Evolutionary Based Matrix Factorization Method For Collaborative Filtering Systems

Dariush Zandi Navgaran *, Parham Moradi **, and Fardin Akhlaghian ***

*Department of Electrical and Computer Engineering University of Kurdistan, dariush.zandi.n@gmail.com
**Department of Electrical and Computer Engineering University of Kurdistan, p.moradi@uok.ac.ir
***Department of Electrical and Computer Engineering University of Kurdistan, f.akhlaghian@uok.ac.ir

Abstract: *The Matrix-Factorization (MF) based models have become popular when building Collaborative Filtering (CF) recommender systems, due to the high accuracy and scalability. Most of nowadays matrix factorization models don't have acceptable execution time during to large datasets. In this article, we introduce a new collaborative filtering recommender system, based on matrix factorization by using genetic algorithm.*

Keywords: Recommender System, Collaborative filtering, Matrix factorization, Genetic algorithm.

1. Introduction

Recommender system methods are usually classified into three categories, based on how recommendations are made: content based, collaborative filtering and hybrid approaches [1]. In content based methods, the system learns to recommend items based upon a description of the item and a profile of the user's interests. The similarity of items is calculated based on the features associated with the compared items. For example, if a user has rated a comedy movie , then the system will recommend the other comedy movies to the user [1–5]. Unlike content based methods, collaborative filtering methods try to predict the rate of items for a particular user based on the rated items by the other users. For example, in a movie recommendation application, in order to recommend movies to the user, the system tries to find the similar users which have similar rates to the movies and then the movies which are most rated by the similar users will be recommended [1] [4] [6] [7]. The hybrid approach combines collaborative and content based methods, to avoid certain limitations of content-based and collaborative systems [1], [8].

Inside a collaborative filtering recommender system, the user preferences on involved items are quantized into a user-item rating matrix, in which high rates denotes strong preferences. So, the problem of predicting rates in collaborative filtering methods can be regarded as the missing value estimation of user-item rating matrix. Collaborative methods can be divided into two models: the Neighborhood Based Model (NBM) [7], [9–12] and the Latent Factor Model (LFM) [9], [13–15]. In NBM based methods, first of all the similarity of the active user by the other users will be calculated, then the predicted rate of a item for the active user will be the weighted average of all the ratings of the similar users for the item. So similarity computation between items or users is an important part of this approach. On the other hand, LFM based methods, transforms both items and users to the same latent factor space. The latent space can be used to explain characterizing both products and users in term of factors.

Some of the most successful realizations of latent factor models are based on matrix factorization (MF). In the basic form of matrix factorization methods, both items and users charactizes by factor vectors which have been inferred from the user-item rating matrix. These methods have become popular in recent years due to their accurate results. In addition, they offer much flexibility for modeling various real applications [6]. Recently, several MF based approaches have been proposed, including Maximum Margin Matrix Factorization [14], Expectation Maximization for Matrix Factorization [16], and Probabilistic Latent Semantic Analysis [17]. Regularized Matrix Factorization (RMF), which is based on Gorrell et al.'s works on Generalized Hebbian Learning[18], [19] published by Brandyn Webb [15]. RMF produced accurate results. Moreover it is highly scalable and easy to implement.

In this work a new evolutionary based matrix factorization method has been proposed. Firstly, we propose a fast genetic algorithm matrix factorization, and finally, we developed MF model based on genetic algorithm. The main contribution of this work is use genetic algorithm in MF model.

2. Matrix Factorization

Suppose we have a set of users, U, and a set of items, I. Let R of size $|U| \times |I|$ or $n \times m$, is the matrix that contains all the ratings that the users have assigned to the items. Also, we assume that we would like to discover K latent features. Our task, then, is to find two matrices P (a $|U| \times K$ matrix) and Q (a $|I| \times K$ matrix) such that their product approximates R (1):

$$R \approx P \times Q^T = \hat{R} \quad (1)$$

978-1-4673-5634-3/13/$31.00 ©2013 IEEE

$$R = \begin{bmatrix} r_{11} & \dots & r_{1m} \\ \dots & r_{ij} & \dots \\ r_{n1} & \dots & r_{nm} \end{bmatrix} \approx \begin{bmatrix} p_{11} & \dots & p_{1k} \\ \dots & \dots & \dots \\ p_{n1} & \dots & p_{nk} \end{bmatrix} \times \begin{bmatrix} q_{11} & \dots & q_{1m} \\ \dots & \dots & \dots \\ q_{k1} & \dots & q_{km} \end{bmatrix}$$

