Grouping Like-Minded Users for Ratings' Prediction

Soufiene Jaffali, Salma Jamoussi, Abdelmajid Ben Hamadou, Kamel Smaili

Grouping Like-Minded Users for Ratings' Prediction

Soufiene Jaffali[1], Salma Jamoussi[1], Abdelmajid Ben Hamadou[1], and Kamel Smaili[2]

[1] MIRACL Laboratory
Higher Institute of Computer Science and Multimedia
University of Sfax, BP 1030, Sfax, Tunisia
{jaffali.soufiene, jamoussi, abdelmajid.benhamadou}@gmail.com
[2] Campus Scientifique LORIA , Nancy, France,
smaili@loria.fr

Abstract. Regarding the huge amount of products, sites, information, etc., finding the appropriate need of a user is a very important task. Recommendation Systems (RS) guide users in a personalized way to objects of interest within a large space of possible options. This paper presents an algorithm for recommending movies. We break the recommendation task into two steps: (1) Grouping Like-Minded users, and (2) create model for each group to predict user-movie ratings. In the first step we use the Principal Component Analysis to retrieve latent groups of similar users. In the second step, we employ three different regression algorithms to build models and predict ratings. We evaluate our results against the SVD++ algorithm and validate the results by employing the MAE and RMSE measures. The obtained results show that the algorithm presented gives an improvement in the MAE of about 0.42 and 0.5201 in the RMSE.

Keywords: Rating prediction, Social recommendation, Grouping Like-Minded users

1 Introduction

Given the huge amount of products to purchase or information to browse on the web, matching users with the most appropriate items becomes a very important task. Recently, the interest to the Recommendation Systems (RS) has increased considerably. Indeed, RS play an important role in the well noted web sites such as Netflix[3], Tripadvisor[4], etc. The RS aim to provide suggestions for items to be of use to a user. An item refers to any object, it can be a music to listen, product to buy, film to watch, etc. In this scope, many algorithms and methods are proposed. Burke [3] distinguishes six families of RS approaches (Content-Based, Collaborative Filtering, Demographic, Knowledge-Based, Community-Based and Hybrid Recommendation Systems). As one of the most representative

[3] https://www.netflix.com/
[4] https://www.tripadvisor.com/

categories, collaborative filtering (CF) is popular for its high performance and simple requirements. The principle of CF is to create patterns based on user preferences and use them to produce suggestions.

One of the emergent tasks in RS field is *Movie Recommendation*. The Netflix and MovieLens are the most popular *Movie Recommendation* tasks [15, 18]. Those tasks attract many researchers and many algorithms are proposed in this topic. Most of the CF algorithms used for *Movie Recommendation*, handle the problem as a *Partial Matrix Factorization*, and aim to find the missing values in the matrix $users \times movies$ containing user-movie ratings [14]. The major problem with these solutions is the sparsity of data (given the huge number of users and movies, in one hand and the lack of stored users-movies ratings, in other hand). Furthermore, we cannot predict recommendations for a new movie or user that does not appear in the training data.

We propose a new algorithm to recommend movies based on the Like-Minded user groups (groups of users having similar cinimatographic tastes). The main idea of our algorithm is to infer user u rating on a movie m based on the stored ratings of the user u and users having similar interests. Such prediction can improve the recommendation accuracy and decrease the complexity of the prediction model, as it builds patterns for groups of homogeneous users. To retrieve Like-Minded users, we use the Principal Components Analysis (PCA). Thus, we reduce data dimensions and solve the sparsity problem that characterizes RS data. Also, the Like-Minded user groups are generated based on the tastes of users (genres of rated movies), without any knowledge about the social network relation between users. Thus, we remedy the limitation of *community-Based RS*.

The remaining of this paper is organized as follows. Section 2 gives an overview of the related work. In Section 3, we present our approach to estimate ratings. Then, we describe the data set and the baseline in Section 4. The experimental results are presented in Section 5, and we conclude in Section 6 by pointing out some future works.

