varied choices and tastes of people. The project will work through each method and arrive at the Collaborative Filtering approach for recommender systems. The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person.

From a machine learning perspective, the idea is to find the most similar (cosine similarity and correlation similarity method) users to your target user (Nearest Neighbours approach) and weight their ratings of an item as the prediction of the rating of this item for target user. There are quite a few limitations of this method. It doesn't handle sparsity well when no one in the neighbourhood rated an item that is what you are trying to predict for target user. Also, it's not computational efficient as the growth of the number of users and products.

The repository for this project can be found here:
https://github.com/joel295/movie_recommender

Experimentation and Implementation

Explanation of proj.py script:

Proj.py was written mainly for exploration and experimentation of the dataset which provides some useful insights and results that are used in the final outputs of the project.

1) Data Processing:

- Once the MovieLens data was acquired, it was cleaned and analysed for prediction purposes.
- Of the three files (Movies, Ratings and Tags), Ratings and Tag files were combined on user_id to get a compacted table.
- For all NaN values, -1 was filled in. From this timestamp was dropped as it was not useful for our classification methods.
- Using this data, a rating matrix was also created with rows as userId and columns as MovieId. Shape of the matrix is userId x movieId. For all NaN values, 0 was filled in place.
- Rating_matrix shape = 610 x 193609

2) Classification using Classifiers:

- 5 classifiers were used, namely: KNN, Logistic Regression, Random Forest, Neural Nets, SVC
- These classifiers were applied once to the already curated data with just training and testing set.
- Then these classifiers were fed in a pipeline which standardised the data and applied models on top of it.
- This was not very intuitive hence data standardisation was not done. Through this pipeline, K-fold validation was applied on the data and classifiers were used to find accuracy score on each fold and the mean accuracy score.
- Through rigorous tuning and experimentation, it was found that KNN gives the best accuracy of all the classifiers but even that was not substantial as number of features were far too less to predict rating on a movie by a user.

The results are posted below for each classifier:

1. kNN :- approx. 50% accuracy. The best of all the models. Initial accuracy increased by 10% upon using 4-fold validation.
2. Random Forest:- Overfitting the data
3. Neural Nets:- Very less number of features for correct prediction (solver = [‘adam’,’lbfgs’], hidden layers = [10,5])
4. Logistic Regression:- Less number of meaningful features result in low accuracy (<30%)
5. SVC :- Doesn’t work, Low accuracy (<30%)
6. Gaussian Naïve Bayes :- very low accuracy
7. Stochastic Gradient Descent:- low accuracy
8. Decision Tree: -100%, overfitting the dataset

3) Collaborative Filtering:

- For the recommendation systems, Memory based Collaborative filtering was used.
- The rating matrix that we constructed during data processing stage is used for user-based and item-based similarity via cosine similarity and correlation similarity
- Weighted and mean ratings were used to predict the rating of a particular movie on the basis of similarity between users and items.
- Through this filtering process, whenever a user id and movie ID is given, the system finds out the k most similar users and k most similar movies on the basis of cosine similarity.
- Mainly cosine similarity is used because the rating matrix is very sparse.
- The evaluation of predicted ratings is done on the basis of RMSE (root mean squared error)
- In applications where user-base is large, user-based approaches face scalability issues, as their complexity grows linearly with number of users. Item-based approaches address these scalability concerns to recommend items based on item similarities.

Classifier Results

Results of Classifiers with and without K-fold Validation. As the data set was 100k, 4-fold validation was used.

Classifier	Accuracy without K-fold validation	Accuracy with K-fold validation
K-Nearest Neighbour	39.70%	49.4% +/- 0.2%
Multi-Layer Perceptron	26.46%	26.5% +/- 0.2%
Logistic Regression	26.47%	26.5% +/- 0.2%
Random Forest	100.0%	100.0%
Gaussian NB	25.73%	25.8 +/- 0.2%
Decision Tree	100.0%	100.0%
SGD	20.17%	14.7% +/- 6.2%

Table 1: Summary of Classifier Results

Collaborative Filtering continued…

Initially, the focus was on implementing Collaborative Filtering on the overall set to determine how it performed as an approach compared to our initial experimentation methods. It performed well in comparison. It had met our initial hypothesis of predicting results however we decided that it could still be improved.

The idea here was that outliers and people who were vastly different from the target user could be adversely affecting the target user's prediction, even taking into account similarity weightings. To test this idea, we modified our Collaborative Filtering approach to only consider the Top-K most similar users when predicting a movie rating. The K value can represent any number less than the total number of users in the dataset.

As can be seen in the appendix in *Figure 1* below, the result drastically improved from the overall when taking the Top-25 and Top-50 users for comparison, *Figure 2* shows that picking top-30 users for comparison returned the most accurate predictions.

