COMPARATIVE ANALYSIS OF MOVIE RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING IN SPARK ENGINE

Article · October 2017

CITATIONS
6

READS
3,069

3 authors:

Goutham Miryala
North Dakota State University
4 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Rahul Gomes
University of Wisconsin - Eau Claire
38 PUBLICATIONS **416** CITATIONS
SEE PROFILE

Karanam RAVICHANDRAN Dayananda
North Dakota State University
7 PUBLICATIONS **59** CITATIONS
SEE PROFILE

Volume 8, No. 10, October 2017

Journal of Global Research in Computer Science

RESEARCH PAPER

Available Online at www.jgrcs.info

ISSN 2229-371X

COMPARATIVE ANALYSIS OF MOVIE RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING IN SPARK ENGINE

Goutham Miryala[*1], Rahul Gomes[2] and Karanam Ravichandran Dayananda[3]
Department of Computer Science, North Dakota State University, 1420 Bolley Dr, Fargo, ND 58105, USA
goutham.miryala@ndsu.edu[1]
rahul.gomes@ndsu.edu[2]
Karanam.dayananda@ndsu.edu[3]

***Abstract*—**A personalized recommendation system learns user specific profiles from user feedback, so it can deliver information tailored to each individual user's interest. A system serving millions of users can learn a better user profile for a new user, or a user with little feedback, by borrowing information from other users using a regression classifier. Regression is a technique borrowed by machine learning from the field of statistics. This paper proposes to build a recommendation system for the users. This paper uses a machine learning algorithm, which is imported from the spark. Apache spark is a fast and general engine used for large scale data processing. The efficacy and efficiency of the Alternating Least Squares (ALS) algorithm is compared with Singular Value decomposition, K-Nearest Neighbor algorithm, and Normal predictor algorithm. The ALS algorithm is justified by theory and demonstrated on actual user data from Movie Lens.

***Keywords*:** ALS, Apache Spark, regression classifier, recommendation systems

I. INTRODUCTION

The future of the web is personalization, and has achieved great success in industrial applications. For instance, online stores, such as Netflix and Amazon, provide customized recommendations for additional products or services based on a user's history. Recent applications such as My MSN, My Google, flip kart, Facebook and Google News have attracted attention due to their potential ability to infer a user's interests from the user history.

There are different recommendation techniques, which aims for the same purpose of recommending items to the users, these different techniques can be generalized into three(Adomavicius and Tuzhilin, 2005):

a. Collaborative filtering

b. Content-based filtering

c. Hybrid systems

One major personalization topic in the information retrieval community is content-based personal recommendation systems. These systems learn user-specific profiles from user feedback and recommend information tailored to each individual user's interest without requiring the user to make an explicit query. Learning the user profiles is the core problem for these type of systems.

A user profile is a classifier, which identifies whether a document is relevant to the user or not, or a regression model(Melville, Mooney and Nagarajan, no date)(Agarwal and Chen, 2009) that explains how relevant a document is to the user. One major challenge of building a personalization or recommendation system is that the profile learned for a particular user is usually of low quality when the amount of data from that particular user seems to be small. This can be stated as the "cold start" problem. The cold start problem is that any new user must endure poor initial performance until sufficient feedback from that particular user is provided to learn a reliable user profile.

This project builds a user based movie recommender systems which focuses on collaborative filtering. The dataset is retrieved from MovieLens(Harper and Konstan, 2015), which is the most recent one with a little over 20 million ratings for 33,000 movies by almost 240,000 users. These ratings are given from values in between 1 to 5, in which the prior means less preferable and the later represents highest preference. The recommender system was built on top of spark(Geyer *et al.*, 2008), an open source, scalable machine learning library from Apache. Apache Spark is an open-source cluster-computing framework developed at the University of California, Berkeley's AMPLab. Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance.

In spark collaborative filtering is used, which is commonly used for recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix. The spark.mllib(Gopalani and Arora, 2015) supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries. The spark.mllib uses the alternating least squares (ALS) algorithm to learn these latent factors.

The standard approach to matrix factorization based collaborative filtering(Yehuda Koren, Bell and Volinsky, 2009) treats the entries in the user-item matrix as explicit preferences given by the user to the item. For instance, users giving ratings to movies. Instead of trying to model the matrix of ratings directly, this approach treats the data as numbers in turn representing the strength in the observation

of user actions (such as the number of clicks, or the cumulative duration someone spent viewing a movie). The calculated numbers are related to the level of confidence in observed user preferences, rather than explicit ratings given to the items. The model tries to find latent factors that can be used to predict the expected preference of a user for an item.

