Comparison between Collaborative Filtering Algorithms: Weighted Bipartite Graph Projection and Alternating Least Squares

April 29, 2021

Abstract

This paper aims to compare the performance of two different collaborative filtering methods in aspect of prediction accuracy and time efficiency. The two approaches chosen in this paper are the weighted bipartite graph projection (WBGP) approach with recommendation power as the similarity measure and the matrix factorization approach with alternating least squares (ALS) learning algorithm. The two algorithms are implemented on a small subset of Yelp Open Dataset where the businesses are all from Washington state and the root mean squared error (RMSE) is chosen as the evaluation metrics for the comparison. The implementation of the algorithms is based on Apache Spark platform in Python programming language (Pyspark). The result shows that although the alternating least squares (ALS) algorithm beats the weighted bipartite graph projection (WBGP) in terms of computational time, the latter can produce better prediction accuracy.

Introduction

Recommender systems have become quite popular in recent years since e-commerce companies rose. They to a great extent help costumers discover what they need or are interested in and in the meantime improve the user experiences. At present, recommender systems have been applied almost anywhere, not only by e-commerce platforms like Amazon and Netflix but also by social platforms like Facebook and Instagram. Since the data for these companies are usually enormous, the systems are often required to be scalable to a huge amount of data and computations.

Generally speaking, there are two main strategies in recommender systems: content filtering and collaborative filtering [3]. The basic idea of content filtering is to construct a profile of a user or an item (products, businesses, movies etc.) with external information like the gender or salary of users and the genre or price of items. Content filtering believes that the information of these qualities can help capture the nature of users and items and therefore improve the prediction on users' preferences. In contrast with it, collaborative filtering concentrates only on past user behaviors such as transactions and ratings. It studies the relationship between users and items to predict new user-item associations without any external information.

This paper focuses on comparison between two collaborative filtering approaches in two different areas: neighborhood methods and latent factor models [3]. Neighborhood methods aim to find the similarities between users or items while latent factors models attempt to explain the transactions or ratings by some latent factors (the number of latent factors is usually far smaller than the number of all users or items). For neighborhood methods I apply the weighted bipartite graph projection (WBGP) approach proposed by [10] with similarity measure defined in [6] [7]; for latent factors model methods I employ the matrix factorization with alternating least squares (ALS) learning algorithm [3].

Data

The collaborative filtering approaches are implemented on Yelp Open Dataset (https://www.yelp.com/dataset). Yelp.com is a crowd-sourced local business review and social networking site, and its open dataset mainly contains information of businesses, users and users' reviews on businesses from 8 metropolitan areas in Canada and United States. Due to the limitation of cluster resources and run time, I only select businesses from Washington state and eliminate those have fewer than 15 reviews for reducing the sparseness of user-business matrix similar to [8]. The summary of data this paper uses is shown in Table 1.

Number of Businesses	1589
Number of Users	44780
Number of Reviews	102383
Average Number of Reviews for User	2.2863555158552926
Average Number of Reviews for Business	64.43234738829453

Table 1: Summary of Data

Methodology

1 Weighted Bipartite Graph Projection

1.1 Background and Concept

Weight Bipartite Graph Projection (WBGP) approach is in the domain of neighborhood methods which focus on evaluating the similarities between users or items. These methods can be generally classified into two types: user-oriented and item-oriented. User-oriented methods learn similarities between users so they estimate unknown ratings based on recorded ratings of like-minded users; item-oriented methods learn similarities between items so they make predictions based on recorded ratings made by the same user on similar items. WBGP approach can be implemented from both perspectives. This paper applies the item-oriented (business-oriented) WBGP method on account of its better scalability and improved accuracy in many cases [2]. Note that user-oriented method is mathematically equivalent by just switching the roles of item and user.

