Yelp Dataset Final Report

Authors: Ying Du, Jiaqi Tang, Chengyou Ju, Chenlu Jia

Part 0. Baseline Model

Before the first part of our prediction process, we create a baseline model as our baseline accuracy test. We use the DummyClassfier from the sklearn package. Since there are only 5 ratings from 1 to 5, we treat this problem as a classifier problem, and use this classifier model to predict the latest rating of each user. We use the “most frequent method” as our prediction method, which means we predict each unknown rating by the most frequent rating in the training set. Then we test the accuracy on the test set, and the accuracy score for this classifier model is about 0.5, which is close to the score of a random prediction model. After setting the baseline accuracy, we continue on predicting by other models.

Part 1. Rating Prediction using SVD

In this part, we use the SVD model for the rating predictions. We will use the ratings dataset only to predict the last rating of each user.

First, we get our ratings data frame from the review dataset. Each row in our data frame consists of the ID of this user, the ID of the merchant, the ratings given by the user on this merchant, and the date and time the user made the rating.

Then, we treat all users who have made at least five ratings as active users and select all these users and put in a new data frame. Based on this new data frame, we divide our data into a training set and a testing set, where the training set consists of ratings of each user except the latest one, while the test set contains the last rating of each user.

After getting our data frame ready, we first try the DummyClassifier, and receive a mean accuracy score of around 0.5. We treat this as one of our baselines, and we want to see later if we can beat this score with our SVD model.

After using cross validation on the training set, we can see that the FCP and MAE scores indicate that the model fits pretty well on the training set and we can proceed with our experiments.

First, we train on the whole active users data frame, and then we try to predict the last ratings of the first 20 users in our data frame. After comparing pairs of predicted rating and the actual rating, we can see that most of them are very close to each other, which makes us believe that our SVD model works well.

After this, we also test on the first 10K, the last 10K, and the second last 10K active users. From our results and the graphs we draw based on the test scores, we can see that there is almost no difference among these results. Therefore, we tend to believe that when trained on the whole active users dataset, the accuracy of predictions is consistent across all users.

Second, we try to train and test our data on users that are not very prolific on ratings, and we can see clear differences right now. As the users become more prolific and write more reviews, we can get better test scores. In other words, we should train on users with more ratings to make better predictions, which makes sense.

Last but not least, we are also interested in whether the number of ratings received by merchants will also affect our predictions. So similar as our previous tests on users, we try on businesses with more than 100, 200, and 500 reviews, and we can also get similar conclusions. When training on more popular businesses, we can get better results in prediction.

Therefore, we can conclude that in order to make better predictions, we should train on a dataset in which users are prolific and merchants are popular. Our results match our common sense, as we can make accurate predictions only if we have enough information.

Detailed results with numbers and graphs can be found in our ipynb or pdf files.

Part 2. Rating Prediction using review text

We use the user reviews only for prediction of the rating. First we convert text data into TF-IDF vectors and then classify the text data using a LinearSVM.

Sometimes, the reviews will contain phrases like "not good", so we cannot only consider words independently. When we break a text into n-grams, we consider several words grouped together to be a single word. “The food was not great” would be represented using bi-grams as (the food, food was, was not, not great), and this would allow our system to learn that not great is a typically negative statement because it appears in many negative reviews.

We will use unigrams (single words) and bigrams (two words at a time) since longer combinations of words will lead to great memory cost.

First we use all the review data from the original dataset to turn all of our reviews into vectors. The accuracy score was 0.7388 and the classification report indicates that the model is biased to different ratings. Particularly for ratings 2,3,4 , their accuracy is much lower than 1 and 5. We think about improving the model on balanced dataset.

After an analysis of the counts between different ratings, we can see that there are more examples of texts that typically have a 5-star rating than texts that typically have a 2-star rating. Because of the probabilistic models at the base of most machine learning classifiers, we'll get less biased predictions if we train the system on balanced data. This means that ideally we should have the same number of examples of each review type. So we turned all the active users' reviews into vectors and train a balanced dataset. The accuracy score was 0.6918 and the classification report still indicates great bias between the precision of different ratings. A balanced train dataset does not lead to higher accuracy than the previous dataset. This may result from the decrease in the size of vectors transformed from text data which contains less information. Thus we transformed all the text reviews into vectors. However, accuracy score was 0.69199 and the classification report was still greatly biased. Maybe balanced text will not lead to a more accurate prediction.