In this way, each row of P would represent the strength of the associations between a user and the features. Similarly, each column of Q^T would represent the strength of the associations between an item and the features. To get the prediction of a rating of an item $item_j$ by $user_i$, we can calculate the dot product of the two vectors corresponding to $user_i$ and $item_j$ (2):

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{K} p_{ik} q_{kj} \quad (2)$$

Now, we have to find a way to obtain P and Q. One way to approach this problem is the first initialize the two matrices with some values, calculate how `different' their product is to M, and then try to minimize this difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference.

The difference here, usually called the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair (3):

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 \quad (3)$$

Here we consider the squared error because the estimated rating can be either higher or lower than the real rating.

To minimize the error, we have to know in which direction we have to modify the values of p_{ik} and q_{kj}. In other words, we need to know the gradient at the current values, and therefore we differentiate the equation (2) with respect to these two variables separately (4):

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij} q_{kj} \quad (4)$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij} p_{ik}$$

Having obtained the gradient, we can now formulate the update rules for both p_{ik} and q_{kj} (5):

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj} \quad (5)$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}$$

Here, α is a constant whose value determines the rate of approaching the minimum. Usually we will choose a small value for α, say 0.0002. This is because if we make too make a step towards the minimum we may run into the risk of missing the minimum and end up oscillating around the minimum.

The above algorithm is a very basic algorithm for factorizing a matrix. There are a lot of methods to make things look more complicated. A common extension to this basic algorithm is to introduce regularization to avoid over fitting [20]. This is done by adding a parameter β and modifies the squared error as follows (6):

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^{K} (\|P\|^2 + \|Q\|^2) \quad (6)$$

In other words, the new parameter β is used to control the magnitudes of the user-feature and item-feature vectors such that P and Q would give a good approximation of R without having to contain large numbers. In practice, β is set to some values in the range of 0.02. The new update rules for this squared error can be obtained by a procedure similar to the one described above [21–23]. The new update rules are as follows (7).

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha(2e_{ij} q_{kj} - \beta p_{ik}) \quad (7)$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2e_{ij} p_{ik} - \beta q_{kj})$$

3. Proposed Method

In this section we will described our proposed evolutionary based matrix factorization method.

In this method, each chromosome will be represented as concatenation of two factorized P and Q matrices (figure 1).

$$chromosome = \begin{bmatrix} p_{11} & \dots & p_{1n} & q_{11} & \dots & q_{1m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ p_{k1} & \dots & p_{kn} & q_{k1} & \dots & q_{km} \end{bmatrix}$$

Figure 1. chromosome representation

In this representation, each chromosome represents as a matrix in which the number of its columns will be equal to sum of the number of users and items and the number of rows will be equal to the number of latent features (K).

Moreover, the fitness value of each chromosome will be computed of the factorized user-item rate matrix and our goal is to minimize this error (8).

$$fitness = \sum_{i=0}^{n} \sum_{j=0}^{m} \left(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj} \right)^2 \quad (8)$$

After generating the primary population, we choose two chromosomes using the roulette wheel, and apply crossover and mutation operations on them. Then we replace them by the two worst chromosomes in the population.

For the crossover operation, two random points in the chromosome are generated, so that one of them is at the beginning of the chromosome (in P) and the other is at the end of it (in Q).

