1 User-Based Collaborative Filtering

1.1 Basic user-based collaborative filtering algorithms:

Cosine similarity MAE: 0.79035

Pearson Correlation MAE: 0.78730

1.2 Extension to basic user-based collaborative filtering (Pearson Correlation):

1). inverse user frequency MAE: 0.76987

2). case modification MAE: 0.76974 (only case modification), 0.76703(IUF + case modification)

2 Item-based Collaborative Filtering

Item-based collaborative filtering based on adjusted cosine similarity MAE: 0.83984

3 My Own Algorithm – User-based Collaborative Filtering with movie controversy

Movie controversy is an important factor (and more important than IUF) in comparing user similarity. If two users have the similar rating for a very controversial movie, we have strong confidence to say that the two users are similar. Since movie controversy can be measured by standard deviation, my algorithm uses movie rating standard deviation as weight when calculating the user similarity. Here's what I did:

For each movie in the training dataset, calculate their rating standard deviation and store in a dictionary with key as movie ID and value as the movie standard deviation. When calculating user similarity (here I use Pearson Correlation), each movie rating will be multiplied by **log10(1 + movie standard deviation).** The purpose I take a log of standard deviation here is for smoothing (because if one movie's standard deviation is two times of another's we can't say the former is 2 times important than the latter). Besides 10, I've also tried other base numbers for the log such as 2, e, and base number as 10 performs best. Also, the reason to add 1 to the standard deviation is to deal with the situation that the standard deviation of rating in some movies is 0 since we can't calculate log10(0), and add 1 here is reasonable since if all users give the same rating for a movie, that means this movie is not important at all when calculating user similarity and log10(1 + 0) will be 0, which means we give 0 weight to the movie when calculating user similarity (similar to IUF we give 0 weight to movie rated by all users). After getting the user similarity with movie

controversy as weight, I select top K similar users and apply case modification (same as 1.2) in rating prediction to predict movie rating. The MAE for my algorithm is 0.76634.

By combining the result rating from IUF_Case_Modification with the result of my algorithm to get the average rating, that's to say, rating = 0.5 * IUF_Case_Modification rating + 0.5 * std_case_modification rating, I got better MAE: 0.76568. Then I implement the weighted slope one algorithm as mentioned in the paper (https://arxiv.org/abs/cs/0702144) and again, take the average of the result rating from it and my algorithm, got the MAE: 0.75460. By combining results of cosine similarity (with IUF and case modification), Slope One and my algorithm, got a better MAE: 0.74372

4. Results Discussion

The following is a table summarizing the overall MAE for different algorithms I implemented:

Algorithms	**MAE**
Basic user-based collaborating filtering (cosine similarity)	0.79035
Basic user-based collaborating filtering (Pearson Correlation)	0.78730
Modified user-based collaborating filtering-Inverse User Frequency (Pearson Correlation)	0.76987
Modified user-based collaborating filtering-Case Modification only (Pearson Correlation)	0.76974
Modified user-based collaborating filtering-IUF + Case Modification (Pearson Correlation)	0.76703
Item-based collaborating filtering (adjusted cosine similarity)	0.82984
Modified user-based collaborating filtering-Standard Deviation + Case Modification (Pearson Correlation) --My own algorithm	0.76634

Combine my own algorithm (std_Case Modification with IUF_Case Modification)	0.76568
The weighted Slope One algorithm (sourced from the paper professor mentioned)	0.75938
Combine weighted Slope One with my own algorithm(std_iuf_case)	0.75460
Combine Cosine similarity, Slope One, and my own algorithm (std_iuf_case)	0.74372

From above, we can see in basic user-based collaborating filtering, Pearson Correlation performs better than cosine similarity, this is reasonable since basic cosine similarity doesn't consider user difference in rating, some users tend to give lower rating for all movies while some users tend to be generous, and Pearson correlation offsets this drawbacks by subtracting each user's average rating from the original rating to reflect user's rating preference, which is more reliable when calculating the similarity. Though basic Pearson correlation considers user preference, it doesn't consider the movie importance in calculating similarity, and IUF solved this since less common movies are more useful and important than universally rated movies when calculating similarity, so we can see the MAE for IUF with Pearson correlation has greatly improved than the basic Pearson correlation. The case modification on Pearson correlation emphasizes high weight and punish low weight on similarity as the higher the similarity of the user, the more important the user is, so we can see from the above table that case modification also has a lower MAE than the basic Pearson correlation. Combining IUF with case modification, the MAE improved than just using one of them since it optimizes both the similarity calculating process and rating prediction process.

The logic of my algorithm is very similar to combination IUF with case modification above, except that in my algorithm, I use standard deviation as weight instead of IUF when calculating user similarity, that's because controversial movie is more important than the less common movie, which can more reflect user similarity. And the MAE of my algorithm proves this is true since it has lower MAE (0.76634) than the combination IUF with case modification algorithm (0.76703). By combining the result of my own algorithm with the result of previous IUF_case_modifiction algorithm (get the average rating of the two algorithms), the result improved a little bit (0.76568).

By implementing the weighted Slope One algorithm from the paper, the MAE improved a lot (0.75938), which I think the reason is as what the paper mentioned, Slope One considers both users who rated the same item and items rated by the same user, that's to say, it considers the feature from both item-based collaborating filtering and user-based collaborating filtering. Based on the advantage of ensemble learning, combining rating result of weighted Slope One with the result of my algorithm to get the average rating, the MAE improved again (0.75460). Combining results of weighted Slope One, my algorithm and cosine similarity, got a better MAE: 0.74372. It seems like the more the number of the algorithm is, the better the ensemble learning result is. This is reasonable since each algorithm might perform well on some data and less accurately on others. When combining all of them, they cancel out each other's weaknesses.

Thus, for the algorithms discussed above, I think my results are reasonable. But one thing surprises me is that for item-based collaborating filtering based on adjusted cosine similarity, the MAE I got is the highest among all of the algorithms I implemented. I originally expect it to be better than used-based collaborating filtering (at least the basic one) since item similarity is usually more stable than user similarity as a user may have multiple interests, but the MAE I got seems not very well (though it is still in acceptable range), the reason might be that the dataset size is not large enough to prove the advantage of item-based collaborating filtering algorithm. Since in small dataset, the algorithms may easily learn the patterns that do not exist, which results in high variance and error on predicting test dataset and caused the overfitting.

Another thing I want to discuss is that the MAE result in each of my submission has the same trend that the value of MAE is: result_5 > result_10 > result_20 (the following is a screenshot for one of my submission as an example), I think this is also reasonable since in test5, each user only provide 5 movie ratings, and it is possible that all of these 5 movies are not important at all in calculating similarity (for example, all of these 5 movies are commonly rated movie), but we have no choice and can only use these 5 movie ratings. In test10 and test20, each user provides more movie ratings so the probability of all movies provided are not important is decreased.

MAE of GIVEN 5 : 0.816556208578217
MAE of GIVEN 10 : 0.754166666666667
MAE of GIVEN 20 : 0.734638757596219
OVERALL MAE : 0.766335577080939

Screenshot of MAE result for one submission

Overall, I think the result of my implementation of the algorithm is reasonable.

5. Instructions to run my code

Please check the README.md file in code folder