MovieLens Movie Recommendation System

ANALYSIS REPORT

Guided by:

Dr. David Belanger

Gaurav Sawant

Dhaval Sawlani

BIA-678-A

Spring 2018

1. Introduction 3

2. Dataset and Cleaning 3

3. Recommender Systems & Collaborative Filtering: 4

4. Alternating Least Squares Matrix Factorization 5

Alternating Least Squares(ALS) Results 7

Cosine Similarities Results 9

6. Pearson's Correlation Coefficient(r) 10

Pearson's Correlation Coefficient Results 11

7. Conclusion 12

References 12

1. Introduction

In this project we attempt to build a movie recommender system using the Movie Lens data set found on the group lens website(grouplens, https://grouplens.org/). The motivation behind building a recommender system is the fact that recommender systems are one of the most popular and widely used applications of Big Data that affect people everyday and yet we know very little about the actual working of these wonderful systems. Recommender Systems are being actively used by many small and large corporations and websites around the world and we were interested in learning the underlying working of these huge and highly effective applications of Big Data. Other goals of this project include, studying the various popular and well-established recommender algorithms, building a recommender system of our own and studying the performance and scaling of the various algorithms in our recommender engine. We have narrowed our focus to only collaborative filtering techniques as they are the most popularly used techniques in real world applications.

2. Dataset and Cleaning

We have used 2 sizes of datasets for this project both available on Grouplens website (grouplens, https://grouplens.org/). First one contains 100,000 ratings collected from 1000 users on 1700 movies and the second one contains 1 million ratings from 6000 users on 4000 movies. Multiple datasets are a part of the movielens data suite and are present in both the data sizes. From both the data sizes we have used two datasets which are as follows: Ratings Data(UserId, MovieId, Ratings(1-5), TimeStamp) and Movie Data(Movie_id, Titles and Genres). All the datasets are clean datasets and we have not performed any data cleaning activity.

3. Recommender Systems & Collaborative Filtering:

Recommender Systems have become extremely popular today and are currently being used for a wide variety applications and areas including movies, restaurants, music, online shopping, jobs among many others. Recommendation systems generally produce recommendations using either Content Based Filtering or Collaborative Filtering approaches and while Collaborative Filtering methods build models based on users' past decisions and decisions of made by other similar uses, Content-based filtering methods use a series of discrete charecteristics of an item in order to recommend additional items with similar properties(Wikipedia, Recommender Systems, https://en.wikipedia.org/wiki/Collaborative_filtering). Among the two major categories, Collaborative Filtering is today the more popular approach in building Recommender Engines and we will be focusing on Collaborative Filtering techniques in this project. Collaborative Filtering Models are based are based on the assumptions that people like things similar to other things that they like and things that are liked by other people with similar taste (Prince Grover, Various Implementations of Collaborative Filtering, Towards Data Science, 2017). Figure 1 is an example of the basic idea behind Collaborative

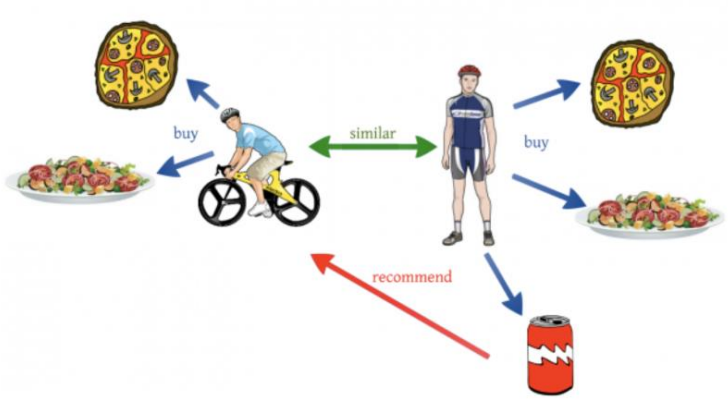

Figure 1. Collaborative Filtering Basic Idea Example

(Prince Grover, Various Implementations of Collaborative Filtering, Towards Data Science, 2017, Figure 1)

Filtering. Collaborative Filtering techniques are broadly divided into 2 approaches namely Memory Based Approaches and Model Based Approaches and in this project we have Memory Based as well as Model Based techniques. Memory based techniques use user ratings to compute similarity between users/items to make recommendations. In Model Based techniques, models are developed using different machine learning algorithms to predict users' rating of unrated items.

