A Survey of Collaborative Filtering Recommender Algorithms and Their Evaluation Metrics

Mahdi Jalili
School of Engineering, RMIT University, Melbourne, Australia
Mahdi.Jalili@rmit.edu.au

Abstract—**Recommender systems are often used to provide useful recommendations for users. They use previous history of the users-items interactions, e.g. purchase history and/or users rating on items, to provide a suitable recommendation list for any target user. They may also use contextual information available about items and users. Collaborative filtering algorithm and its variants are the most successful recommendation algorithms that have been applied to many applications. Collaborative filtering method works by first finding the most similar users (or items) for a target user (or items), and then building the recommendation lists. There is no unique evaluation metric to assess the performance of recommendations systems, and one often choose the one most appropriate for the application in hand. This paper compares the performance of a number of well-known collaborative filtering algorithms on movie recommendation. To this end, a number of performance criteria are used to test the algorithms. The algorithms are ranked for each evaluation metric and a rank aggregation method is used to determine the wining algorithm. Our experiments show that the probabilistic matrix factorization has the top performance in this dataset, followed by item-based and user-based collaborative filtering. Non-negative matrix factorization and Slope 1 has the worst performance among the considered algorithms.**

Keywords—Social networks analysis and mining, big data, recommender systems, collaborative filtering.

I. Introduction

The volume of information has grown exponentially in recent years. In various applications, users rate (or purchase) some items. Such user-item interaction data along with their contextual information can be further used to provide efficient item recommendation for any target user [1-4]. Recommender systems (RSs)are now inevitable part of the world-wide-web and many service-oriented applications. RSs methods can be classified into three categories: Content-based, Collaborative Filtering (CF) and hybrid approaches. For many applications the contextual information of users and/or items are not available and CF methods are the only choices. CF recommender algorithms have been extensively applied in both academia and industry [5-10], and are divided into two classes: memory- and model-based.In memory-based approaches the user-item matrix is used to find similar users (or items) and generate a prediction. While in model-based CF approaches, first a model is constructed in an offline learning phase, and then the model is applied online to generate recommendation.

Despite all the progress on RS, they encounter several challengesand need further improvement. There is a need for algorithms with better quality of recommendation, e.g. algorithms exploiting more details such as contextual information (e.g. time and location), algorithms with more flexible results or recommenders utilizing multi-criteria ratings. Moreover, there is a need for scalable algorithms which are applicable in real-world situations with large-scale datasets. As more RS algorithms are developed, it is necessary to put efforts on developing proper evaluation methods. Indeed, evaluating performance of RSs is one of the challenging tasks within the community. There is no a single metric that can efficiently measure the performance and often one has to use a number of them at the same time. In this manuscript, we first review a number of well-known CF recommender algorithms, and then compare their performance on various evaluation metrics. Finally, we rank the algorithms based on their performance.

II. Recommender Algorithms

CF recommendation algorithms can be divided into two categories: memory- and model-based [6]. Memory-based methods use rating (or purchase) histories (user-item interaction data) in order to provide efficient recommendations. Examples include item- and user-based CF algorithms [7, 10]. Model-based algorithms (such as those based on matrix factorization using singular value decomposition [9]) build a model and use a training method to find the optimal values of the missing parameters. Let's consider a set of users $U = \{u_1, u_2, u_3, \dots, u_m\}$ and a set of items $I = \{i_1, i_2, i_3, \dots, i_n\}$. r_{ui} represents the rating given by user u to item i, and usually has a numerical scale (this is a binary value for purchase history). Indeed, the user-item interaction data can be modelled as a bipartite network with links from users to items. Often, this network is converted to a simpler version with only *like* and *dislike* values (1 for like, -1 for dislike and 0 for no interaction). In the following, we briefly review a number of well-known memory- and model-based collaborative filtering methods.

