Latent Feature Based Recommender System for Learning Materials Using Genetic Algorithm

Mojtaba Salehi*
Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran
mojtaba.salehi@kntu.ac.ir

Received: 21/Nov/2012 Revised: 16/Jun/2014 Accepted: 10/Aug/2014

Abstract
With the explosion of learning materials available on personal learning environments (PLEs) in the recent years, it is difficult for learners to discover the most appropriate materials according to keyword searching method. Recommender systems (RSs) that are used to support activity of learners in PLE can deliver suitable material to learners. This technology suffers from the cold-start and sparsity problems. On the other hand, in most researches, less attention has been paid to latent features of products. For improving the quality of recommendations and alleviating sparsity problem, this research proposes a latent feature based recommendation approach. Since usually there isn't adequate information about the observed features of learner and material, latent features are introduced for addressing sparsity problem. First preference matrix (PM) is used to model the interests of learner based on latent features of learning materials in a multidimensional information model. Then, we use genetic algorithm (GA) as a supervised learning task whose fitness function is the mean absolute error (MAE) of the RS. GA optimizes latent features weight for each learner based on his/her historical rating. The method outperforms the previous algorithms on accuracy measures and can alleviate the sparsity problem. The main contributions are optimization of latent features weight using genetic algorithm and alleviating the sparsity problem to improve the quality of recommendation.

Keywords: Collaborative Filtering; Hybrid Recommendation; E-Learning; Learning Materials; Sparsity Problem.

1. Introduction

The information available on the internet is increasing exponentially and it is necessary to create technologies which can assist learners to discover the most valuable information to them from all available information. To help users deal with information overload and provide personalized recommendations, recommender systems have become an important research area since the first paper on collaborative filtering in the mid-1990s [1]. The task of delivering personalized e-learning material is often framed in terms of a recommendation task in which a system recommends items to an active learner [2]. Several educational recommender systems have been proposed in the literature which most of them focus on recommending suitable materials or learning activities [3].

In recent years, recommender system is being deployed in more and more e-commerce entities to best express and accommodate customer's interests. According to their strategies, recommender systems can be divided into three major categories: content-based, collaborative, and hybrid recommendation [1].

Majority of researches have used collaborative filtering as a recommendation strategy in recommender systems. The main idea of collaborative filtering is grouping like-minded users together. These systems which are also called clique-based systems, assume users who had similar choices before will make the same selection in the future. Initial hints of relating collaborative filtering to education have appeared in early relevant papers [4], but this strategy has been developed by other researchers [5-10]. Soonthornphisaj et al. [11] used collaborative filtering for material recommendation. First, the weights between the target learner and all the other learners are calculated by Pearson correlation. Then, n learners with the highest similarity to the active learner are selected as the neighborhoods. Finally, using the weight combination obtained from the neighborhood, the rating prediction is calculated. Bobadilla et al. [7] incorporated the learners score (obtained from a test) in the calculations by a new equation in collaborative filtering for item (material) prediction. Their experiment showed that the method obtained high item-prediction accuracy. Some of the (or some researchers) researchers used hybrid approaches for material recommendation. Liang et al. [12] implemented the combination of content-based filtering and collaborative filtering to make personalized recommendations for a courseware selection module. At first, a learner *u* enters some keywords on the portal of courseware management system. Then, the courseware recommendation module finds the *k* courseware with the same or similar keywords that others within the same learner interest group as learner *u*, choose. A relevance degree will be calculated for each *k* courseware by multiplying the degree of trust between learner *u* and other learners and evaluation of courseware by learner *u*. Finally the top five recommended courseware is outputted according to the recommendation degree. Their experiment revealed that the algorithm

* Corresponding Author

which they have used can reflect learners' preferences with high efficiency.