2 Related Work

In this section, we review a small set of Collaborative Filtering (CF) researches. We can find two major groups of CF methods, namely: *Neighborhood* and *Latent-Model-Based* methods. In the case of *Neighborhood* methods, the user-item ratings stored in the system are used to predict ratings for other items directly. The prediction is done either by the *user-based* or the *item-based* way. In the first case, for a given user u and an item i, Heckmann et al. [7] use the stored ratings of i, given by users which are similar to u (called neighbors) to infer R_{ui} (the interest of the user u to the item i). In the *item-based* case, Khabbaz and Lakshmanan [13] use the ratings of u for items similar to i to predict R_{ui}. The methods used in *Neighborhood* CF can be grouped into two main approaches:

1. *Dimensionality reduction* such as Latent Semantic Indexation (LSI) [23] and Singular Value Decomposition (SVD) [14].
2. *Graph-based* such as Random Walk [4].

The major problem of *Neighborhood* methods is the limited space of predicted items. As the system recommends only items similar to those rated by the user or by his neighbors, it is hard to cover the huge number of items in the system, such as products in the *Amazon.com*[5] or films on *IMDB*[6].

The *Latent-Model-Based* methods use the stored ratings to learn a model. The main idea is to model the user-item interactions using factors representing latent characteristics of both users and items within the system. This model is learnt using available data and then utilized to predict user ratings on new items. In this context, many technics are used, such as Bayesian clustering [2], probabilistic Latent Semantic Analysis (pLSA) [8], Latent Dirichlet Allocation (LDA) [1], SVD [16], etc.

Thanks to its accuracy and scalability, the SVD technic is deeply used in CF [19, 22]. Most of the recent researches and recommendation tools (*LingPipe*[7], *pyrsvd*[8]) are based on the basic model or improved models of SVD like SVD++ [17] and timeSVD++ [16].

Although it is a very prevalent Matrix Factorization (MF) method, just a few works use Principal Component Analysis (PCA) in RS field. Goldberg et al. [6] propose *Eigentaste*, a CF algorithm to recommend jokes. The proposed algorithm starts by creating M, an $n \times m$ matrix, containing the ratings given by n users to m items. In the second step, the algorithm extracts the two eigenvectors having the highest eigenvalues of the correlation matrix of M. Then, it projects data in the eigenvectors plan. Finally, it uses *Recursive Rectangular Clustering* to create clusters of neighbors. Those clusters are used to predict users' preferences.

With the growth of social networks, a new kind of RS referred to as *Social Recommender Systems* (also known as *Community Recommender Systems*) gained popularity. The main idea is that users tend to trust more the recommendations from their friends than an anonymous user or seller [26]. FilmTrust [5] is the ideal example, it combines a movie rating and review system with a trust network (A social network expressing how much the members of the community trust each others). Yang et al. [26] present a deep review of collaborative filtering based social recommender systems. As the *Community RS* are based on link information only, they are limited to the user firends recommendations. Furthermore, given the big amount of items to recommend, and the fact that the *egocentric network*[9] is very small compared to the the whole social network, by looking only in the egocentric network, a considerable part of items are ignored. Also, the social network is not available in all RS.

[5] http://www.amazon.fr
[6] http://www.imdb.com
[7] http://alias-i.com/lingpipe/index.html
[8] https://code.google.com/p/pyrsvd/
[9] The network around a single node (ego)

3 The GLER Algorithm

In this section, we describe our algorithm GLER (**G**rouping **L**ike-minded users to **E**stimate **R**atings). GLER algorithm aims to create groups of Like-Minded people, in order to estimate future user-movie ratings, based on the stored ratings of that user and those having similar cinimatic tastes. Such a prediction leads to triple improvements:

1. Enhancing the training base of the predictor, as we exploit Like-Minded users' experiences.
2. Facilitating the deployment of the recommendation algorithm, by creating groups based on the interest centers and not on the information about social network links, which are not available in all RS.
3. Improving the accuracy of the predictor by dividing the training base into sub-bases containing homogeneous entries.