The experiments include comparative analysis of ALS algorithm's Root Mean Square Error (RMSE) values with Singular Value decomposition algorithm, K-Nearest Neighbor algorithm, and Normal predictor algorithm. Later, ALS algorithm is used to build the recommendation system.

II. BACKGROUND

In the IR community providing recommendation to users has been a major problem since 1970's. The approaches used for solving can be categorized into two categories(Pilászy and Tikk, 2009): content-based filtering and collaborative filtering. Content-based filtering is used in the scenario where a recommendation system monitors a document stream and push documents that match a user profile to the corresponding user. The users read the delivered documents and provide explicit relevance feedback, which the filtering system then uses to update the user's profile using relevance feedback retrieval models (e.g. Boolean models, traditional probabilistic models, inference networks, vector space models, and language models) or machine learning algorithms (e.g. Singular Value Decomposition (SVD), neural networks, K nearest neighbors (K-NN) clustering, Normal Prediction). Collaborative filtering uses document content to recommend items to a user by leveraging information from other users with similar tastes and preferences in the past. Memory based heuristics and model based approaches have been used in collaborative filtering task.

$$f(U,M) = \sum_{(i,j)} w_{i,j} \left(r_{i,j} - u_i \times m_j\right)^2 + \lambda\left(\sum_i n_{u_i} \|u_i\|^2 + \sum_i n_{m_j} \left\|m_j\right\|^2\right)$$

The K-nearest neighbor algorithm(Wen, 2008) is used to determine the rating on basis of the user rating. The assumption is that the rating of user u on a movie m is determined based on the ratings provided by user u on movies similar to m. However, conditioning on the set of users who rated the item can introduce bias into the predicted rating(Marlin *et al.*, 2007). Another algorithm, which is known as Singular Value Decomposition can be used to factorize a complex matrix. In the scenario of user-movie rating, which is considered as a symmetric matrix with positive eigen values and further evaluated. However, missing values in the user-item rating matrix can pose a serious challenge to the applicability of SVD in certain scenarios(Y. Koren, Bell and Volinsky, 2009).

Machine learning frameworks have been used extensively in various domains. Their implementation ranges from geospatial data(Gomes and Straub, 2017) to self-driving cars(Straub *et al.*, 2017). Since they produce optimum results in minimal time they can be used for recommendation systems. According to (Gopalani and Arora, 2015), Apache Spark performs better than the MapReduce for Machine Learning algorithms. Apache Spark (*Research | Apache Spark*, no date), created by UC Berkeley, was designed to support much wider class of applications than MapReduce. Resilient Distributed Dataset (RDD) (Zaharia *et al.*, 2012) is created to support the spark application without replication of memory. During the event of failure, the data is recreated using lineages.

Spark MLlib is a distributed machine learning framework on top of Spark Core that can be used in large part of distributed memory-based Spark architecture, is as much as nine times as fast as the disk-based implementation used by Apache Mahout (according to benchmarks done by the MLlib developers against the Alternating Least Squares (ALS) implementations, and before Mahout itself gained a Spark interface), and scales better than Vowpal Wabbit. Many common machine learning and statistical algorithms have been implemented and are shipped with MLlib which simplifies large scale machine learning pipelines.

Alternating Least Squares (ALS) has been implemented in many machine learning libraries and widely used in both academia and industry. ALS models the rating matrix (R) as the multiplication of low-rank user (U) and product (V) factors, and learns these factors by minimizing the reconstruction error of the observed ratings. The unknown ratings can subsequently be computed by multiplying these factors. In this way, companies can recommend products based on the predicted ratings and increase sales and customer satisfaction.

III. OVERVIEW OF ALS ALGORITHM

The first step in ALS algorithm is to create a matrix, which contains a set of values of movies and user ratings of the movies. The system as shown in figure 1 does not contain all ratings for each movie. ALS computes the predicted values using the following base equation (1)(Zhou *et al.*, 2008).

Table 1. Values of movies and user ratings\

	Goblet of Fire	Matrix	Terminator 2	La La Land	Iron Man	Up
U1	4			5	1	
U2	5	5	4			
U3				2	4	5
U4		3				

Equation 1. User movie rating matrix

This equation tries to minimize the cost function. Values for empty slots as shown in Table 1 is filled by approximating rating as UxM, where user matrix U is a N_{user} by N_{factor} matrix and item matrix M is a N_{factor} by N_{movie} matrix(Zhou *et al.*, no date).