While the collaborative filtering algorithms try to address the information overload and personalization problem, with growing number of users and items tremendously, these algorithms will suffer from serious sparsity problems [13]. In addition, most traditional recommendation algorithms have been developed for e-commerce applications which cannot cover some necessary requirements of e-learning environments. One of these drawbacks is that most traditional algorithms only consider user's rating information and cannot take the contextual information of user and item such as their features into account. These algorithms assume that there are sufficient historical data for measuring similarity between items or users. However, most of the time this assumption is not held in e-learning environments, where we do not have adequate information about learners and also new e-learning materials are added to the system every day. However, by considering leaner's features and learning materials for the recommendation process, we can get better recommendations in learning environment. Therefore, it is necessary to consider features of materials and learners to improve the quality and accuracy of recommendations in learning environment. In our previous research, we introduced an adaptive hybrid recommender framework that considers features of materials and learners and also learners' dynamic interests in the unified model [14]. We used learner preference matrix that can model leaners' interest based on attributes of materials using historical rating of accessed materials by learners. In addition, a new adaptive strategy was used to model dynamic preference of learners. Unfortunately, in most cases we cannot find appropriate features and collect the corresponding data because some data are involved people's privacy and some features could not be described and coded formally. It leads towards low accuracy of prediction based only on the limited observed features [15]. Therefore, in this research, latent features that are extracted by factorization technique and are optimized by genetic algorithm using historical rating of users to alleviate sparsity problem, are introduced. Therefore, in the previous approach we had used observed features of learner and material but in this approach, we used latent feature of learner and material by matrix factorization technique.

An appropriate recommendation technique can be chosen according to pedagogical reasons. These pedagogical reasons are derived from specific demands of lifelong learning [16]. Therefore, some recommendation techniques are more suitable for specific demands of lifelong learning than others. One way to implement pedagogical decisions into a recommender system is to use a variety of recommendation techniques in a recommendation strategy. Therefore, this research combines results of feature-based techniques with traditional recommendation techniques. such new method employs the history rating data to get the optimized latent features weight for learners and materials using genetic algorithm and then uses these weight to generate recommendation.

The rest of the paper is organized as follows, at first the concept of latent features is described in section 2. In section 3, Methodology of research is described step by step. The empirical evaluations of the proposed approach are showed in section 4 and conclusion and review of the approach are presented in Section 5.

2. Latent Features

Collaborative filtering is the most successful technology for building recommender systems and is extensively used in many commercial recommender systems. this algorithms assumes that there are sufficient historical data for measuring similarity between items or users. However, this assumption is not held in various application domains such as e-learning environments where new learning materials are introduced every day and also adequate information about learners is not accessible. To address this drawback especially in e-learning environment, we introduce latent features of user and item that can be discovered by genetic algorithm and incorporated in the recommendation process to improve recommendation results and alleviate the sparsity problem.

Let $\boldsymbol{U}$ be the set of all users and let $\boldsymbol{I}$ be the set of all possible items. U_i denotes user i, and I_j denotes item j. Both user and item have their features, assuming the number of user's features is p and the number of features for items is q, features vectors define the user and item as: $U_i = (w_{i1}, w_{i2}, \dots, w_{ip})$, $I_j = (e_{j1}, e_{j2}, \dots, e_{jg})$. These features describe users and items in two high-dimensional spaces: Item space $\boldsymbol{I}$ and User space $\boldsymbol{U}$. Collaborative filtering uses the rating matrix which is the result of interaction between two high-dimensional spaces in order to find the relationships between them. According to collaborative filtering assumption, the user's historical ratings are the results of the interaction between features of the user and items, thus the hidden (unknown) ratings should also depend on the features of user and items too. Therefore, by discovering these features and also establishing the relationship between the ratings and these features, we can predict the rating of unrated items.

Interaction between users and items makes a rating matrix. Each cell can be illustrated as a triple set $H = \{(U_i, I_j, r_i^j)\}$ which represents user's historical preferences, where r_i^j is a user preference and it usually is represented by a rating, and can be obtained explicitly by the user or implicitly by some measures such as the frequency of user accesses to the corresponding item.

Since ratings depend on the characteristics of users and items, the rating prediction function could be denoted as $= (M, U_i, I_j)$, M is a prediction model which is learned from the historical rating data H; U_i and I_j are features of the user i and the item j, respectively. Because the selection of suitable features for user and item in a CF

problem is almost an impossible mission, which needs a lot of prior expert knowledge in the art fields rather than technology, unfortunately, in most cases we cannot apply the mentioned method directly. Even if the feature set is chosen, it is approximately impossible to collect the corresponding data because some data are privacy information of people or some features could not be coded formally. It leads towards low accuracy of prediction based only on the limited observed features [15].

However, we can use the historical rating data in a user-item matrix for discovering some valuable features of user and item that are called reflected characteristics of latent features of items and users. The predication models built based on the observed features plus latent features should relatively have higher prediction accuracy.