A. Memory-based CF

Memory-based CF methods use the rating histories to first find the most similar users (or items) for a target user (or item). Then, the recommendation list for any target user is built by looking at the items positively rated by his/her most similar users. The underlying assumption in these methods is that if two users have similar history of ratings for common items, they will likely have similar preferences for the rest of items. As for the two items rated similarly by several users, they would probably be rated in the same manner by the rest of users. There are a number of methods to calculate the similarity between two users (or items). Here, we use Pearson similarity, which is calculated as the following for any pair of users$u$ and u'

IJSMS

$$S(\mathrm{u},\mathrm{u}') = PC_{uu'} = \frac{\sum_{i\in I_{uu'}} (r_{ui}-\overline{r}_u)\times(r_{u'i}-\overline{r}_{u'})}{\sqrt{\sum_{i\in I_{uu'}} (r_{ui}-\overline{r}_u)^2}\sqrt{\sum_{i\in I_{uu'}} (r_{u'i}-\overline{r}_{u'})^2}}, \quad (1)$$

where $\overline{r_u}$ is the average rating of user u.

As the similarity values are computed, one can use top-k similar users to a target user in the prediction algorithm[11]. Here we set k = 30. For any target user$u$, one should use the following formula to predict his/her ratings on item i

$$\tilde{r}_{ui} = \overline{r}_{u'} + \frac{\sum_{u'\in NS_u} (r_{u'i})\times(\mathrm{S}(\mathrm{u},\mathrm{u}'))}{\sum_{u'\in NS_u} |(\mathrm{S}(\mathrm{u},\mathrm{u}'))|}, \quad (2)$$

where $NS_u$is the neighbor set of target user $u$with k members and $\overline{r_{u'}}$is the average ratings by the u'. This method is referred to as user-based CF.

User-based CF suffers from scalability issues and item-based method was proposed to handle this issue [10]. First, the similarity values between any item pair is calculated using equation (1), and then the ratings are predicted as

$$\tilde{r}_{ui} = \frac{\sum_{i'\in NS_i} (r_{ui'})\times(\mathrm{S}(\mathrm{i},\mathrm{i}'))}{\sum_{i'\in NS_i} |(\mathrm{S}(\mathrm{i},\mathrm{i}'))|}, \quad (3)$$

where NS_i is the neighborhood set of item i. Item-based CF is scalable for larger data setssince the item-neighborhood is rather static, and therefore, the similarities can be efficiently pre-computed. Thus, the online (prediction phase) performance is better than user-based CF.

Recently, a method based on Resource Allocation (RA) has been proposed to improve performance of classical CF algorithms [7].RA technique has been frequently used for link prediction in complex networks [12]. RS-based CF indeed assumes that more popular items should be less informative for the recommendation task. The only difference between the RA-based method and user-based CF is the way the similarity values are calculated. In RA-based recommendation, the similarity scores are calculated as

$$S_{CF-RA}(\mathrm{u},\mathrm{u}') = PC_{uu'}\times \sum_{i\in\Gamma(u)\cap\Gamma(u')} \frac{1}{d_i}, \quad (4)$$

where d_i is degree of item $i$and $\Gamma(u)$ is the set of items rated by u.

B. Model-based CF

Model-based CF algorithms consider a specific model and use proper methods to learn the missing parameters of the model. Examples of such RS methods include Slope1[13] and latent factor models such as Matrix Factorization (MF) and Singular Value Decomposition (SVD)[9, 14, 15]. Here we consider a number of well-known model-based CF algorithms and compare their performance.

Lemire and Maclachlan proposed Slope1CF algorithm[13]. For each pair of items, this algorithm calculates how much an item is more preferred than the other one, which is then used to predict thetarget user's rating of one of those items, given their rating of the other. Given two rating vectors v_i and w_i, $i = \{1,2,\dots,n\}$, Slope1 searches for the best predictor of the form $f(x) = x + b$ to predict w from v by minimizing $\sum_i(v_i + b - w_i)^2$. Therefore, b must be chosen to be the average difference between the two vectors. The average deviation of item i with respect to item j is calculated as

$$dev_{ji} = \sum_{u\in S_{ji}(\mathrm{trainSet})} \frac{r_{uj}-r_{ui}}{card(S_{ji}(\mathrm{trainSet}))}, \quad (5)$$

where $u \in S_{ji}(trainingSet)$is the set of users who have rated both items$i$ and j in the training set and$card(S)$ is the number of elements in set S. Then, the unseen ratings are predicted as

$$\tilde{r}_{uj} = \frac{1}{card(R_j)} \sum_{i\in R_j} (dev_{ji} + r_{ui}), \quad (6)$$

where $R_j = \{i | i \in S(u), i \neq j, card(S_{ji}(trainSet)) > 0)\}$ is the set of all relevant items and $S(u)$ is the set of items rated by user u.