SPARK Mlib uses the ALS algorithm to estimate Y using X where Y represents the column matrix and X represents the row matrix or the users. This process is followed with the ALS algorithm estimating X using Y. The process is continued using a certain regularization parameter

λ. After certain number of iterations, a convergence point, the RMSE values become constant. Higher the numbers of iterations lower the RMSE values.

Item Matrix M is computed calculated based on other factors such as rating, genre , and years. For the study, ratings of the users are used to compute matrix M. Some considerations in this equation include the value of w_{ij} , which is 0 if no rating is present and 1 if the user rates the movie. The term n_{ui} refers to the number of ratings available for the user U_i and n_{mj} refers to the number of ratings available for movie Mj. To summarize, the first term minimizes R – U x M using the weighting factor w_{ij} and second term is the weighted λ regularization ensuring that a smaller subset of U x M values generate acceptable results.

The regularization parameter λ is responsible for ensuring proper fit of the data values. Higher λ values increase the RMSE values producing a higher deviation from the actual ratings and the predicted ratings. However, this makes the model versatile enough, so that if unknown values are fed in the model, it can predict values with a higher degree of accuracy. Lowering the regularization parameter reduces the RMSE values. However, the model cannot handle unknown data effectively and sometimes may fail to provide dependable predictions. Thus, it is evident that a lower λ values leads to overfitting of data and larger λ values cause under fitting of data.

IV. SPARK IMPLEMENTATION

The entire design of the system can be divided into two parts, as shown in figure 2.

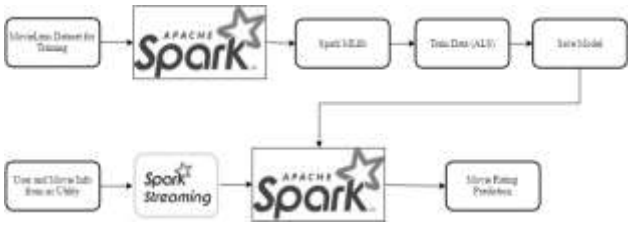

Figure 2. Implementation of recommendation in Spark

1. Building the model
   a. This stage starts with loading the dataset with one million movie ratings into the system.
   b. Once the dataset is loaded into Spark, various combinations of iterations (1, 3, 5, 15, 20) and Regularization parameters (0.01, 0.1, 1, 2, 3) for training and testing will be used.
   c. The data is also trained and tested with 80:20 and 60:40 ratios for the evaluation purposes.
   d. Each RMSE value is collected from testing the and evaluated to find the best combination to be applied for the prediction model.
   e. The model with the lowest RMSE is saved for the future usage in the prediction system.
2. Prediction System
   a. This stage starts with loading the ALS model into the Spark Core built in the previous stage.
   b. Then the Apache Spark Stream is started to pick the streams from the network every second.
   c. User and Movie information is sent over the network using 9999 as port via Netcat Utility in Linux.
   d. The collected data in Spark Stream is sent to Apache Spark Core for the further processing in the distributed computing environment.
   e. The loaded ALS model will be applied on each incoming stream message.
   f. Finally, a predicted movie rating is appended to output console.

V. RESULTS

a) ANALYSIS OF ALS ALGORITHM

Testing was done in two phases. the first set shown in figure 3 contains RMSE values for an 80-20 dataset where 80% is training data and 20% is testing data. it is observed that as the number of iterations is increased, the RMSE converges to a minimum value. The minimum value differs according to the regularization parameter λ used for the dataset. Higher λ reduces overfit of data thereby reducing the accuracy of the model and increasing the RMSE. The second set of results shown in figure 4 contains the results of 60 – 40 dataset where 60 % is training data and 40 % is testing data. The same trend of RMSE was observed along with a similar variation of the λ. However it is observed that using a larger training dataset of 80-20 yields a model that has a lower RMSE when compared to the 60-40 dataset.