A bipartite graph is a graph where the set of vertices are partitioned in two disjoint sets X and Y such that each edge of the graph connects two vertices only in different sets (as illustrated in Figure 1(a)). The projection of bipartite graph onto one set, for example X, is a graph whose nodes only come from the set X and they are connected with each other when they have at least one common neighboring Y node in the original bipartite graph. If we additionally assign some "weights" to each edge in the projection, we then get the weighted bipartite graph projection (as illustrated in Figure 1(b) and Figure 1(c)). The simplest weight assigned to each edge is the number of common neighboring Y nodes the two vertices have which is exactly the case in Figure 1. Beyond that we can also assign similarity measures which are more complicated in order to capture the relationship between each pair of nodes from the original graph. There are some popular similarity functions usually applied in neighborhood approaches such as Pearson correlation similarity, Cosine similarity and Tanimoto coefficient [4]. The similarity measure applied in this paper is defined in [6] [7]as a measure of recommendation power.

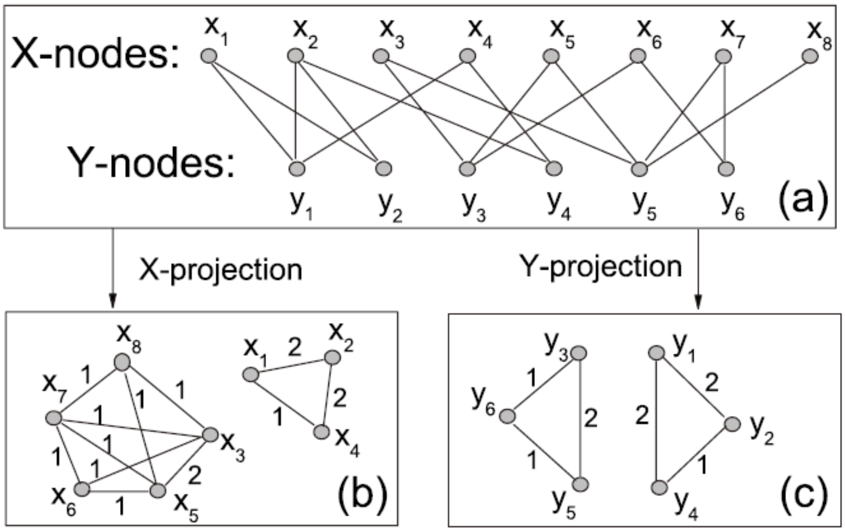

Figure 1: Diagram of Weighted Bipartite Graph Projection

1.2 Similarity Measure

As mentioned above, similarity measures can be thought of as the weights assigned to each edge in a WBGP. However, the WBGP we talked about in the previous section is undirected, meaning the similarity or weight between two nodes is the same. This situation is suitable for symmetric similarity measures like Pearson correlation similarity and Cosine similarity because if we switch the role of the two nodes the similarity remains unchanged. Now we consider a directed WBGP that doubles the edges of the undirected one since there are two edges between each pair of nodes. With this directed graph projection, we need to apply an asymmetric similarity measure (i.e. $sim(i,j) = sim(j,i)$ does not always hold) for each edge. The recommendation power measure can capture the asymmetric features in any pairs of businesses i, j and is defined as

$$rp\left(i,j\right) = \sum_{u \in U} \frac{r_{i,u} r_{j,u}}{R_i R_u} \tag{1}$$

where U is the set of all users who have rated both i and j, $r_{i,u}$ is the rating that user u gives to business i, R_i is the sum of ratings business i has received and R_u is the sum of ratings user u has given.

This measure is obviously asymmetric due to the existence of R_i. Actually, the recommendation power measure represents how likely business j is to appeal to a particular user provided that he or she has been to business i and rated it according to [6], hence it gives a reasonable explanation for why this measure is asymmetric.

1.3 Algorithm

First I build a bipartite graph (one set for users and the other for businesses) with weights as ratings using the Yelp review data. Next I calculate the recommendation power for all pairs of businesses according to equation (1) and build the weighted bipartite graph projection on the business set. Finally, the prediction of rating from user u to business i can be computed by the following formula which is similar to the one in [6] [1](simply transfer from user-oriented aspect to business-oriented aspect):

$$\hat{r}_{i,u} = \bar{r_i} + \sum_{j \in B_u} rp\left(i,j\right)\left(r_{j,u} - \bar{r_j}\right). \tag{2}$$

Here $\bar{r_i}$ represents the average rating received by business i and B_u is the set of businesses the user u has ever rated. Notice that in the extremely sparse situation where the recommendation power measures from business i to any other businesses are all 0, the predicted rating of business i given by any new user u simply equals to the average rating of business i.

