Prakash Dhimal
Manav Garkel
George Mason University
CS 657 Mining Massive Datasets
Assignment 4: Recommendation System

Introduction

The goal of this assignment is to build a recommendation system for movie ratings for the 20M Movielens dataset using Apache Pyspark. The dataset is found on movielens and contains different csv files. The file used in this assignment is the *ratings.csv* file. This file contains approximately 20M rows of movie ratings data. Each row contains ratings for a movie represented by the *movieId*, given by a user represented by the *userId*. The goal of this project is to implement an ALS based recommender system, an Item-Item Collaborative Filtering recommender system and a hybrid recommender system. Given a specific *userId* and *movieId*, each system predicts the rating it believes the given user would give to that movie. We also use the movies dataset from *movies.csv* to get the name of the movies recommended for a specific user.

Data

The data was read into a spark dataframe with columns “userId”, “movieId”, and “ratings”. After this the data was split into training and testing data with an 80 - 20 split. *Five - Fold cross validation* was used to train the ALS model.

```
ratings = spark.read.option("header", "true").csv('../data/ml-20m/ratings.csv')
ratings = ratings.withColumn('userId', F.col('userId').cast(IntegerType()))
ratings = ratings.withColumn('movieId', F.col('movieId').cast(IntegerType()))
ratings = ratings.withColumn('rating', F.col('rating').cast(DoubleType()))
ratings = ratings.select("userId", "movieId", "rating")
```

Data preprocessing:

There was minimal data preparation to do here since we only had a few columns. Nonetheless, we still had to make sure that our data was read as an integer (*movieId* and *userId*) and double (*ratings*). We dropped the *timestamp* column from the dataset.

The movie dataset did not require any preprocessing steps.

```
movies = spark.read.option("header", "true").csv('../data/ml-20m/movies.csv')
```

ALS

The ALS recommendation system uses an Alternating Least Squares technique with a regularization approach to building a recommendation system. It uses Item-User matrix R, an Item-Factors matrix P, and User-Factors matrix P^T to determine latent factors within the data to explain user to movie ratings. Using this, it tried to find optimal factor weights to minimize the least squares error between the predicted and actual rating. In each iteration, one of the unknown matrices is fixed and the other one is optimized using Gradient Descent. The algorithm continues to find the optimal solution until the error gradient does not change much or until a fixed number of iterations.

Parameters:

We start with a simple ALS model that fits our data as show below:

```
# coldStartStrategy="drop"
als_model = ALS(itemCol='movieId',
                userCol='userId',
                ratingCol='rating',
                nonnegative=True,
                seed=5)
```

Parameter tuning and Cross Validation:

We initialized a param grid search space with multiple values of the rank and maxIter parameters for the ALS model.

```
ranks = [1, 2, 4, 8]
iterations = [10, 20]
param_grid = ParamGridBuilder() \
    .addGrid(als_model.rank, ranks) \
    .addGrid(als_model.maxIter, iterations) \
    .build()
```

The next step was to build a cross validator using the param grid shown above, the als mode, and an evaluator. We use RMSE for the evaluator here.

```
cross_validator = CrossValidator(estimator=als_model,
                                 estimatorParamMaps=param_grid,
                                 evaluator=rmse_evaluator,
                                 numFolds=folds)
```

Item - Item Collaborative Filtering

Item - Item collaborative filtering is a simple yet intuitive approach used to predict the movie rating for a given movie by the given user. In this approach, the first step is to find the k most similar movies to the given movieId. Cosine similarity was used to find the similarities of movies to one another. The next step was identify all the movies among these **k** similar movies that the given user has provided a rating for. Once these movies and their ratings are obtained, the mean of these ratings is used to predict the rating for the given movieId. The advantages of Collaborative filtering approaches is that it works for all kinds of items since no feature selection is involved. However, one of the main disadvantages is that it suffers from the cold start problem. In this scenario, it is difficult to predict movie ratings since Collaborative Filtering is dependent on past data for the given user. Similarly, it also suffers from sparse data problems where past data may be present for the given user, but it may not be sufficient enough to make an accurate prediction.

It is important to note here that we use top K movies based on their similarity to a given movie to then get the ratings. We simply take the average ratings of these movies by the given user to find the rating for a [user, movie] pair.