To create groups of Like-Minded users, we use PCA, which proved its effectiveness in this area [12, 10]. Once created, we use these groups to train the RS, and so, to build a model for each group. To estimate a new user-movie rating for an existing user, we use the model corresponding to the group that the user belongs to. For a new user, we calculate the distance between this user and the users of each group, and we use the model of the closest group. Figure 1 presents the proposed method. In the remainder of this section, we detail the steps of our algorithm.

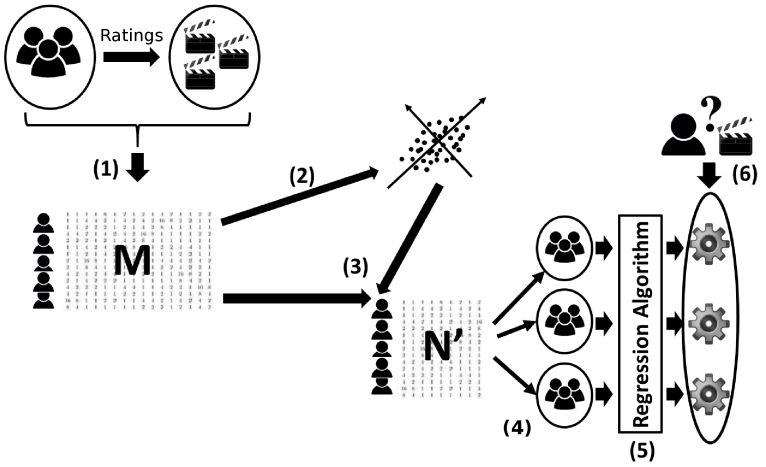

Fig. 1. The proposed method steps: (1) Use the stroed user-movie ratings to create the matrix M. (2) Retrieve the latent interest centers using PCA. (3) Project the matrix M into the eigenvectors space. (4) Group like-minded users according to their interest centers. (5) Use the regression algorithm to build a model for each group. (6) Predict user-movie rating using the model corresponding to his group.

3.1 Grouping Similar Users

Create the Matrix of Users' Interests Most of the RS algorithms use a $n \times m$ matrix of raw ratings from n users and m items. The matrix is partial (several rating are missed). Those algorithms aim to infer the missing ratings using a matrix factorization algorithm, such as SVD or LSI. In our work, we create a matrix representing users interests, and then use this matrix to retrieve the latent common interest centers. We start by creating a vector U for each user. The vector U contains the averages of user ratings per movie genre (Section 6). Next, we group users' vectors into a matrix M. Then, we normalize the matrix M to produce the matrix N. For the normalization, we use two methods. The first consists of normalizing all values in the matrix M as fllows:

$$n_{ij} = \frac{max_j - m_{ij}}{max_j - min_j} \tag{1}$$

Where max_j and min_j present respectively the maximum and minimum values in the column j of the matrix M. m_{ij} is the average of ratings by the user i of the movie genre j. In the second normalization method, we divide each value by the standard deviation of each column.

$$n_{ij} = \frac{m_{ij}}{\sigma_{m_{kj}}} \; ; \; with \; k = 1..n \tag{2}$$

With n is the number of users. In the remainder of this article, we use $Norm_1$ and $Norm_2$ to refer to the first and the second normalization methods respectively.

Retrieve Latent Factors In this step, we seek the latent factors to model users using PCA which is a popular matrix factorization method, generally used to reduce the data dimensions or to retrieve the latent centers where the data is concentrated [11]. In our work, we use PCA for both reasons:

1. Reduce the data dimensions, and so reduce the sparseness and data noise.
2. Retrieve the latent axes where the data is concentrated which represent the users' interest centers.

