TP1: Collaborative Movie Recommendation

Waner de Oliveira Miranda - 2010054916
waner@dcc.ufmg.br

Abstract: *This report is a collective of decisions made to accomplish a task given during the classes of the discipline Recommender Systems. The task was to implement a collaborative filtering algorithm to make movie recommendations based on past ratings.*

1.Introduction

Recommender Systems are very widely used today for movies, products and many others subjects. A class for this discipline is administered to the Computer Science Course at the Universidade Federal de Minas Gerais (UFMG). During the classes the professor made an assignment, in form of a practical work to be delivered within a month. This is work is to build a recommender system based on collaborative filtering models to predict ratings for movies based on past ratings. Two files were supplied, one containing the past ratings for users and another with users and items to have its ratings predicted.

2.Implementation

For this work we implemented the algorithms in CPP language and distributed the main procedures in classical cpp and h files. The main.cpp which contains the flow of methods and main structures such as maps to users and items and vectors for the stats (average, count) for item and users too. We made some decisions trying to find a point where the time and memory costs could be reasonable, for that we decided to use a map structures to hold the item and user values along with its positions in the array of average values or feature vectors, in the case of item-based or user-based methods. To keep the votes as feature vectors we opted to use simple multidimensional arrays which are less costly to maintain in memory and fast to threat in loops.

The prediction methods for the collaborative filtering were spreaded with the Predictor.cpp and Predictor.h files. In these files we will find the methods for prediction we tried during the competition:

```
void avg_predictions(unordered_map<string, size_t> &users,
unordered_map<string, size_t> &items, const vector<vector<float>>
&users_stats, const vector<vector<float>> &items_stats, const
vector<vector<string>> &targets, vector<float> &predictions,
vector<float> &missing_predictions);
```

With a O(N) complexity where N is equal the number of predictions to be made, this method is used to find the user or item in the structures and set to the pair user:item a value of rating based on user or item average ratings.

```
void avg_predictions_personalized(unordered_map<string, size_t> &users,
unordered_map<string, size_t> &items,
const vector<vector<float>> &users_stats, const vector<vector<float>>
&items_stats, const vector<vector<string>> &targets, vector<float>
&predictions, vector<float> &missing_predictions);
```

With a O(N) complexity where N is equal the number of predictions to be made, this method is used to find the user or item in the structures and set to the pair user:item a value of rating based on user or item average ratings, but when a item average is assigned it was supposed fix it using the user bias. (Abandoned during the evaluations).

```
vector<vector<pair<size_t, float>>> rank_vectors(float
**feature_vectors, vector<size_t> targets,
size_t size_rows,size_t size_cols);
```

This method was made in attempt to calculate distances between the users or items, its complexity is given by O((T*M)*N), where M is equal to the number of rows of the matrix and N the number of columns and T the number of targets. We were still evaluating this implementation but during the tests it took around 30 minutes to execute for the number of targets, users and items supplied. (Abandoned during the evaluations).

```
void review_predictions(const vector<vector<string>> &targets,
vector<float> &predictions, vector<float> &missing_predictions);
```

During the predictions there were some items or users not listed among the past ratings, so their predictions were missing during the evaluation. At this point we started to implement a method that would check and average the predictions output for the know users and items and apply them to the missing ones. The complexity of this method is of O(T^2), where T stands for the number of targets.

We also let some of old implementations as proof that we were trying another methods and approaches during the time the practical assignment was given.

3.Submissions

The first submission made was to set what is a goal for the competition. Along with the datasets there was also a file with a random values for the prediction. This file was used as the first submission, which gave us a score of 4.10318 points. This value was considered a baseline for us to start the implementation.
The next submission to be considered was a simple average of votes by items, to predict the output value we setted the average vote for the target item in the past and output it as the result for the target and this submission earned 2.266925 score.
Seeing that the average of votes was an improvement we tried to mix the item vote average with the user average vote, and this method that scored 1.75709 points.
Our best submission happened when we applied a review to the given votes before print it out. This review was made by checking which pair of user/item that had no assigned value of prediction, and for them we made another average of votes within the already predicted values

for users and items. This method scored 1.70644 points and took only 30 seconds to be executed.
Others submissions, we trying a sort of methods, our best item-based collaborative filtering could achieve only 1.79472 points considering a thousand neighbors. The performance of the method was not good either, taking 30 minutes to be completed using 7 threads.

4.Conclusion

After three weeks straight doing implementations, our best implemented method was the simple mix of the average of the votes for users and items, it was not only the fastest (30 seconds) but it achieved a good result compared to the others in the competition (1.70644 points). We tried to implement a item-based filtering but, due some bugs and time to execute each iteration I presume, its results were not good. We also started to implement a user-based algorithm but we did not finish it in time to give it a try.