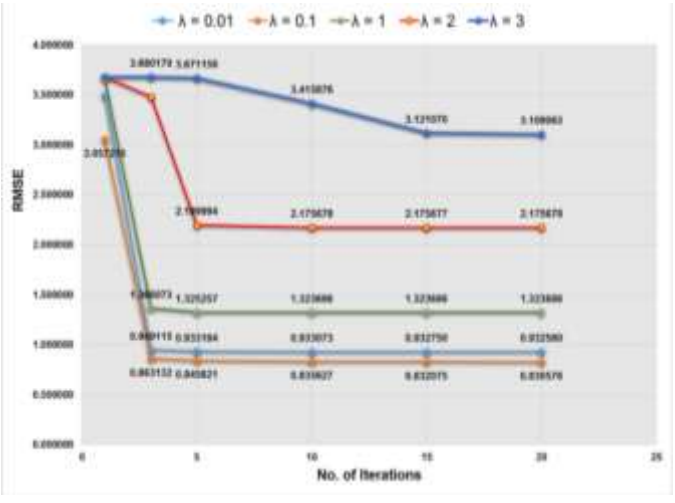

Figure 2. RMSE Values with 80-20 dataset

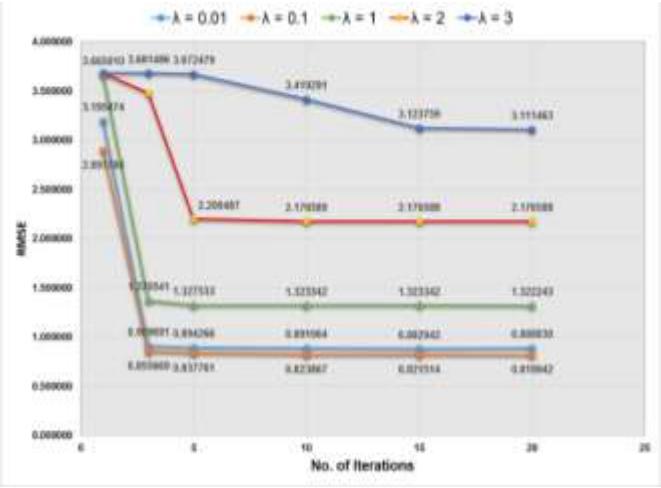

Figure 3. RMSE Values with 60-40 dataset

The tables2 and 3 demonstrates that tests were done for five regularization parameters and six different iterations. The values of λ chosen were 0.01, 0.1, 1.0, 2.0 and 3.0. The ALS algorithm was run for 1, 3, 5, 15 and 20 iterations. Further iterations were not considered as the data converged to a minimum value and the change in the observed RMSE was negligible.

Table 2. 80 – 20 Dataset

# Iteration	λ	RMSE	λ	RMSE	Λ	RMSE	λ	RMSE	Λ	RMSE
1	0.01	0.894266	0.1	2.897784	1	3.66501	2	3.681256	3	3.68338
3	0.01	0.909691	0.1	0.855669	1	1.370541	2	3.478916	3	3.681486
5	0.01	0.894266	0.1	0.837761	1	1.327533	2	2.200487	3	3.672479
10	0.01	0.891064	0.1	0.823867	1	1.323342	2	2.176589	3	3.419291
15	0.01	0.882942	0.1	0.821514	1	1.323342	2	2.176588	3	3.123759
20	0.01	0.88883	0.1	0.819942	1	1.323343	2	2.176588	3	3.111463

Table 3. 60 – 40 Dataset

# Iteration	λ	RMSE	λ	RMSE	λ	RMSE	λ	RMSE	λ	RMSE
1	0.01	3.493471	0.1	3.057215	1	3.663589	2	3.680112	3	3.682051
3	0.01	0.949115	0.1	0.863132	1	1.366073	2	3.480148	3	3.68017
5	0.01	0.933184	0.1	0.845621	1	1.325257	2	2.199994	3	3.671156
10	0.01	0.933073	0.1	0.835627	1	1.323686	2	2.175678	3	3.415876
15	0.01	0.93275	0.1	0.832075	1	1.323686	2	2.175677	3	3.12107
20	0.01	0.93258	0.1	0.830576	1	1.323686	2	2.175678	3	3.109063

b) COMPARATIVE ANALYSIS OF ALS WITH OTHER ALGORITHMS

To compare the efficacy of ALS algorithm in movie recommendation system, the RMSE values were evaluated with three different algorithms: Singular Value Decomposition, Normal Prediction Algorithm and K-Nearest Neighbor Algorithm. The tests showed that ALS Algorithm with 80-20 dataset having regularization parameter 0.1 with 20 iterations generates the smallest RMSE when compared to all other algorithms. This model is later used for prediction of movie ratings.