4. Alternating Least Squares Matrix Factorization

One of the most popular approaches to solve co-clustering problem in a collaborative filtering recommender systems is Matrix Factorization and in its simplest form, it assumes a matrix $R \varepsilon R^{m*n}$ of ratings given by '*m*' users to '*n*' items. Applying this technique on R will end up factorizing R into matrix $U \varepsilon R^{m*k}$ and $P \varepsilon R^{m*k}$ such that R is approximately equal to *U*P* introduces a new quantity, *k*, that serves as both *U's* and *P's* dimensions(How do you build a "People who bought this also bought that" -style recommendation engine, Data Science Made Simpler, 2015).

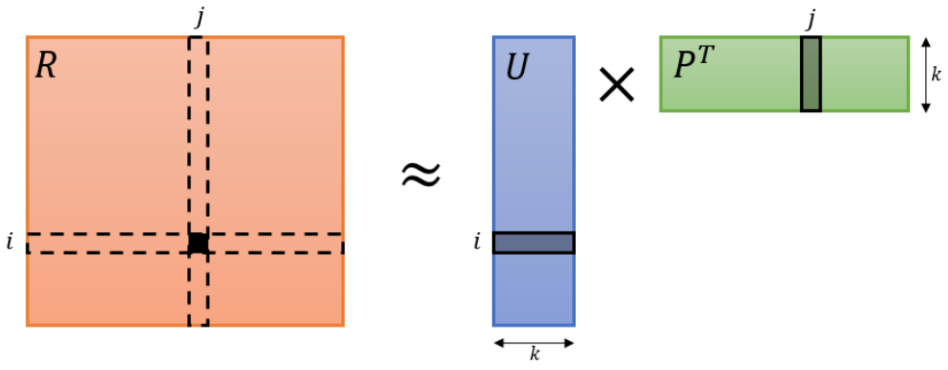

Figure 2. Visual Representation of Matrix Multiplication

(How do you build a "People who bought this also bought that" -style recommendation engine, Data Science Made Simpler, 2015)

Matrix Factorization makes the following assumptions:

- Each user can be described by *k* attributes or features
- Each item (in our case movie) can be described by a set of *k* attributes
- If we multiply each feature of the user by the corresponding feature of the movie and add everything, we'll get a good approximation of the ratings that the user would give to the movie which can be seen in the equation in Figure 3 (Explicit Matrix Factorization: ALS, SGD, and all that Jazz, Insight Data, Mar 16,2016)

User *u* takes form of *k*-dimensional vector *x_u* and the item/movie *i* takes form of *k*-dimensional vector *y_i* and the *k* attributes are called latent factors (Explicit Matrix Factorization: ALS, SGD, and all that Jazz, Insight Data, Mar 16,2016).

$$\hat{r}_{ui} = \mathbf{x}_u^\top \cdot \mathbf{y}_i = \sum_k x_{uk} y_{ki}$$

Figure 3. Matrix Factorization Approximation

We finally produce a loss function by minimizing the square of difference between predictions and ratings.

We have used Alternating Least Squares method in this project which uses Matrix Factorization to produce recommendation. For ALS minimization, we hold one set of latent vectors constant and then we take the derivative of the loss function with the other set of vectors (Explicit Matrix Factorization: ALS, SGD, and all that Jazz, Insight Data, Mar 16,2016). We set derivative equal to zero and solve for the non-constant vectors and then in the alternating part we hold the solved vectors constant and take derivative of the

loss function with respect to the previously constant vectors and then keep alternating back and forth until convergence.

Alternating Least Squares(ALS) Results

We have tested the ALS algorithms on our local machines as well as on the cluster to check how the algorithm scales. The following Figures are intuitive graphs about the scalability and results of the ALS algorithm.