On the other hand, we can assume that users are grouped by their interests e.g., users in a group will prefer some kind of items and also item are grouped by the rating mode e.g., items in a group will have similar rating values to some kind of users. Then, to explain our approach we assume that the probabilities which belong to user groups could be seen as latent features for users, and the probabilities which belong to item groups could also be seen as a latent feature for items. Therefore, the main problem will be selecting suitable methods to find the probability which belongs to a user or an item in one of these groups. In this research, we proposed a genetic algorithm based method that can optimize this probability using the historical rating. in fact, each probability indicates value of one dimension in the latent features space of users or items [15].

Let U^L; I^L denote latent feature space for users and items, respectively; let the vectors $U_i^L = (w_{i1}^L, w_{i2}^L, \dots, w_{ik}^L)$ and $I_j^L = (e_{j1}^L, e_{j2}^L, \dots, e_{jk}^L)$ represent user and item latent attributes, respectively. The prediction function could be denoted as $\varphi' = (M, U_i, I_j, U_i^L, I_j^L)$ and the historical rating data could be converted to $H' = \{(U_i, I_j, U_i^L, I_j^L, r_i^j)\}$. This research uses Nearest Neighborhood as prediction model and also uses Genetic Algorithm to discover latent features.

Genetic algorithm is used because with tremendously growing number of learners and materials for learning environments, recommendation algorithms will suffer serious scalability problems. This happens because computational materials are going beyond practical or acceptable levels rapidly. In the features space, different people may place different emphases on the interrelated features. The goal of GA is to find the relationship between the overall rating and the underlying features ratings for each learner. More specifically, given the ratings data of a learner, GA computes his/her preference model in terms of feature weights.

3. Methodology

As mentioned before, information acquisition is a challenging problem in many real-world applications since collecting information about features of users and items is often very expensive or even not possible at all. Therefore to predict rating of users, we use latent features instead of observed features. Latent factor models which aim at mapping users and items to a common latent space by representing them as vectors in that space, were used to find latent features of users and items. dimensions of this space are called the factors. Here we should mention that we usually do not know the exact meaning of these factors and we are just interested in the correlation between the vectors in that space. Matrix factorization technique belongs to the family of latent factor models. We thoroughly describe this technique in the reminder of this section in order to describe the scientific base of our approach.

3.1 Matrix factorization technique

We presented a hybrid model in [17] that uses a preference matrix (PM) which can model the interests of a learner based on explicit attributes of learning materials in a multi-dimensional information model. In that research, fitness function used rating of users directly. to improve recommendation accuracy in this paper, we use matrix factorization to approximate a rating matrix $R \in S^{|U|\times|I|}$ which is the product of two smaller matrices U^L and I^L, i.e.

$$R \approx U^L.(I^L)^T \quad (1)$$

Where $U^L \in S^{|U|\times|K|}$ is a matrix where each row u is a vector containing K latent factors describing the user u, and $I^L \in S^{|I|\times|K|}$ is a matrix where each row i is a vector containing K latent factors describing the item i.

Let w_{iK}^L and e_{jK}^L be the elements of U_i^L and I_j^L in the vectors of U^L and I^L, respectively. The rating $r_x^{x'}$ given by a learner x to a material x' is predicted by Eq. (2):

$$p_x^{x'} = \sum_{k=1}^{K} w_{xk}^L.e_{x'k}^L = U_x^L.I_{x'}^L \quad (2)$$

The main issue of this technique is to find optimal values of the parameters U^L and I^L by a criterion such as Root Mean Squared Error (RMSE), which is determined by Eq. (3):

$$RMSE = \sqrt{\frac{\sum_{(x,x')\in D^{test}} (p_x^{x'} - r_x^{x'})^2}{|D^{test}|}} \quad (3)$$

Where D^{test} denotes the test data.

3.2 Optimization of latent features weight

In order to find optimal weight of the latent features of each user and item, $U_i^L = (w_{i1}^L, w_{i2}^L, \dots, w_{ik}^L)$, $I_j^L = (e_{j1}^L, e_{j2}^L, \dots, e_{jk}^L)$, we use genetic algorithms as a supervised learning task [18] whose fitness function is the Mean Absolute Error MAE of the RS. In this way (doing so), the population of our genetic algorithm becomes the set of different vectors of features weight. GA codes each possible latent features weight or solution candidate as a string, called chromosome. Each solution candidate is called individual, and the set of individual is called population. The population evolutes and a selection strategy are applied to choose better solution. While running our genetic algorithm, the successive population

generations tend to improve the MAE in the RS. Our genetic algorithm stops generating populations when MAE in the RS for a vector of features weight is lower than our threshold.