To find the latent interest centers and reduce the data dimensons using PCA, we start by calculating the covariance matrix C, given by:

$$C = \frac{1}{n-1} N^T N \tag{3}$$

With N is the normalized form of the Users' Interests matrix, created in the previous step. Then, we retrieve the matrices Λ and E by solving the equations (4) and (5):

$$C = E^T \Lambda E \tag{4}$$

$$ECE^T = \Lambda \tag{5}$$

The columns of the matrix E correspond to the eigenvectors of C. Λ is a diagonal matrix whose elements correspond to the eigenvalues of C.
As the principal components correspond to the axis around which the data is concentrated, we can say that those axis present the groups of users where the correlation is maximal. We use the retrieved eigenvalues to determine the number of eigenvectors to retain, and the number of user groups [10].
To retrieve the number of eigenvectors to retain and user groups from eigenvalues, we adopt the Scree Test Acceleration Factor proposed in [21]. Finally, we project our data in the space of the retained eigenvectors:

$$N' = NE_v^T \tag{6}$$

Where v is the number of eigenvectors to retain, E_v is the matrix formed by the retained eigenvectors, and N' is the reduced form of the matrix N.

Create Groups Many algorithms can be used to cluster users. In this work, we adopt the K-Means algorithm to group the similar users, for its simplicity and effectiveness. As inputs, K-Means takes the set of data N' and the number of groups to be identified k. In our work, we set $k = v$, as the number of eigenvectors to retain reflects the number of interest centers within the input data. At the end of this step, we obtain v clusters of users. Each cluster presents a group of users having similar proprieties and tastes.

3.2 Predict Ratings

Predicting a user rating could be treated as both a regression and a classification problem. If the ratings are in the form of classes or labels, the prediction is treated as a classification problem. In revange, the prediction is considered as a regression issue, if the ratings are discrete numerical values (such as the five star evaluation in the MovieLens RS). In this work, we adopt three regression algorithms using Weka[10]. The adopted regression algorithms are:

M5P M5P is a reconstruction of Quinlan's M5 algorithm [20] for inducing trees of regression models. It builds regression trees whose nodes are chosen over the attribute that maximizes the expected error reduction, and the node leaves are composed of multivariate linear models.

Multi-Layer Perceptron (MLP) An MLP can be viewed as a logistic regression classifier where the input is first transformed using a learnt non-linear transformation Φ. This transformation projects the input data into a space where it becomes linearly separable. Hornik et al. [9] prove that three-layer perceptron networks are theoretically universal approximators.

[10] http://www.cs.waikato.ac.nz/ml/weka/

Support Vector Regression (SVR) Support Vector Machines (SVM) can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The SVR uses the same principles as the SVM for classification, with the introduction of an alternative loss function. The loss function must be modified to include a distance measure. In Weka, there are two alternatives of SVR, namely: ε-SVR [25] and ν-SVR [24].

For each user group produced in the previous step, we use the stored users ratings to learn the regression algorithm. Once learnt, the system is able to infer user ratings in this group. For a new user - who is not included in the learn set - we assign him to the nearest group, based on the cosine distance between the new user vector and the average of uesrs vectors in each group.

4 Dataset and Baseline

4.1 Dataset

To evaluate the accuracy of the proposed algorithm, we use the *MovieLens-100K data sets*[11]. The data set was collected by the GroupLens Research Project, through the MovieLens web site[12] during the seven-month period from September 19th, 1997 through April 22nd, 1998. This data has been cleaned up - users who had less than 20 ratings or did not have complete demographic information were removed from this data set. This data set consists of 100,000 ratings [1-5] from 943 users on 1,682 movies. Demographic information about the users are: age, gender, occupation and zip code. The information about a movie in the dataset are: movie title, release date, video release date, *IMDB* URL and 19 binary values corresponding to the 19 movie genre. Those values indicate if the movie is or is not of the corresponding genre.
The data is distributed over 5 dataset. Each dataset contains a train and test data. The train and test sets contain respectively 80,000 and 20,000 entries. Each entry is in the form: *"user id"* | *"item id"* | *"rating"* | *"timestamp"*.