Hybrid Approach

In most production environments, a stand alone recommendation system is usually not enough to provide the most accurate results. It is common practice to use a combination of systems to predict the target variable. In this report, once the stand alone ALS and Item-Item Collaborative Filtering recommendation systems are built, the final approach was to use a hybrid of the two methods. This was done by taking a weighted mean of Item - Item CF (60%) and ALS (40%).

```
def hybrid_calculation_function(rating_item_item, rating_als):
    if math.isnan(rating_als):
        return rating_item_item
    return (rating_item_item * 0.6) + (rating_als * 0.4)
```

Recommendation

Once the above mentioned approaches were implemented to predict movie ratings for the given users, a movie recommendation system was also built (for simplicity, the results shown in the output file are only for 15 selected users but this can be easily extrapolated to every user in the data). In order to recommend movies to a given user, their predicted ratings by the hybrid approach was used. The ratings were sorted in descending order to get the movieId this user would likely give very high ratings to. Following this, the movies.csv file was used to extract the

movie names based on the movieId's and ten recommendations for movies were made. Example is shown below:

```
Movie Recommendation for user 1959 :
+-------+--------------------+--------------------+
|movieId|               title|              genres|
+-------+--------------------+--------------------+
|    293|Léon: The Profess...|Action|Crime|Dram...|
|    296| Pulp Fiction (1994)|Comedy|Crime|Dram...|
|   2858|American Beauty (...|        Comedy|Drama|
|   2959|   Fight Club (1999)|Action|Crime|Dram...|
|   3761|Bound by Honor (a...|Action|Crime|Dram...|
|   4011|       Snatch (2000)|Comedy|Crime|Thri...|
|   5445|Minority Report (...|Action|Crime|Myst...|
|   5816|Harry Potter and ...|   Adventure|Fantasy|
|   6333|X2: X-Men United ...|Action|Adventure|...|
|   6539|Pirates of the Ca...|Action|Adventure|...|
+-------+--------------------+--------------------+
```

Results

In order to build and evaluate the above methods, the data was split into training (80%) and testing (20%) sets. Additionally, for the ALS model, a five fold cross validation technique was used during the training phase. The main performance measures used to evaluate the approaches were Root Mean Squared Error, Mean Absolute Error and Mean Squared Error.

As can be seen below, the Hybrid method has the lowest error out of all three approaches whereas the ALS approach had the greatest approach out of all the methods. This is in line with prior machine learning techniques wherein it is known that ensemble techniques are known to outperform standalone classifiers. Even though ALS does perform the worst, its performance is only marginally worse than Item - Item Collaborative Filtering.

```
ALS RMSE:  0.8714055532118293
ALS MAE:  0.6780906677816083
ALS MSE:  0.7593476381684141

Item-Item CF RMSE:  0.8678297002539033
Item-Item CF MAE:  0.6530145507271837
Item-Item CF MSE:  0.7531283886427795

Hybrid RMSE:  0.8343864404418501
```

```
Hybrid MAE:  0.6354972325180701
Hybrid MSE:  0.6962007319932211
```

The table below shows the first twenty rows of predictions made by each model. As can be seen below as well, the item - item CF is outperforming the ALS prediction in many rows. This is why the hybrid approach was built to give greater importance to Item - Item CF.

userId	movieId	rating	prediction_item_item_cf	prediction_als	prediction
18	3507	4.0	3.7333333333333334	3.6497035	3.6998814010620116
25	22	4.0	3.3666666666666667	3.0944626	3.257785053253174
35	3793	4.0	4.466666666666667	4.03802	4.295208053588867
39	648	4.0	3.7333333333333334	3.4047287	3.601891460418701
50	5618	5.0	4.566666666666666	4.4561696	4.522467842102051
53	1272	3.0	4.133333333333334	4.2321353	4.172854118347168
62	3911	4.0	4.4	4.5641127	4.465645065307617
64	588	4.0	4.266666666666667	4.4000607	4.320024261474609
68	261	4.0	3.466666666666667	3.6663845	3.546553783416748
82	1203	5.0	4.666666666666667	4.456306	4.582522392272949
91	2617	3.0	3.2333333333333334	2.933093	3.1132372283935545
91	5995	4.0	4.066666666666666	3.767976	3.9471904182434083
104	1014	2.0	3.0	2.641909	2.856763553619385
104	1243	4.0	3.4	3.0300703	3.2520281219482423
104	3167	2.0	3.2	2.7450957	3.0180382919311524
116	112	2.0	2.3	2.2580326	2.283213024139404
119	165	2.0	3.933333333333333	3.8793314	3.911732540130615
119	616	4.0	4.2	3.7850516	4.03402063369751
129	3552	3.0	1.8666666666666667	2.8428931	2.2571572494506835
137	4308	5.0	4.466666666666667	3.9247928	4.249917106628418

The table above was saved to a csv file in order to perform additional analysis on the predictions of each method to better understand their performance. As can be seen in fig. 1 below, the predicted ratings were plotted for each of the approaches along with the true ratings.