As usual, we use only a part of the whole rating data (training rating) in order to obtain optimal latent features weight. After obtaining the weights, we carry out our experiments to evaluate the feature weights using the test rating only.

3.2.1 Coding strategy

As usual, individuals of populations are represented in binary form as strings of 0s and 1s. Chromosome scheme which has been shown in the Fig. 1 represents the latent features weights for learners and materials where the N first rows indicate latent features weights for N learners or $U^L = (U_1^L, U_2^L, \dots, U_N^L)$. The remaining rows indicate latent features weights for M learning materials or $I^L = (I_1^L, I_2^L, \dots, I_M^L)$. Each component, w_{ik}^L or e_{jk}^L, in the latent features weight vectors, $U_i^L = (w_{i1}^L, w_{i2}^L, \dots, w_{ik}^L)$ or $I_j^L = (e_{j1}^L, e_{j2}^L, \dots, e_{jk}^L)$, will be represented by 10 bits.

The value of latent features weight is continuous and also between 0 to 1. in order to express these values with 1/1000th precision, because $512 = 2^9 < 1000 < 2^{10} = 1024$, 10 binary bits were used. Therefore, the number of columns in a chromosome will be $K \times 10$. by applying Eq. (4), These 10-bit binary numbers are transformed into decimal floating numbers, ranging from 0 to 1

$$x' = \frac{x}{2^{10}-1} = \frac{x}{1023} \tag{4}$$

Where x is the decimal number of each latent feature weight binary code. For example, the binary code for the weight of the *1*st feature of *user 1* in Fig. 1 is $(1110000100)_2$. The decimal value is $(900)_2$ and it is interpreted as $x' = \frac{900}{2^{10}-1} = 0.8797653 \approx 0.880$.

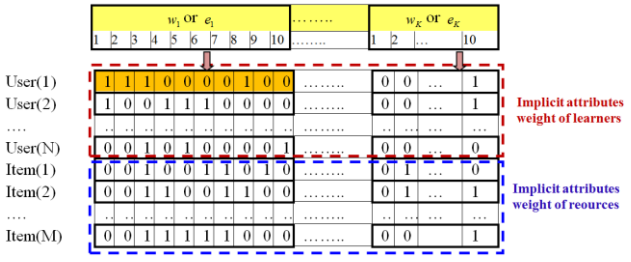

Fig. 1. Representation of latent features weight vectors in a chromosome

GA begins the process with a random population of chromosomes. Each chromosome is assigned alternately and tested by the fitness function. The fitness function measures the prediction accuracy of the rating using weights as defined in the current chromosome.

3.2.2 Fitness function

Two latent features weight matrixes, I^L and U^L which indicate latent features weight vectors for N learners and M materials respectively become the optimization targets. In our proposed genetic algorithm, the fitness function will be the MAE of the RS (indeed we only use the training rating since we are trying to find two optimal latent features weight matrixes) for particular matrixes, I^L and U^L.

The MAE is obtained by comparing the real ratings with the predicted ratings made according to the matrixes. In order to calculate the MAE of the RS for particular matrixes I^L and U^L, which have been generated in one of itterations of GA, we need to follow the next steps:

Obtaining the set of H neighbors of user x, H_x (the H most similar users to a given user x) and also the set of H neighbors of item x', $H_{x'}$ through latent features weight matrixes, U^L and I^L using Eq. (5) and Eq. (6):

$$sim(u_x, u_y) = \frac{\sum_{i=1}^{K} w_{xi}^L . w_{yi}^L}{\sqrt{\sum_{i=1}^{K} (w_{xi}^L)^2} . \sqrt{\sum_{i=1}^{K} (w_{yi}^L)^2}} \tag{5}$$

$$sim(i_{x'}, i_{y'}) = \frac{\sum_{i=1}^{K} e_{x'i}^L . e_{y'i}^L}{\sqrt{\sum_{i=1}^{K} (e_{x'i}^L)^2} . \sqrt{\sum_{i=1}^{K} (e_{y'i}^L)^2}} \tag{6}$$

Predicting the rate of item x' by user x, $P_x^{x'}$ is obtained through the Deviation From Mean (DFM) as an aggregation approach. First, the rate is predicted according to user-based similarity (Eq. (7)), $sim(u_x, u_y)$, and item-based similarity (Eq. (8)), $sim(i_x, i_y)$, separately and then results of both methods are unified by linear combination (Eq. (9)):

$$p_x^{x'}(u) = \bar{r}_x + \frac{\sum_{y \in H_x} sim(u_x, u_y) \times (r_y^{x'} - \bar{r}_y)}{\sum_{y \in H_x} sim(u_x, u_y)} \tag{7}$$