For mutation, we select a random point, from the known rates in the matrix R. Assume the selected known rate is r_{ij}, we should update the ith column of the randomly selected chromosome and the $(j + n)$th column

of the chromosome alternatively for P_i and Q_j respectively based on the equation (7). So the proposed update procedure for the chromosome is as follows:

$$p'_{ik} = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik}) \quad (9)$$

$$q'_{kj} = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj})$$

$$\text{where; } e_{ij} = (r_{ij} - \sum_{k=1}^{K} p_{ik}q_{kj})$$

For example suppose the 6×5 multiplication table as follow:

$$optimal\ R = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \\ 4 & 8 & 12 & 16 & 20 \\ 5 & 10 & 15 & 20 & 25 \\ 6 & 12 & 18 & 24 & 30 \end{bmatrix}$$

In this table the 10 cell values randomly are missed, and we try to predict the values for the unknown cells:

$$R = \begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & 5 \\ 2 & \mathbf{0} & 6 & 8 & \mathbf{0} \\ 3 & 6 & \mathbf{0} & 12 & 15 \\ \mathbf{0} & \mathbf{0} & 12 & \mathbf{0} & 20 \\ 5 & 10 & \mathbf{0} & 20 & 25 \\ 6 & 12 & 18 & 24 & 30 \end{bmatrix}$$

After the procedure of genetic algorithm based matrix completion, the matrices P and Q are obtained as follows for $K = 4$:

$$P = \begin{bmatrix} 0.550 & -0.084 & 0.505 & 0.785 \\ 1.242 & 0.726 & 0.475 & 1.043 \\ 1.150 & 1.073 & 1.423 & 1.552 \\ 1.649 & 1.790 & 1.466 & 2.019 \\ 2.192 & 2.480 & 2.101 & 1.910 \\ 2.515 & 2.535 & 2.243 & 3.069 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.041 & 1.399 & 1.489 & 2.035 & 2.985 \\ 0.846 & 1.164 & 1.681 & 2.321 & 2.931 \\ 0.756 & 0.639 & 1.416 & 2.157 & 2.826 \\ 0.673 & 1.371 & 2.208 & 2.676 & 2.812 \end{bmatrix}$$

More over the predicted values for the multiplication table will be obtained as follows:

$$\hat{R} = P \times Q$$

$$= \begin{bmatrix} 0.861 & \mathbf{2.070} & \mathbf{3.125} & \mathbf{4.113} & 5.0281 \\ 1.728 & \mathbf{4.320} & 6.049 & 8.035 & \mathbf{10.116} \\ 3.078 & 5.899 & \mathbf{8.963} & 12.060 & 14.969 \\ \mathbf{4.052} & \mathbf{8.101} & 12.004 & \mathbf{16.083} & 19.994 \\ 5.065 & 9.920 & \mathbf{14.630} & 19.866 & 25.125 \\ 6.014 & 12.117 & 17.966 & 24.063 & 29.913 \end{bmatrix}$$

4. Experiments

The experiments are conducted 6×5 multiplication table and Movilens data set. The movie lens dataset used in this article, has been collected by the University of Minnesota, from the MovieLens website, during a period of 7 months [25]. This dataset consists of 100,000 ratings (based on 5 scores), on 1682 movies, rated by 943 users. Thus, the rating matrix has 1586126 ratings, of which, only 100,000 ratings are known and the rest (more than 1,400,000) are unknown and must be predicted.

Based on this dataset, the length of the chromosome is 943 + 1682 = 2625 and its width is, equal to the number of latent features, which has a value of 10, in this case. The number of generations, which specifies the termination condition, is equal to 10,000 generations. The crossover rate is equal to 0.8. Next, we describe two important parameters which play a major role in this article, namely population size and mutation rate.

Figure 2 depicts the ratio between the fitness (or genetic algorithm error) and the number of the generations. As shown in the figure, this algorithm has been implemented on the MovieLens dataset, for different chromosome populations, from 50 to 125. For each population size, the algorithm has been iterated 10 times, and the mean has been calculated. Because of low diversity in the population sizes below 50, we see improvements only in the first steps of the iteration. The best result is for population sizes 100 and 125.