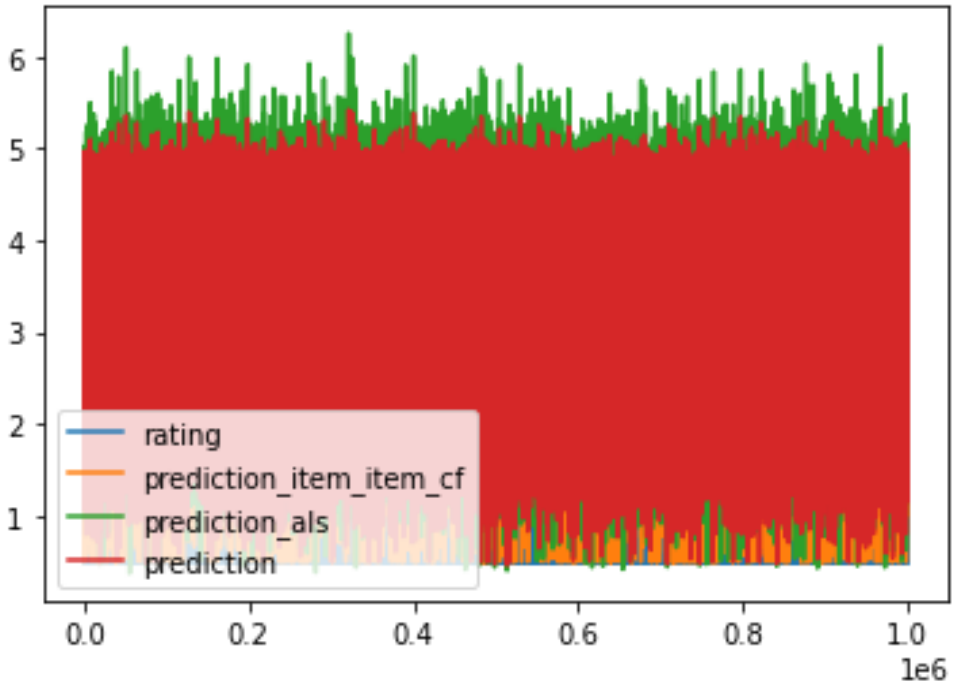

Fig. 1: Ratings for each of the approaches

ALS seems to have the greatest range with a big chunk of predictions exceeding a value of 5.0. Additionally, it can be seen in fig. 2 that ALS does not make many predictions in the range of 0 - 1 whereas Item - Item Collaborative Filtering does have greater coverage in this area (fig. 4).