MF methods such as SVD have been proposed to better handle scalability and sparsity issues of memory-based CF algorithms.TheSVD techniqueis used to first detect the latent user-item relationships, and then to generate a low-dimensional representation of the original rating matrix space to calculate the neighborhood in the reduced space [14]. SVDfactorizes the rating matrix into a product of two low-rank matrices. It produces a set of uncorrelated eigenvectors which represents the users and the items. For a rating matrix M with $m \times i$ dimensions and rank r, SVD is calculated as

$$SVD(\mathrm{M}) = U \times S \times V^T, \quad (7)$$

where U and V are two orthogonal matrices, S is a diagonal matrix – denoted as "singular matrix" that has r nonzero diagonal elements.The first r columns of U and V represent the orthogonal eigenvectors associated with the r nonzero eigenvalues of $MM^T$and M^TM, respectively.U and V are called the *left* and *right* singular vectors, respectively. One can keep only $k$of$r$singular values and discardlower entries. $(r - k)$ columns from $U$and $(r - k)$ rows from$V^T$are eliminated to produce U_k and V^T_k matrices. U_k and $V_k$are multiplied together using S_k to produce M_k. The reconstructed matrixM_k is the closest rank-k matrix to M, with respect to the Frobenius norm of matrix. Indeed,

$$M_k = U_k \times S_k \times V_k^T. \quad (8)$$

Now, the rating values are predicted as

$$\tilde{r}_{ui} = \overline{r}_u + U_k\sqrt{S_k^T(\mathrm{u})}\times\sqrt{S_k}\times V_k^T(\mathrm{i}). \quad (9)$$

Regularized SVD (RegSVD)has been proposed to improve SVD using a learning rate, regularization constants and a method for clipping predictions [15], which minimizes the squared error between the actual ratings and predicted estimations for all available votes.In the Non-negative MF (NMF) algorithm, there is a low-dimensional linear model with a non-negative constraint to indicate the rating matrix. This method uses multiplicative update rules for minimizing the

least squares error between the actual ratings and the predicted ones.Probabilistic MF (PMF) utilizes a probabilistic linear model to represent the latent features of users and items[16]. This model linearly scales with the number of ratings.

III. Evaluation Metrics

In order to assess performance of RS algorithms, one can use a number of metrics [2, 6]. Each metric evaluate the performance from certain perspective. Evaluation metrics can be classified into three categories: those based on accuracy and precision, based on novelty and diversity and rank-based metrics.

C. Accuracy metrics

Root Mean Square Error (RMSE) of a prediction algorithm can be obtained as,

$$RMSE = \frac{1}{|\text{TestSet}|} \sum_{(u,i)\in \text{TestSet}} \sqrt{(\tilde{r}_{ui} - r_{ui})^2}, \quad (10)$$

and the normalized RMSE as

$$NRMSE = \frac{RMSE}{Rating_{\max} - Rating_{\min}}. \quad (11)$$

An alternative approach to evaluate accuracy of algorithms is their precision and recall. Precision P is obtained by dividing the True Positives (TP) by the summation and TP and False Positives (FP). Recall R is obtained by dividing TP by the summation of TP and False Negatives (FN). One often needs to know each item is relevant or not, which means every item must be rated by the user. To this end, often *P@N* and *R@N* instead (*N* being the size of the recommendation list) is used [17].

D. Rank-based metrics

These evaluation metrics consider the rank of the recommendation list instead of the exact value of the predicted ratings. Half-life utility metric suppose that as the rank of the item in the recommendation list decreases, the probability of user's tendency to examine it reduces exponentially [18]. Half-life utility is defined by

$$H_u = \sum_{i=1}^{N} \frac{\max(r_{ui} - d, 0)}{2^{(r_i - 1)/(h-1)}}, \quad (12)$$

The half-life utility index H can be obtained by averaging over all users.