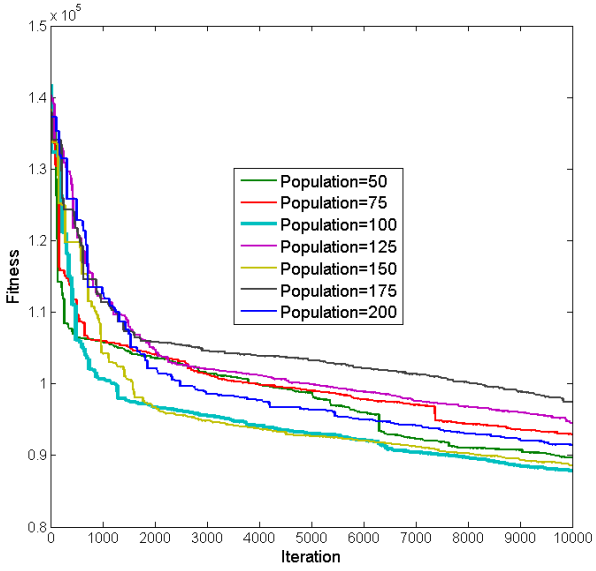

Figure 2. Compariosn between different population size on the MovieLens dataset, with crossover rate 0.8, and mutation rate 0.3.

As shown in the figure 3, mutation rate is estimated from small values to large ones. For each mutation rate, this test has been implemented 10 times on a population of size 100, and their mean has been calculated. According to the figure 3, the best mutation rate is 0.3. The reason for this high mutation rate is because of the type of the mutation.

The error in this method is almost the same as matrix factorization methods. What makes this method preferable is the speed of execution, which is about 10 times faster than other methods. This is because, in this method, instead of updating individual matrix elements, the whole matrix is updated.

For multiplication table we experiment another limitation which is significant in this article is the value of K, or the number of latent feature. The figure 4 shows that which by growth K, the error rate is reduced. By growing K, the algorithm's execution time is increased. This is because by grow the number of latent feature, the size of chromosome is enlarged, and the calculation time is increased.

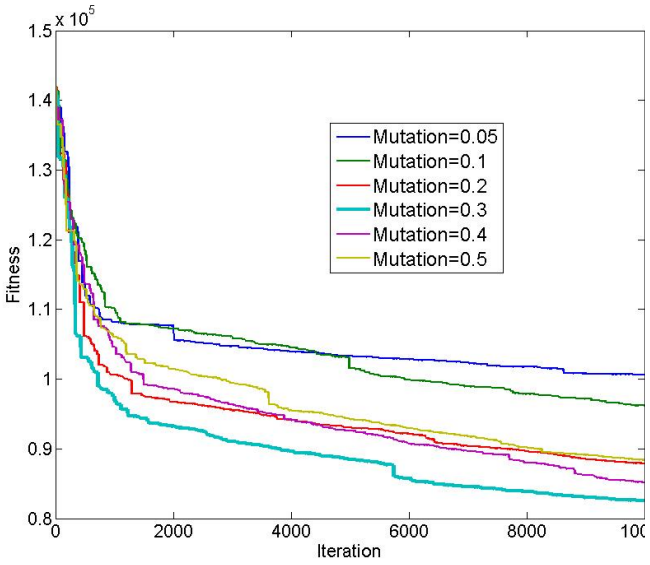

Figure 3. Compariosn between different mutation rate on the MovieLens dataset, with crossover rate 0.8, and population size 100.

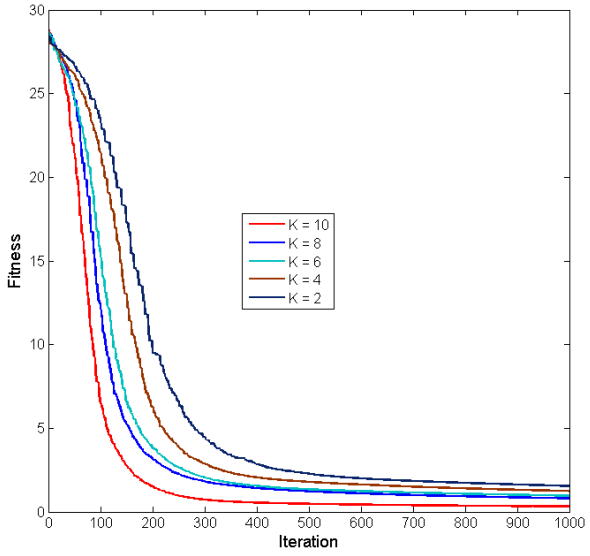

Figure 4. Comparison between different number of latent feature, with crossover rate 0.8, and mutation rate 0.1, and poprlation size 60 on 6×5 multiplication table.