4.2 Baseline

We use SVD++ to evaluate and position our new algorithm. SVD++ is a popular Matrix Factorization RS. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. The latent space tries to explain the ratings by characterizing both products and users on factors automatically inferred from user feedback [16].
Accordingly, each item i is associated to a vector $q_i \in R^f$, and each user u is associated to a vector $p_u \in R^f$. For a given item i, the elements of q_i measure the extent to which the item possesses those factors. For a given user u, the elements of p_u measure the extent of interest the user has in items that are high

[11] http://grouplens.org/datasets/movielens/
[12] https://movielens.org

on the corresponding factors. In the movie RS context, the elements of q_i express the extent to which the movie i belongs to movie genres, and the elements of p_u measure the extent of interest the user u has in movies of corresponding genres. The resulting dot product, $q_i^T p_u$ captures the interaction between the user u and the movie i.
Let b_i and b_u denote the observed deviations of user u and item i, respectively, from the average μ (the overall average rating). For example, if the average of the observed ratings $\mu = 2.5$, and the average of the stored ratings on the item i is 3, and the average of the ratings of the user u is 2, then $b_i = 3 - \mu = 0.5$ and $b_u = 2 - \mu = -0.5$.
The prediction rule of the basic SVD recommender is given by adding the product $q_i^T p_u$ to the sum of the three parameters μ, b_i and b_u:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u \tag{7}$$

To improve the basic SVD recommender by adding implicit feedback, Koren [14] adds a second set of item factors, relating each item i to a factor vector $y_i \in R^f$. Those new item factors are used to characterize the users based on the set of items that they rated. The new rule to predict ratings is given by:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right) \tag{8}$$

Where $R(u)$ contains the items rated by the user u. Thus, the user u is represented by $p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j$, insteed of p_u. Thus, the user representation in the equation 7 is enhanced, by adding the perspective of implicit feedback.

5 Accuracy Measures

Typically, the rating dataset is split into a training set R_{train} and a test set R_{test}. R_{train} is used to generate models and tune the recommender system parameters. R_{test} is used to evaluate the recommender system. Let $\hat{R}_{u,i}$ the predicted ratings, and $R_{u,i}$ the recorded ratings. To assess the recommender system accuracy, we use two prevalent measures of accuracy, namely: *Root Mean Squared Error* and *Mean Absolute Error*.
Root Mean Squared Error (RMSE)
RMSE is one of the most popular metric used in evaluating the accuracy of predicted ratings. RMSE between known and predicted ratings is given by:

$$RMSE = \sqrt{\frac{\sum_{(u,i) \in R_{test}} (\hat{r}_{ui} - r_{ui})^2}{|R_{test}|}}$$

RMSE disproportionately penalizes large errors. For example, given a test set with four hidden items, RMSE would prefer a system that makes an error of 2 on three ratings and 0 on the fourth to one that makes an error of 3 on one

rating and 0 on all three others.

Mean Absolute Error (MAE)

MAE is a popular alternative of RMSE, given by:

$$MAE = \sqrt{\frac{\sum_{(u,i)\in R_{test}} (\hat{r}_{ui} - r_{ui})^2}{|R_{test}|}}$$

Unlike RMSE, MAE prefers systems that make the least number of errors. In the previous example, MAE would prefer the system that makes an error of 3 on one rating and 0 on all three others.

6 Results and Discussion

To evaluate our algorithm on the the MovieLens-100k dataset, we start by creating the input matrix, as mentionned in the section 3.1. Thus, we create a user vector $U_x = \{age, gend, occ, zip, year, rg_1, ..., rg_{19}\}$, where "$age$", "$gend$", "$occ$", and "$zip$" represent respectively the age, gender, occupation and zipcode of the user x. "$year$" is the average of rated movies release years. "rg_i" is the average of ratings that the user x assigns to movies belonging to the genre i. Thus, the vector U presents a user not only by his personal information (age, gender, etc.), but also by his interest to movies genre and release year. Then, we gather user vectors to create the matrix M. Thereafter, we normalize M to obtain the matrix N, and retrieve the eigenvalues and eigenvectors of the covariance matrix of N. The obtained eigenvectors correspond to the latent centers where the users interests are concentrated. Using those centers, we form groups of like-minded users. Finally, we divide the training base into several dataset according to the users' groups, and learn the regression algorithm to build a model for each group. Once learnt, we use these models to estimate ratings for users of the corresponding group.