Discounted Cumulative Gain (DCG) evaluates the usefulness of an item based on its rank in a recommended list, and is defined as [19]

$$DCG_b_u = \sum_{i=1}^{b} R_i + \sum_{i=b+1}^{N} \frac{R_i}{\log_b r_i}. \quad (13)$$

where r_i is the rank of item *i*in the recommendation list. The lower is the rank of the item in the recommendation list (i.e., toward the end of the list), its share in the cumulated gain becomes less.If the item is relevant, we have $R_i = 1$, otherwise $R_i = 0$.Different values of *b*, as the base of the logarithm, control the degree of reduction in items' shares in DCG; i.e., the greater the values of *b*, the slower the shares decrease. For the item at rank 1 and smaller than *b*, it is logical to use the first part of the above equation, rather than using the logarithmic discount function. According to [19], we set $b = 2$ for the experiments.The Normalized DCGis obtained through dividing by the maximum possible gain:

$$NDCG_u = \frac{DCG_u}{DCG_{\max}}. \quad (14)$$

The above values are averaged over all users.

Similar to DCG, the Rank-biased Precision (RBP) metric attempts to evaluate the recommendation list through giving more shares to highly ranked relevant items for each user. RBP for each user's recommended list is defined by[20]

$$RBP_u = (1-\text{p})\sum_{i=1}^{N} (R_i \times p^{r_i - 1}), \quad (15)$$

wherethe underlying assumption is that users often examine the first item of the recommended list, and then check the next item with probability *p*. The overall RBP averaged for all users.

E. Novelty and diversity metrics

In many cases, the users would like to be recommended diverse list of items (i.e., those with high novelty), as they are likely to be aware of the popular ones[21-23]. Self-Information Based Novelty (SIBN) has been proposed to measure novelty of recommendation list, and is calculated as

$$SIBN_i = \log_2(\frac{m}{d_i}). \quad (16)$$

The average SIBN can be simply obtained by making the average of the above relation over all users. A good recommender not only recommends likeable items to users, but also covers a wide variety of items which the system has to offer. In its simplest form, coverage indicates that as the percentage of the items a recommender can suggest grow higher, the system is performing better. Catalog coverage refers to the percentage of distinct items in the top-*N* recommendation lists of users

$$C = \frac{I_r}{n}, \quad (17)$$

where *n* is the total number of items in the system and $I_r$indicates the total number of distinct items in users' top-*N* lists.An improvement to this definition is to consider only the items which are relevant to a user.

IV. Results

In order to assess the performance of RS algorithms on the above evaluation metrics, we useMovielens dataset as one of the most frequently used datasets within the community of RSs. It contains 943 users, 1682 items and about 100,000 ratings from users to items, representing rating density of approximately 6%. Table 1 shows the values of evaluation metrics for different algorithms. As it is seen, not a single algorithm is the best in all evaluation metrics, which makes it difficult to choose a wining algorithm. One can choose an algorithm based on the application in hand. For example, if high precision is required, item-based CF and PMF are the best

choices among these algorithms. Table 2 shows the ranking of the algorithms and the final aggregate rank using MedRank aggregation method. This indicates that PMF is the top-performer algorithm followed by the item- and user-based CF.

Table 1: Performance of CF algorithms on different evaluation metrics.

Metrics /algorithms	NRMSE	P@10	R@10	H	N	C	NDCG	RBP
User-Based	**0.20**	0.009	0.010	0.75	8.2	0.17	**0.82**	0.008
Item-Based	**0.20**	**0.04**	0.028	0.77	5.4	0.19	**0.82**	0.029
RA	0.21	0.012	0.012	**0.88**	7.1	**0.24**	0.81	0.009
Slope1	0.24	0.0001	0.0001	0.53	**9.3**	0.08	0.81	0.000
Reg-SVD	0.23	0.03	0.023	0.60	7.0	0.20	**0.82**	0.021
NMF	0.25	0.001	0.0001	0.48	9.1	0.04	0.81	0.000
PMF	0.24	**0.04**	**0.034**	0.75	5.4	0.14	0.81	**0.032**

Table 1: Ranking of the algorithms for different evaluation metrics. The last column shows the aggregated ranking using MedRank aggregation approach.