5. Conclusion

In this paper, a new evolutionary based matrix factorization method has been proposed for recommender systems. In the proposed method, the factorize parts have been initialized randomly in a single chromosome to form genetic algorithm's population. The population is updated during each generation, in order to reduce the error. The simulation results on different dataset shows the effectiveness of the proposed algorithm for predicting unrated items for user-item rate matrix.

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, no. 6, pp. 734–749, Jun. 2005.

[2] Y. Peng, G. Kou, Y. Shi, and Z. Chen, "a Descriptive Framework for the Field of Data Mining and Knowledge Discovery," *International Journal of Information Technology Decision Making*, vol. 07, no. 04, p. 639, 2008.

[3] C. HAYES, "Context boosting collaborative recommendations*1," *Knowledge-Based Systems*, vol. 17, no. 2–4, pp. 131–138, May 2004.

[4] J. Yang and K. Li, "Recommendation based on rational inferences in collaborative filtering," *Knowledge-Based Systems*, vol. 22, no. 1, pp. 105–114, 2009.

[5] J. Bobadilla, F. Serradilla, and A. Hernando, "Collaborative filtering adapted to recommender systems of e-learning," *Knowledge-Based Systems*, vol. 22, no. 4, pp. 261–265, 2009.

[6] F. Ricci, L. Rokach, and B. Shapira, "Introduction to Recommender Systems Handbook," *Recommender Systems Handbook*, vol. 40, no. 21, pp. 1–35, 2011.

[7] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowledge-Based Systems*, vol. 23, no. 6, pp. 520–528, 2010.

[8] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artificial Intelligence Review*, vol. 13, no. 5, pp. 393–408, 1999.

[9] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," *Proceedings of the tenth international conference on World Wide Web WWW 01*, vol. 1, pp. 285–295, 2001.

[11] R. M. Bell, Y. Koren, P. Ave, and F. Park, "Improved Neighborhood-based Collaborative Filtering," *Methods*, vol. 5, no. 2, pp. 7–14, 2007.

[12] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[13] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender

systems: a case study," in *In ACM WebKDD Workshop*, 2000, vol. 67, no. 3B.

[14] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola, "Maximum-Margin Matrix Factorization," *Advances in Neural Information Processing Systems 17*, pp. 1329–1336, 2004.

[15] "netflix update: try this at home." [Online]. Available: http://sifter.org/~simon/journal/20061211.html. [Accessed: 13-Jan-2012].

[16] A. B. K. C. M. Kurucz, "Methods for large scale SVD with missing values," *Proceedings of KDD Cup and Workshop*, pp. 31–38, 2007.

[17] T. Hofmann, "Unsupervised Learning by Probabilistic Latent Semantic Analysis," *Machine Learning*, vol. 42, no. 1, pp. 177–196, 2001.

[18] G. Gorrell, "Generalized Hebbian Algorithm for Incremental Singular Value Decomposition in Natural Language Processing," *International Journal*, vol. 11, pp. 97–104, 2006.

[19] G. Gorrell and B. Webb, "Generalized hebbian algorithm for incremental latent semantic analysis," *Ninth European Conference on Speech Communication and Technology*, pp. 1325–1328, 2005.

[20] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk, "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *Journal of Machine Learning Research*, vol. 10, pp. 623–656, Jun. 2009.

[21] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Matrix factorization and neighbor based algorithms for the netflix prize problem," *Proceedings of the 2008 ACM conference on Recommender systems RecSys 08*, p. 267, 2008.

[22] P. Ott, "Incremental Matrix Factorization for Collaborative Filtering," *Applied Sciences*, 2008.

[23] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, no. 1, pp. 556–562, 2001.

[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989, p. 432.

[25] "MovieLens Data Sets | GroupLens Research." [Online]. Available: http://www.grouplens.org/node/73. [Accessed: 20-Jan-2012].