Table 4.RMSE results of algorithms

Algorithm	RMSE
Alternating Least Squares	0.819942
K-Nearest Neighbor	0.9329
Singular Value Decomposition	0.9449
Normal Predictor	1.5248

c) MOVIE RATING PREDICTION USING ALS

The model generated by ALS algorithm using 80-20 dataset with regularization parameter 0.1 and 20 iterations was used for movie rating prediction as it had the lowest RMSE value. The prediction results are shown in Table 5.

Table 5. Predictions using ALS

movieID	Rating	Timestamp	UserID	Prediction
3702	4.0	1139000265	130	3.1674788
44	4.0	1302655511	199	2.679484
319	3.5	1344870428	199	4.928402
2150	3.0	1260759194	1	3.6615803

VI. CONCLUSION

Recent applications such as My MSN, My Google, Flipkart, Facebook and Google News have attracted attention due to their potential ability to infer a user's interests from the user history. The movie recommendation system is built using collaborative filtering technique, which is predicted based on other users rating and suggested to the required user. The collaborative filtering is achieved using ALS algorithm implemented in spark engine. Different types of recommendation techniques have been explained in detail. A comparative study between regularize ation parameter and RMSE is calculated in ALS algorithm. From this study, it can be inferred that, higher regularization parameter increases RMSE and vice versa. The ALS algorithm is compared with SVD, KNN and Normal Predictor and the results show that the ALS is the best algorithm for the recommendation system. A simple Recommendation System has been built using the best ALS model.

VII. REFERENCES

Adomavicius, G. and Tuzhilin, A. (2005) 'Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions', IEEE Transactions on Knowledge and Data Engineering, 17(6), pp. 734–749. doi: 10.1109/TKDE.2005.99.

Agarwal, D. and Chen, B.-C. (2009) 'Regression-based latent factor models', in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09. New York, New York, USA: ACM Press, p. 19. doi: 10.1145/1557019.1557029.

Geyer, W. et al. (2008) 'Recommending topics for self-descriptions in online user profiles', in Proceedings of the 2008 ACM conference on Recommender systems - RecSys '08. New York, New York, USA: ACM Press, p. 59. doi: 10.1145/1454008.1454019.

Gomes, R. and Straub, J. (2017) 'Genetic algorithm for flood detection and evacuation route planning', in Velez-Reyes, M. and Messinger, D. W. (eds) SPIE Defense+ Security, p. 1019816. doi: 10.1117/12.2266474.

Gopalani, S. and Arora, R. (2015) 'Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means', International Journal of Computer Applications, 113(1), pp. 975–8887.

Harper, F. M. and Konstan, J. A. (2015) 'The MovieLens Datasets', ACM Transactions on Interactive Intelligent Systems. ACM, 5(4), pp. 1–19. doi: 10.1145/2827872.

Koren, Y., Bell, R. and Volinsky, C. (2009) 'Matrix Factorization Techniques for Recommender Systems', Computer, 42(8), pp. 30–37. doi: 10.1109/MC.2009.263.

Koren, Y., Bell, R. and Volinsky, C. (2009) 'Matrix Factorization Techniques for Recommender Systems', Computer, 42(8), pp. 42–49. doi: 10.1109/MC.2009.263.

Marlin, B. M. et al. (2007) 'Collaborative Filtering and the Missing at Random Assumption', Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, p. 9. doi: 10.1.1.80.6934.

Melville, P., Mooney, R. J. and Nagarajan, R. (no date) 'Content-Boosted Collaborative Filtering for Improved Recommendations'.

Pilászy, I. and Tikk, D. (2009) 'Recommending new movies', in Proceedings of the third ACM conference on Recommender systems - RecSys '09. New York, New York, USA: ACM Press, p. 93. doi: 10.1145/1639714.1639731.

Research | Apache Spark (no date).

Straub, J. et al. (2017) 'CyberSecurity considerations for an interconnected self-driving car system of systems', in 2017 12th System of Systems Engineering Conference, SoSE 2017. doi: 10.1109/SYSOSE.2017.7994973.

Wen, Z. (2008) 'Recommendation System Based on Collaborative Filtering'.

Zaharia, M. et al. (2012) 'Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing', Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, pp. 2–2.

Zhou, Y. et al. (2008) 'Large-scale parallel collaborative filtering for the netflix prize', in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 337–348. doi: 10.1007/978-3-540-68880-8_32.

Zhou, Y. et al. (no date) 'Large-scale Parallel Collaborative Filtering for the Netflix Prize'.