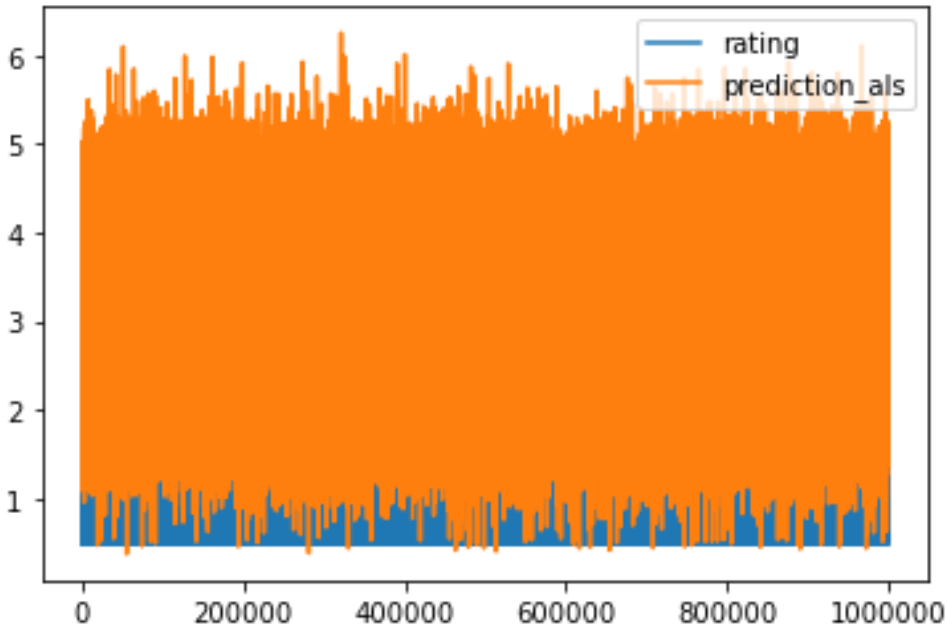

Fig. 2: ALS predictions ratings vs True Rating

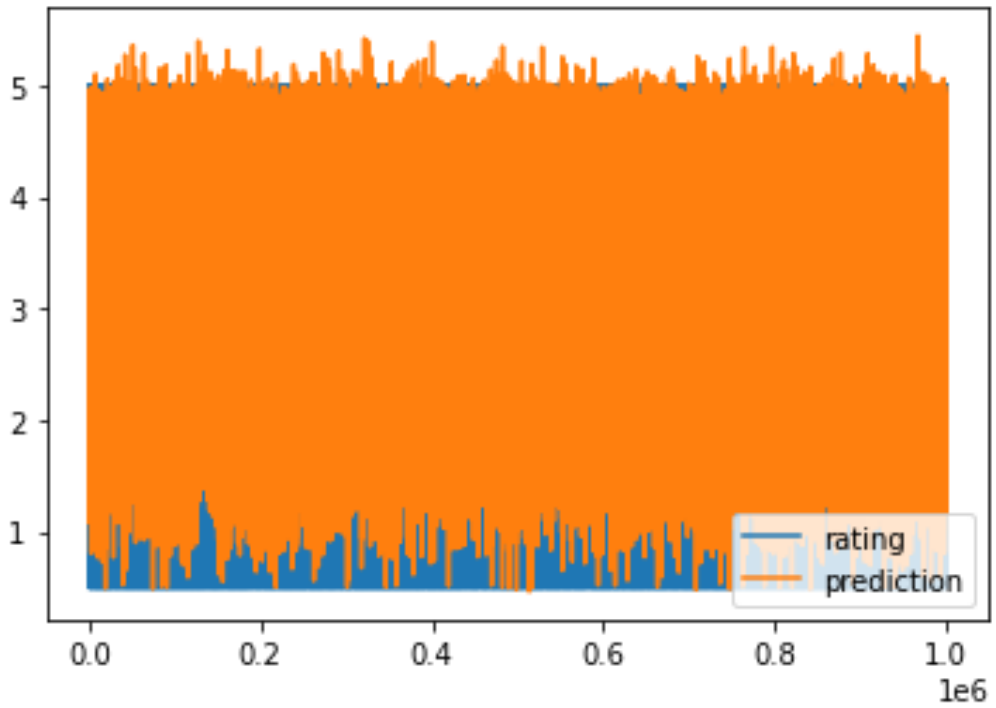

Fig. 3: Hybrid Prediction vs True Rating

Due to ALS's inflated predictions, even the hybrid predictions can be seen to exceed 5.0 and have low numbers of predictions between 0 - 1.

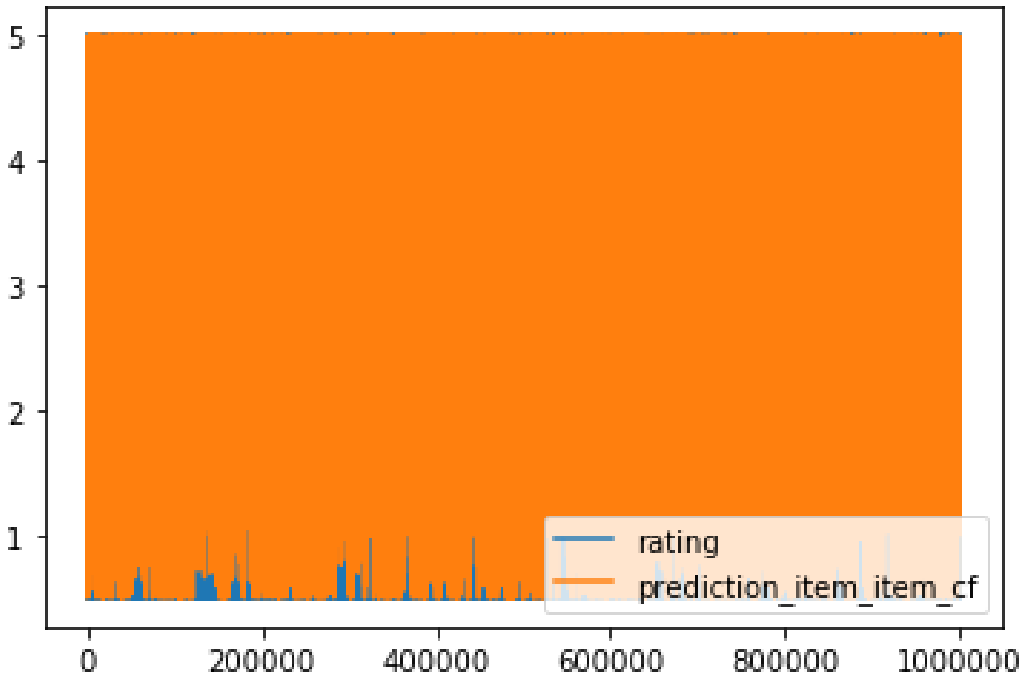

Fig. 4: Item - Item Collaborative Filtering vs True Rating

One of the key areas where Item - Item Collaborative filtering outshines ALS based approach is that it's predicted ratings do not exceed 5.0. This is due to the averaging technique used in the method. Since none of the given ratings will exceed 5.0, an average of those ratings will also not exceed 5.0. Additionally, Item - Item Collaborative Filtering seems to have a reasonable number in the range of 0 - 1 giving better coverage to this area.