As we use four different regression algorithms, we designate by $GLER_{M5P}$ (respectively $GLER_{MLP}$, $GLER_{\varepsilon-SVR}$ and $GLER_{\nu-SVR}$) the $GLER$ using M5P algorithm (respectively MLP, ε-SVR and ν-SVR). We applied the proposed method and the baseline algorithm (SVD++) on the MovieLens-100k dataset, and we calculate the accuracy using MAE and RMSE. Tables 1 and 2 illustrate the accuracy of the four alternatives of GLER on the MovieLens-100k five bases, using the $Norm_1$ and $Norm_2$ respectively.

We remark that the GLER algorithm provides MAE values lower than 0.31 and RMSE values lower than 0.44 for all the tests. Using the SVR regression algorithms ($GLER_{\nu-SVR}$ and $GLER_{\varepsilon-SVR}$), we obtain the best predictions, compared to $GLER_{M5P}$ and $GLER_{MLP}$. We can conclude that the predicted ratings using $GLER_{\varepsilon-SVR}$ are most of the time close from the right ones, and $GLER_{\nu-SVR}$ provides more exact ratings but the errors are larger than those in $\varepsilon - SVR$ case. This is can be explained by the fact that the $\varepsilon - SVR$ tends to control the amount of error in the model and go for the best performance without control of the support vectors in the resulting model, unlike the $\nu - SVR$ algorithm.

Table 1. Ratings prediction accuracy using $Norm_1$

	MAE	**RMSE**
$GLER_{M5P}$	0.3083	0.4312
$GLER_{MLP}$	0.2969	0.4198
$GLER_{\varepsilon-SVR}$	0.306	**0.3909**
$GLER_{\nu-SVR}$	**0.2869**	0.3972

Table 2. Ratings prediction accuracy using $Norm_2$

	MAE	**RMSE**
$GLER_{M5P}$	0.3057	0.4322
$GLER_{MLP}$	0.2902	0.4228
$GLER_{\varepsilon-SVR}$	0.301	**0.3864**
$GLER_{\nu-SVR}$	**0.2809**	0.3928

Table 3 presents the MAE and RMSE of GLER and the baseline algorithm on the five bases of the MovieLens-100k dataset. The results show that the performances of our algorithm GLER exceed those of the baseline, and we obtain an improvement in the MAE of about 0.42 and a drop from 0.9065 to 0.3864 in the RMSE values. Thanks to the PCA used in our algorithm, we overcome the sparsity problem which characterizes RS data, and reduce the data dimension, which is very important and helpful for building the prediction model. Creating Like-Minded user groups based on their tastes leads to a better performance. Indeed, the prediction will be based, not only on the user's rating history, but also on users having the same cinematographic tastes. Furthermore, following the rule of *"divide and conquer"*, creating similar-user groups facilitates the task of the regression algorithms. In fact, creating a model for a smaller number of homogeneous users is more easier and improves the prediction accuracy.