Results for Memory-based CF using user 1 and movie 1:

```
Enter the user id: 1

Enter the movie_id: 1

Enter the metric - cosine or correlation: cosine


USER BASED SIMILARITY

 4 most similar users for User  1:

 1: User 266, with similarity of 0.35741
 2: User 313, with similarity of 0.35156
 3: User 368, with similarity of 0.34513
 4: User 57, with similarity of 0.34503

Predicted rating for user  1 -> movie  1:  2
ITEM BASED SIMILARITY

 4 most similar movies for Movie Toy Story (1995):

 1: Toy Story 2 (1999), with similarity of 0.57260
 2: Jurassic Park (1993), with similarity of 0.56564
 3: Independence Day (a.k.a. ID4) (1996), with similarity of 0.56426
 4: Star Wars: Episode IV - A New Hope (1977), with similarity of 0.55739

Predicted rating for user  1 -> movie  1:  3
```

MSE comparison between recommender and Top-K values

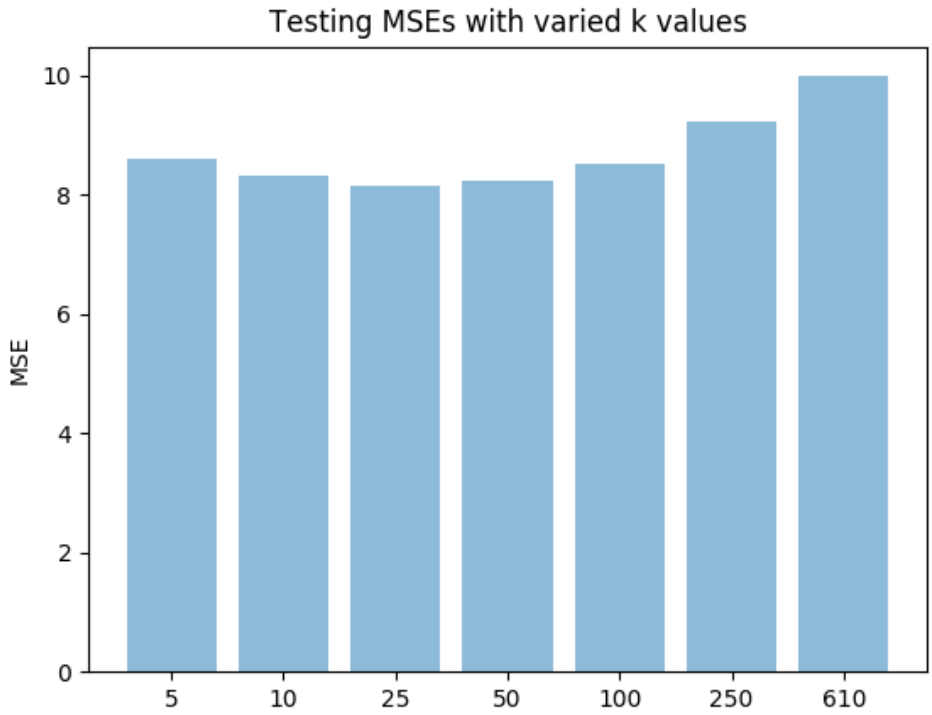

Figure 1: Top-K wide spread results

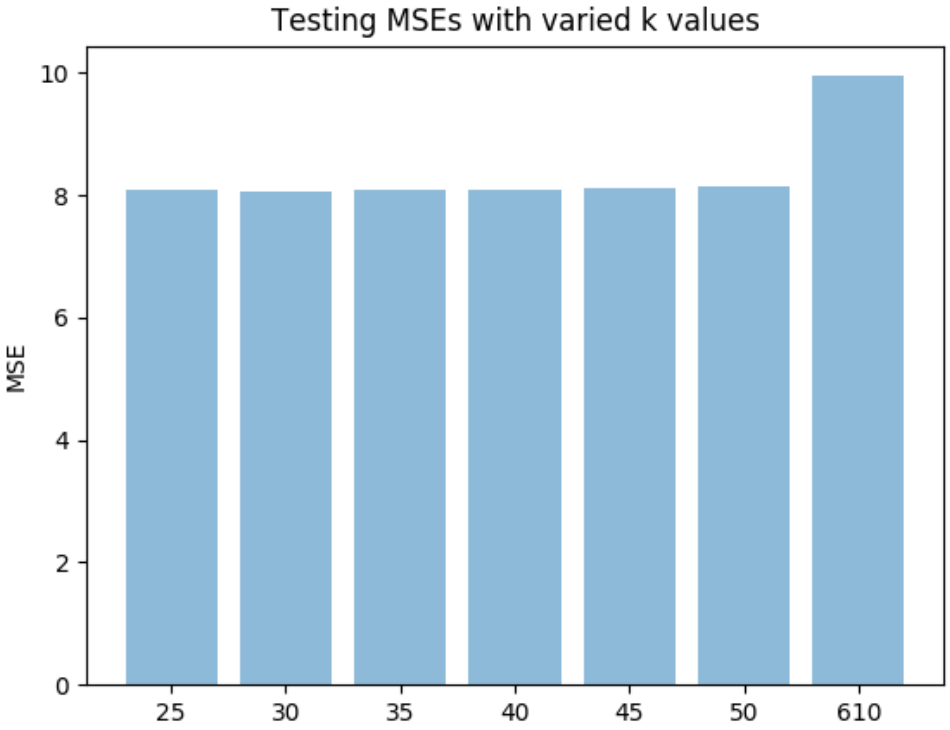

Figure 2: Top-k narrowed down results

References

- For Collaborative Filtering, the cosine similarity method below was utilised.

> Cosine similarity, or the cosine kernel, computes similarity as the normalized dot product of X and Y:
>
> K(X, Y) = <X, Y> / (||X||*||Y||)

- https://towardsdatascience.com/collaborative-filtering-based-recommendation-systems-exemplified-ecbffe1c20b1

- Python Machine Learning - Sebastian Rachka