Metrics /algorithms	NRMS	P@10	R@10	H	N	C	NDCG	RBP	Final Aggregated Rank
User-Based	1	6	5	3	3	4	1	5	3
Item-Based	1	1	2	2	6	3	1	2	2
RA	3	4	4	1	4	1	2	4	4
Slope1	6	7	6	6	1	6	2	6	6
Reg-SVD	4	3	3	5	5	2	1	3	5
NMF	7	5	6	7	2	7	2	6	7
PMF	5	1	1	3	6	5	2	1	1

V. Conclusions

Recommendation Systems (RSs) have many applications in science and engineering. With ever-increasing volume of big data, we have often data overload problem. One can use RSs algorithms to mine user-item interaction data (such as rating (or purchase) history of users on items) and provide useful recommendation lists for any target user. In this paper we reviewed a number of Collaborative Filtering (CF) recommender algorithms from the view of various evaluation metrics, assessing the algorithms in terms of their accuracy, rank-based accuracy and novelty/diversity of recommendations. The performance of the algorithms was tested in Movilense dataset, which has been frequently used a benchmark in RSs.

Acknowledgment

This work was supported by Australian Council Research through project No. DE140100620. The author would like to thank MalihehIzadi for her help in the implementation of the algorithms.

References

[1] G. Adomavicius and A. Tuzhilin, "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,* vol. 17, pp. 734-749, 2005.

[2] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based System,* vol. 46, pp. 109-132, 2013.

[3] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Physics Reports,* vol. 519, 2012.

[4] A. Javari and M. Jalili, "Accurate and novel recommendations: an algorithm based on popularity forecasting," *ACM Transactions on Intelligent Systems and Technology,* vol. 5, p. 56, 2014.

[5] Y. Ding and X. Li, "Time weight collaborative filtering," in *International Conference on Information and Knowledge Management.*, Bremen, Germany, 2005, pp. 485-492.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems,* vol. 22, pp. 5-53, 2004.

[7] A. Javari, J. Gharibshah, and M. Jalili, "Recommender systems based on collaborative filtering and resource allocation," *Social Network Analysis and Mining,* vol. 4, p. 234, 2014.

[8] H. Polat and W. Du, "SVD-based collaborative filtering with privacy," presented at the ACM symposium on Applied computing, 2005.

[9] M. Ranjbar, P. Moradi, M. Azami, and M. Jalili, "An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems," *Engineering Applications of Artificial Intelligence,* vol. 46, pp. 58-66, 2015.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," presented at the 10th international conference on World Wide Web, 2001.

[11] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl, "GroupLens: applying collaborative filtering in usenet news," *Communications of the ACM,* vol. 40, pp. 77-87, 1997.

[12] L. L. Tao Zhou, Yi-Cheng Zhang, "Predicting Missing Links via Local Information," *Europe Physics,* vol. 71, pp. 623-630, 2009.

[13] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *Society for Industrial Mathematics,* vol. 5, pp. 471-480, 2005.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," DTIC Document2000.

[15] B. Webb, "Netflix Update: Try This at Home," 2006.

[16] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2007, pp. 1257-1264.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Proceedings of the 2nd ACM conference on Electronic commerce*, 2000, pp. 158-167.

[18] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1998, pp. 43-52.

[19] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems (TOIS),* vol. 20, pp. 422-446, 2002.

[20] A. Moffat and J. Zobel, "Rank-biased precision for measurement of retrieval effectiveness," *ACM Transactions on Information Systems (TOIS),* vol. 27, p. 2, 2008.

[21] A. Javari, M. Izadi, and M. Jalili, "Recommender systems for social networks analysis and mining: precision versus diversity," in *Complex Systems and Networks*, J. Lü, X. Yu, G. Chen, and W. Yu, Eds., ed: Springer, 2016, pp. 423-438.

[22] A. Javari and M. Jalili, "A probabilistic model to resolve diversity-accuracy challenge of recommendation systems," *Knowledge and Information Systems,* vol. 44, pp. 609-627, 2015.

[23] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *Proceedings of the National Academy of Science of the United States of America,* vol. 107, pp. 4511-4515, 2010.

IJSMS