Table 3. Comparison with the baseline accuracy

		MAE	RMSE
Baseline		0.7135	0.9065
$GLER_{M5P}$	norm1	0.3083	0.4312
	norm2	0.3057	0.4322
$GLER_{MLP}$	norm1	0.2969	0.4198
	norm2	0.2902	0.4228
$GLER_{\varepsilon-SVR}$	norm1	0.306	0.3909
	norm2	0.30102	0.3864
$GLER_{\nu-SVR}$	norm1	0.2869	0.3972
	norm2	0.2809	0.3928

7 Conclusion

In this work we presented the GLER algorithm for recommending movies. Our algorithm creates Like-minded user groups based on the user-movie ratings. Then, it uses those groups to build patterns and predict user-movie ratings. Experimenting on the MovieLens-100k dataset, we showed that GLER achieves better results than the SVD++ algorithm. This improvement manifests in the decrease of MAE and RMSE values of about 0.42 and 0.5201 respectively.
In our algorithm we used four regression algorithms to build recommendation model and infer ratings. The experimentations show that the $GLER_{\nu-SVR}$ and $GLER_{\varepsilon-SVR}$ have the highest accuracy.
Our algorithm uses explicit feedback (user ratings) to build the recommendation model. Also, the training and test experimentation are done offline. In future research, we plan to enhance our algorithm by including users' implicit feedback, and handle the online recommendation.

References

1. D. M. Blei, Andrew Y. Ng, M. I. Jordan, and J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2003.
2. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithm for collaborative filtering. In *Proceedings of the 14 th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.
3. R. Burke. The adaptive web. chapter Hybrid Web Recommender Systems, pages 377–408. Springer-Verlag, Berlin, Heidelberg, 2007.
4. F. Fouss, A. Pirotte, J.M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowl. and Data Eng.*, 19(3):355–369, March 2007.
5. J. Golbeck and J. Hendler. Filmtrust: movie recommendations using trust in web-based social networks. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 1, pages 282–286, Jan 2006.
6. K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
7. D. Heckmann, T. Schwartz, B. Brandherm, M. Schmitz, and M. von Wilamowitz-Moellendorff. Gumo - the general user model ontology. In *User Modeling*, pages 428–432, 2005.
8. T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, pages 89–115, 2004.
9. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989.
10. S. Jaffali, H. Ameur, S. Jamoussi, and A. Ben Hamadou. Glio: A new method for grouping like-minded users. In Ngoc Thanh Nguyen, editor, *Transactions on Computational Collective Intelligence XVIII*, volume 9240 of *Lecture Notes in Computer Science*, pages 44–66. Springer Berlin Heidelberg, 2015.
11. S. Jaffali and S. Jamoussi. Principal component analysis neural network for textual document categorization and dimension reduction. In *Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on*, pages 835–839, March 2012.

12. S. Jaffali, S. Jamoussi, A. Ben Hamadou, and K. Smaili. Clustering and classification of like-minded people from their tweets. In *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*, pages 921–927, Dec 2014.
13. M. Khabbaz and L. V. S. Lakshmanan. Toprecs: Top-k algorithms for item-based collaborative filtering. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 213–224, New York, NY, USA, 2011. ACM.
14. Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.
15. Y. Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 2009.
16. Y. Koren and R. Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 77–118. Springer US, 2015.
17. R. Kumar, B. K. Verma, and S. S. Rastogi. Article: Social popularity based svd++ recommender system. *International Journal of Computer Applications*, 87(14):33–37, February 2014. Full text available.
18. Zhigang Lu and Hong Shen. A security-assured accuracy-maximised privacy preserving collaborative filtering recommendation algorithm. In *Proceedings of the 19th International Database Engineering & Applications Symposium, Yokohama, Japan, July 13-15, 2015*, pages 72–80, 2015.
19. A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 39–42, 2007.
20. J. R. Quinlan. Learning with continuous classes. In *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore, 1992. World Scientific.
21. G. Rache, T. A. Walls, D. Magis, M. Riopel, and J. Blais. Non-graphical solutions for cattells scree test. *Methodology: European Journal of Research Methods for the Behavioral and Social Sciences*, 9(1):23–29, 2013.
22. R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.
23. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.
24. B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Comput.*, 12(5):1207–1245, May 2000.
25. V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
26. X. Yang, Y. Liu, Y. Guo, and H. Steck. A Survey of Collaborative Filtering Based Social Recommender Systems. *Computer Communications*, July 2013.