2 Matrix Factorization with Alternating Least Squares

2.1 Background and Concept

Matrix factorization is one of the widely successful latent factor models which characterizes both items and users by vectors of factors inferred from item rating patterns. Imagine a user-item matrix $M_{m\times n}$ with some known entries representing the feedback or preference a user giving to an item, for instance ratings, the task of collaborative filtering is to fill in the missing entries of the matrix with predictions. Matrix factorization solves this problem by mapping both users and items to a joint latent factor space of dimensionality k (usually much smaller than the number of users or items), such that user-item interactions are modeled as inner products in that space [3]. We can express the idea mathematically as

$$\hat{M_{i,j}} = u_i^T v_j \qquad where \;\; u_i \in R^k \; and \; v_j \in R^k \tag{3}$$

where $\hat{M_{i,j}}$ is the predicted rating given by user i to item j, u_i is a parameter vector associated with user i and v_j is a parameter vector associated with item j. More generally, we can get the matrix expression:

$$\hat{M} = U^T V \qquad where \; U = [u_1, ..., u_n] \; and \; V = [v_1, ..., v_m] \, . \tag{4}$$

Here n and m are the numbers of users and items respectively, and $\bar{M}$ represents the whole predicted user-item matrix.

To achieve the parameter vectors for each user and item, we consider minimize the following loss function:

$$f(U,V) = \sum_{(i,j)\in\Omega} \left(M_{i,j} - u_i^T v_j\right)^2 + \lambda \left(\sum_{i=1}^{n} \|u_i\|^2 + \sum_{j=1}^{m} \|v_j\|^2 \right). \tag{5}$$

Here Ω is the set of (i,j) pairs for which $M_{i,j}$ is known and $\lambda \geq 0$ is a regularization hyper-parameter. This loss function can be derived using the Bayesian model [9].

2.2 Learning Algorithm

Minimizing the loss function above is a non-convex optimization problem, so traditional gradient descent algorithm can only converge to local minima and might also be very slow. Therefore, this paper applies alternating least squares (ALS) algorithm which performs better in many cases.

The main idea of ALS is to fix one of the unknowns u_i and v_j in the loss function and optimize the other by solving a least squares problem. The algorithm rotates between fixing u_i's and fixing v_j's utill some termination criteria are met. Since each phase corresponds to solving a convex optimization problem, it ensures that each phase decreases the loss function until convergence [3].

2.3 Hyperparameters Selection

In matrix factorization with ALS algorithm we have three hyperparameters to choose: the number of latent factors K, the max number of iterations I (termination criterion) and the regularization parameter λ. First I set a fixed grid for K and I, and try different values for λ (0.1, 0.5 and 1). The models are trained on the training data and the root mean squared errors (RMSE) are computed using the test data. The results for different λ values are shown respectively in Figure 2, Figure 3 and Figure 4. The figures indicate that the models with λ equaling to 0.5 and K equaling to 3 perform better in general. The result also shows that RMSE decreases as I increases, hence I move the grid of I forward to see if more iterations can result in better accuracy. However, Figure 5 (in Appendix) displays that the RMSE decreases very slowly after 10 iterations, meaning the algorithm gets extremely close to the optimum value so it makes little sense to infinitely increase I. Finally, I select the model with $K = 3$, $I = 14$ and $\lambda = 0.5$ as the best one.