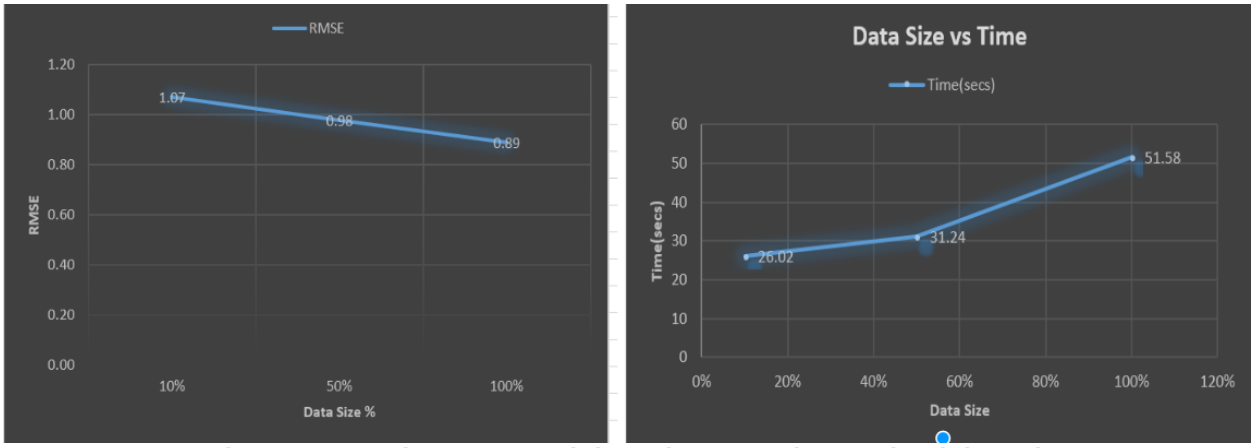

Figure 4 Data Size vs RMSE (Cluster) & Data Size vs Time(Cluster)

Figure 4 actually shows two things. First, it shows a comparison of RMSE and Data Size on cluster and it also shows a comparison between Data Size and Elapsed Time on cluster. As expected RMSE goes on decreasing as the data size increases and elapsed time goes on increasing with increase in data size. Figure 5 shows the time taken on by the ALS algorithm on local as well as cluster at various data sizes. As seen in the figure the amount of time taken by the local system is always a lot more than the time taken for the same data size on cluster.

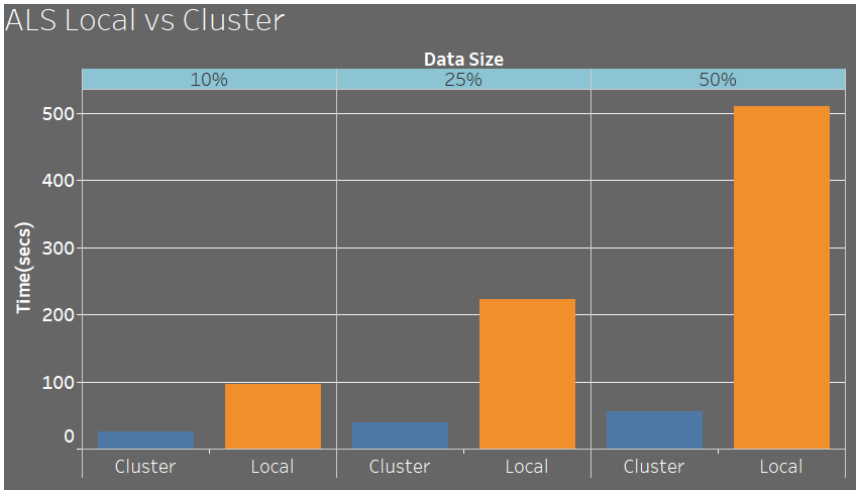

Figure 5 ALS Performance Local vs Cluster

5. Cosine Similarities

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them(Cosine similarity, Wikipedia). In movie recommendation, it is the measure of the similarity between the movies taking the user ratings into account. We have used RDDS and key, value pairs to obtain the cosine similarities between movies. While calculating the Cosine similarities between two vectors, range taken into account is 0-1. Closer to 1 means stronger similarity index and vice versa.Cosine similarity can be calculated using the following formula(Cosine similarity, Wikipedia):

Steps that we followed to calculate cosine similarities are as follows:

1. Created a cartesian join of the data set of the data set(Took maximum time on cluster and on local)
2. Filtered Duplicates
3. Group by movie ID to obtain all the reviews of that movie in a single (key, value) pair