Next we use data from active users only and predict their ratings. Since in all the above step we turned all the reviews from the dataset to vectors and that may be not accurate since we did not separate active users from inactive users. The accuracy score was 0.7328 which is almost the same as using the whole dataset so the review of the inactive users do not make much difference to the active users' review.

After this we used text from inactive users and achieved an accuracy score of 0.7801. It is surprising that prediction for the reviews of inactive users is much higher than those of the active users. This may due to larger size of dataset that makes it more accurate.

Last but not least, we are interested in the average review length for different differences between actual and predicted ratings. We can see that when the length of the review is relatively short, it is mostly likely that the prediction will get completely wrong. Maybe this happens because there is relatively less information in the short reviews and is difficult to decide the sentiment in it. Sometimes less information may lead to incorrect prediction.

In conclusion, we should just use all the text data and turn them into vectors to train active users' rating result.

Detailed results with numbers and graphs can be found in our ipynb or pdf files. (review_rating_prediction.ipynb)

Part 3. Rating Prediction Using Collective Matrix Factorization

The collective matrix factorization model is an extension of the typical low-rank matrix factorization model to incorporate user and/or item side information. In its most basic form, low-rank matrix factorization tries to find an approximation of a matrix X given by two lower-rank matrices A and B, which in recommender systems would represent, respectively, a matrix of latent factors for users and items, which are determined by minimizing the squared differences between their product and X, i.e.:

$$argmin_{A,B}||X - AB^T||$$

In this section, we compared the result of basic model, model with user side information and model with item side information.

First, like what we did in other sections, we loaded the rating data and split it into train set and test set. Second, we loaded user side information with "user.json" in the yelp_dataset. Although this data set contains many features for a single user, we chose the following features for further analysis: UserId, review_count, fans, average_stars, useful, funny and cool, because they are easy to access and closely related to rating. Next, we loaded item side information with "business.json" and particularly focused on the categories of each business. We transformed categories of each business into binary tags with one hot encoding. Eventually we obtained a dataframe with ItemId (business id), and tags. However, this dataframe could be very sparse due to the large number of tags and limited tags for a single business.

We generated the prediction with "cmfrec" package (Documentation is available at readthedocs: http://cmfrec.readthedocs.io/en/latest/). First, we used only the basic train data to predict latest rating, and calculated rmse = 1.10. Second, we generated a model with user side information, and calculated rmse over 6. Last, with item side information, the model generated rmse = 1.4586. Note that the model will apply a sigmoid function to the factorized approximations of these binary tag columns.

Surprisingly, the CMF model did not generate a more accurate prediction than the basic model. A possible reason is that the selected feature in the side information are not very useful for prediction. Due to the lack of data overview, it was hard to interpret each feature in the dataset, so we might miss those can truly reflect ratings.

Detailed results with numbers can be found in our ipynb file. (CMF-final.ipynb)

Part 4. Conclusion

We tried several methods for the prediction. First we used the DummyClassifier as the average baseline model and the result was not satisfying. Next we tried the SVD model as the memory-based methods and tested on different groups of users-prolific users and popular merchants lead to a relatively high accuracy. We also tried content based model to predict ratings from text reviews only. But the limitation is that we have to have reviews first. Finally we tried CMF model and found that side information was not useful for the prediction.

Note that we have used different types of models, including both classifier and non-classifiers. The classifier model predicts the ratings and gives an integer rating prediction from 1 to 5. The non-classifier models predict the ratings and could give non-integer ratings. For evaluating the classifier model, we use the accuracy score as our evaluation metric, since we can test on each prediction and see whether it’s correct or not. For evaluating the non-classifier models, we use the RMSE and MAE scores as our evaluation metrics, since we will not necessarily get an integer prediction score, we cannot compare our prediction score directly with the real score and have to use the differences between them to see whether our predictions are good or not.