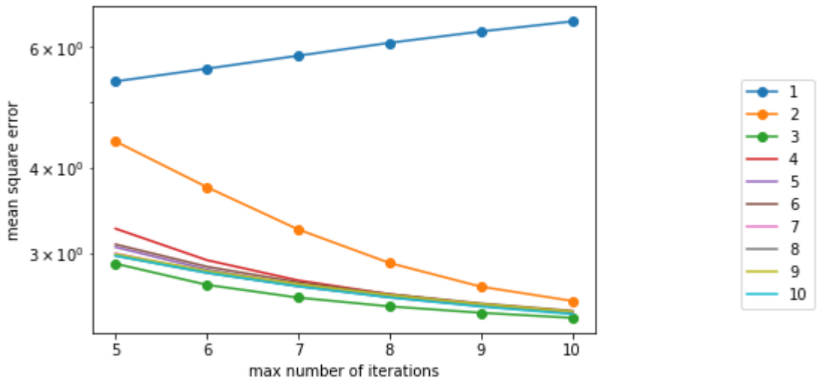

Figure 2: Test RMSE for Models with $\lambda = 0.1$

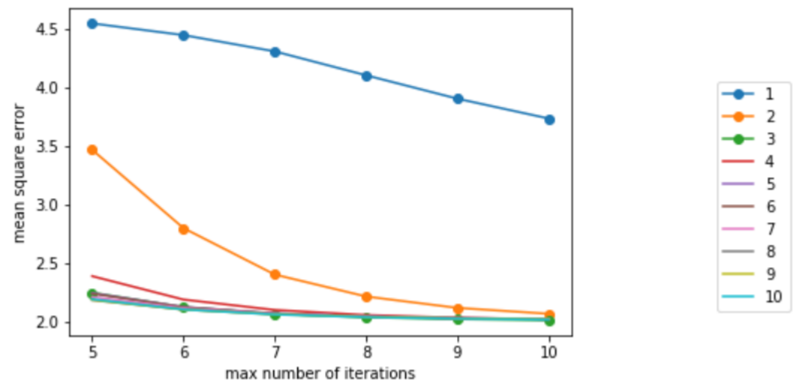

Figure 3: Test RMSE for Models with $\lambda = 0.5$

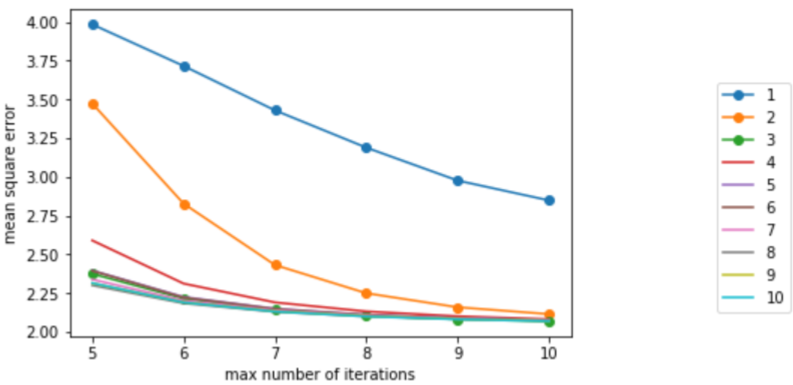

Figure 4: Test RMSE for Models with $\lambda = 1$

Apache Spark

In collaborative filtering based on neighborhood approaches, it is often quite expensive to calculate the similarities between users or items because the algorithms always need to search the entire dataset to find the neighbor of a particular user or item. According to [4], the time complexity in the worst case can achieve $O(UI)$ (U and I are numbers of users and items respectively). In order to deal with the scalability problem, I choose Apache Spark which is a unified analytics engine for big data processing to implement the collaborative filtering algorithms in this paper. It can parallelize the algorithms using distributed systems and thus largely improve the computational efficiency. Moreover, the work of Kupisz and Unold [4] also proved that the system based on Apache Spark is more efficient than Hadoop with respect to the item-based collaborative filtering algorithm. In this paper, the collaborative filtering algorithms for the Apache Spark platform are implemented in the Python programming language (Pyspark).

For implementation of the WBGP algorithm I mainly use the Dataframe APIs and some APIs in the GrahpFrame package like aggregateMessages and for implementation of the ALS algorithm I directly employ APIs in the pyspark.ml package like ALS.

Result

To compare the prediction accuracy of the two algorithms, I randomly split the Yelp data into two sets: one for training (80%) and one for test (20%). The two algorithms are trained on the training set and the RMSE is calculated with respect to the test set. The whole process is repeated for 5 times and we get the mean and standard error of RMSE for both algorithms. The summarized result is shown in Table 2.

Surprisingly, the WBGP approach performs not only better but also more stable than the ALS method. Besides, the mean of RMSE for the WBGP approach is quite similar to the result in [6].