4. Computed Cosine Similarities
5. Filtered Movies with more than 50 reviews and a score greater than 0.97

Cosine Similarities Results

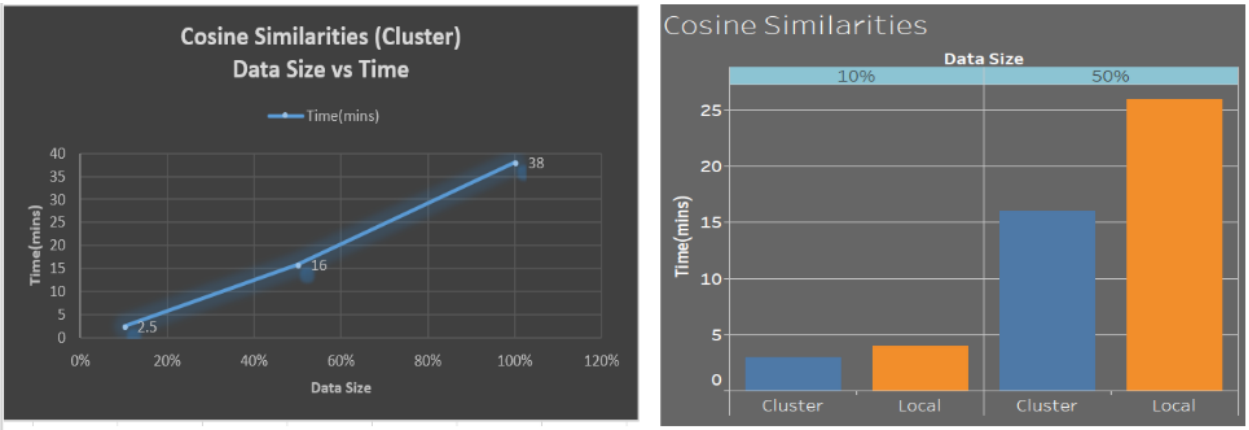

Figure 6 Cosine Similarities Results

Figure 6 shows a graph between data size and time elapsed and another graph comparing the time required on cluster and on local machines. As expected the time elapsed increases almost linearly with increase in data size. Also, after comparing the time taken on cluster and on local according to varied data sizes, we can conclude that time taken by local machines is always significantly greater than the time taken on cluster. The results as seen on cluster can be seen in Figure 7.

```
18/04/28 18:12:15 INFO StandaloneSchedulerBackend: SchedulerBackend is ready for scheduling beginning after reached minRegisteredResourcesRatio: 0.0
Top similar movies for [u'Star Wars (1977)']
[u'Empire Strikes Back, The (1980)']    score: 0.989552207839   strength: 345
[u'Return of the Jedi (1983)']  score: 0.985723086125   strength: 480
[u'Raiders of the Lost Ark (1981)']     score: 0.981760098873   strength: 380
[u'20,000 Leagues Under the Sea (1954)']        score: 0.97893856055    strength: 68
[u'12 Angry Men (1957)']        score: 0.977657612045   strength: 109
[u'Close Shave, A (1995)']      score: 0.977594829105   strength: 92
[u'African Queen, The (1951)']  score: 0.976469222267   strength: 138
[u'Sting, The (1973)']  score: 0.975151293774   strength: 204
[u'Wrong Trousers, The (1993)'] score: 0.974868135546   strength: 103
[u'Wallace & Gromit: The Best of Aardman Animation (1996)']     score: 0.97418161283    strength: 58
```

Figure 7 Results on cluster

6. Pearson's Correlation Coefficient(r)

The Pearson's correlation coefficient, also referred as Pearson's *r,* or bivariate correlation is a measure of the linear correlation between two variables *X* and *Y* (Pearson's correlation coefficient, Wikipedia). It has a range of -1 to 1, where 1 means total positive correlation, 0 means no correlation and -1 means total negative correlation. Pearson correlation coefficient usually normalizes your similarity index and thus punishes the instances where cosine similarity would fail. The limitation of Pearson correlation coefficient is that it does not explain the relation between the dependent and the independent variables. But, this does not matter in our case, because if movie A has an r of 0.8 with movie B and vice-versa, we can anyway say that movie A is similar to movie B. Pearson's correlation coefficient when applied to a sample is commonly represented by the letter *r* and may be referred as sample correlation coefficient(Pearson correlation coefficient, Wikipedia). So if we have one dataset $\{x_1,...,x_n\}$ containing *n* values and another dataset $\{y_1,...,y_n\}$ containing *n* values then that formula for *r* is:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where,

n is the sample size

x_i , y_i are the sample sizes indexed with *i*

x_i is the sample mean and analogously for *y*

Rearranging the formula we get *r* as follows-

$$r = r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sqrt{(\sum x_i^2 - n\bar{x}^2)}\sqrt{(\sum y_i^2 - n\bar{y}^2)}}.$$

The actual steps followed by us are as follows:

1. Created a cartesian join of the data set. (Took maximum time on the cluster and on local)
2. Filtered duplicates
3. Group by movieID to obtain all the reviews of that movie in a single (key,value) pair
4. Computed Cosine similarity.
5. Filtered Movies with more than 50 reviews and greater than 0.5 score

Pearson's Correlation Coefficient Results

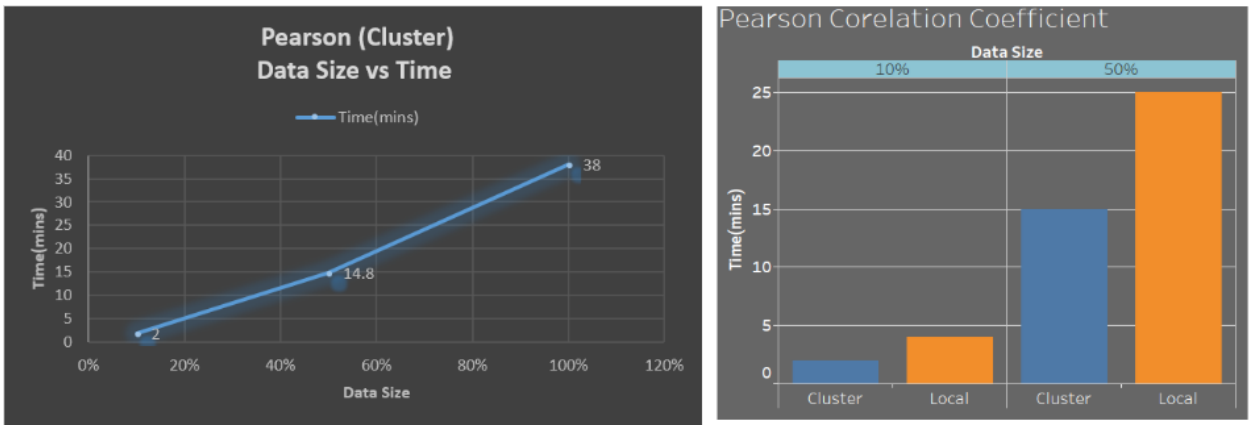

Figure 8 Pearson Correlation Coefficient Results

Figure 8 shows a graph of data size versus time and a comparison between the time taken at the cluster versus the time taken at the local machine for performing Pearson's correlation coefficient. As expected, the time taken on cluster increases almost linearly with the increase in data size. Also, the time taken on the local machine is always more than the time taken on the cluster for the particular data size. Figure 9 shows the actual results seen on the cluster while using Pearson's correlation coefficient.

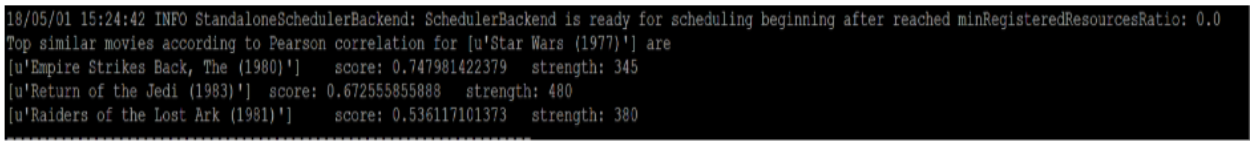

Figure 9 Pearson's Correlation Coefficient Result

7. Conclusion

As expected, in Alternating Least Squares(ALS) method, the time taken increases with an increase in the size of the data and the RMSE decreases with the increase in the size of the data. Cosine Similarities and Pearson's correlation coefficient overall produce almost similar results. Pearson's correlation performs only slightly better than Cosine similarities. To conclude, in all methods that we studied, time taken on cluster is considerably less than the time taken on the local machine. In the future, we are planning to study more collaborative filtering techniques and how they scale. Also, we will try to reduce the RMSE achieved while performing ALS. Finally, we plan to implement some Hybrid Collaborative Filtering techniques to run on larger data sets.

References

1. Wikipedia, Recommender Systems, https://en.wikipedia.org/wiki/Collaborative_filtering
2. Prince Grover, Various Implementations of Collaborative Filtering, Towards Data Science, 2017
3. How do you build a "People who bought this also bought that" -style recommendation engine, Data Science Made Simpler, 2015
4. Explicit Matrix Factorization: ALS, SGD, and all that Jazz, Insight Data, Mar16,2016, https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea
5. (Cosine similarity, Wikipedia, https://en.wikipedia.org/wiki/Cosine_similarity)

6. (Pearson correlation coefficient, Wikipedia, https://en.wikipedia.org/wiki/Pearson_correlation_coefficient)