	WBGP	ALS
Mean of RMSE	1.4032339302172345	2.256193268649647
Standard Error of RMSE	0.0048709873291408355	0.018367415714108097

Table 2: Summarized Result of RMSE

Conclusion

As two completely different collaborative filtering approaches, the WBGP and ALS algorithms have their own advantages and disadvantages. In terms of prediction accuracy, the WBGP method outperforms the ALS method with its average RMSE equaling to about 1.4. In addition, the standard error of RMSE with respect to WBGP is also smaller than ALS, indicating its superior stability. However, in terms of computational time, the ALS algorithm has great advantage over the WBGP algorithm since it has smaller time complexity [9] than WBGP. Actually, when I implement the two algorithms on Pyspark, the WBGP algorithm takes more than 10 times as long as the ALS algorithm takes.

Reflection and Further Research

On account of the large time complexity of the WBGP algorithm and the limitation of cluster resources, the two approaches in this paper are only implemented on a small subset of the entire Yelp review dataset, which cannot make full use of the advantages brought by distributed computing. Larger datasets could be tried in further research.

In collaborative filtering tasks we assume that one user can only have one rating on a particular item or business. However, it is usually not the case in the real-world data since users absolutely have rights to express his or her preference for an item several times. This problem also appears in the Yelp review dataset and what I do is dropping the redundant ratings for each user-item pair randomly. I think maybe better ways are to calculate the mean value of the ratings and regard it as the overall preference of the user to that item, or take the rating the user gave to that item most recently as the latest feedback. Perhaps in future papers these methods can be taken into consideration.

In the implementation of the WBGP algorithm, ideally the recommendation power should be calculated directly from the bipartite graph by some graph algorithms like random walk in [6], but unfortunately I could not find any related APIs in the GraphFrames package in Pyspark so I calculate the recommendation power using some Dataframe APIs on the original data instead. However, I notice that Neo4j which is a graph database platform has an implementation of the random walk algorithm [5]. Further research could try implement the WBGP algorithm using Neo4j and see if it is more efficient than Apache Spark.

References

[1] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.

[2] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[4] Bartosz Kupisz and Olgierd Unold. Collaborative filtering recommendation algorithm based on hadoop and spark. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 1510–1514. IEEE, 2015.

[5] Mark Needham and Amy E Hodler. *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*. O'Reilly Media, 2019.

[6] Sumedh Sawant. Collaborative filtering using weighted bipartite graph projection: a recommendation system for yelp. In *Proceedings of the CS224W: Social and information network analysis conference*, volume 33, 2013.

[7] Ming-Sheng Shang, Yan Fu, and Duan-Bin Chen. Personal recommendation using weighted bipartite graph projection. In *2008 International Conference on Apperceiving Computing and Intelligence Analysis*, pages 198–202. IEEE, 2008.

[8] Jason Ting and Swaroop Indra Ramaswamy. Yelp recommendation system. 2013.

[9] Milan Vojnovic. Lse st446 lecture notes week9, 2021.

[10] Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Physical review E*, 76(4):046115, 2007.

Appendix

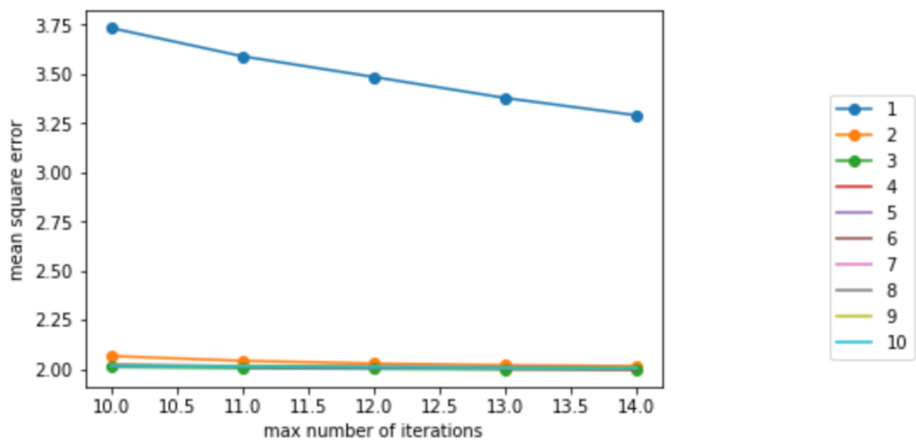

Figure 5: Test RMSE for Models with $I = 10, 11, 12, 13, 14$ and $\lambda = 0.5$