Where $P_x^{x'}(u)$ is the predicted rate of item x' by user x according to user-based similarity and $\bar{r}_x$ is the average of ratings made by the user x.

$$p_x^{x'}(i) = \overline{r^{x'}} + \frac{\sum_{y' \in H_{x'}} sim(i_{x'}, i_{y'}) \times (r_x^{y'} - \overline{r^{y'}})}{\sum_{y' \in H_{x'}} sim(i_{x'}, i_{y'})} \tag{8}$$

Where $P_x^{x'}(i)$ is the predicted rate of item x' by user x according to item-based similarity and $\overline{r^{x'}}$ is the rating average of item x' by the users.

$$p_x^{x'} = \lambda . p_x^{x'}(u) + (1-\lambda) . p_x^{x'}(i) \tag{9}$$

Where $P_x^{x'}$ is the predicted rate of item x' by user x and λ is the weight parameter of the unifying process.

Once every possible prediction is calculated by the unified method of item-based and user-based similarity, we obtain the MAE of the RS as in Eq. (10):

$$fitness(U^L, I^L) = MAE = \frac{1}{\#N} \sum_{x=1}^{N} \frac{\sum_{x'=1}^{M_i} \left| p_x^{x'} - r_x^{x'} \right|}{\#M_i} \tag{10}$$

Where $r_x^{x'}$ is the real rating of item x' by user x. When running the genetic algorithm, $\#N$ and $\#M_i$ represent the number of users and the number of training items rated by the user i respectively. It must be noted that since we only

use the training rating in the learning stage of model, some of rated items by each user are selected as the training set and the remaining items are considered as the test set. When $fitness(U^L, I^L)$ is lower, the rating prediction accuracy would be higher.

3.2.3 Genetic operators

common operators are used in our genetic algorithm: selection, crossover (recombination) and mutation. We have not used other possible operators like migration, regrouping or colonization-extinction because, we have obtained satisfactory results using these three classical operators. features of our operators are as follows.

Selection: fitness proportional selection is used for selection so that the selection probability of an individual depends on its fitness level. The selection probability of each string is calculated by Eq. (11):

$$p_c(U^L, I^L) = 1 - \frac{f_c(U^L, I^L)}{\sum_{C=1}^{PS} f_C(U^L, I^L)} \quad (11)$$

Where $f_c(U^L, I^L)$ denotes the value of fitness function for chromosome *c*, *PS* is the number of individuals in the population or population size and $P_c(U^L, I^L)$ denotes the selection probability of chromosome *c*. Since the sum of fitness in a population is constant, an individual with lower fitness (rating prediction accuracy) has larger probability to be chosen.

Crossover: One-point crossover is used to produce offspring. The crossover probability is set to 0.9. This operator is implemented for each row of chromosome as presented in Fig. 1 (each latent feature weight) separately.

Mutation: A single point mutation technique is used in order to introduce diversity. The mutation probability is set to 0.1.

3.3 Recommendation

After training the model, we use the optimized latent features weights to generate recommendations. The top-N materials with largest $P_x^{x'}$ are regarded as the top-N materials recommendations for learner *x*. Therefore, equation (9) gives us the recommendation score in the proposed latent-feature based method. Since similarity in this method is based on features of items and users, other than depending only on ratings of user, we hope this method alleviates sparsity problem.

4. Implication

Experiments are carried out for the evaluation of the proposed approach. In this section, we introduce the performance metrics and analyze the experiment results to evaluate the effectiveness of our proposed recommendation approach.

4.1 Experiment environment and data set

Learning records of a real-world dataset are applied in our experiments. The learning records dataset origins from the usage data of the course management system, Moodle[1]. Moodle is a free source e-learning software platform, also known as a Course Management System designed using pedagogical principles, to help educators by creating effective online learning communities. Moodle stores detailed records of students' activities and the educator can access summarized reports about these activities according to the categories specified by the Moodle system. The summery of dataset is presented in Table 1. The used dataset contains 40445 lending records from 1980 learners on 2931 books where each record contains timestamp and rating information (as the ratio of certain lending time segment to maximum lending time segment). In order to increase the number of records in the test set as much as possible and to eliminate the effect of accidental factors, the top 60% access records of each learner in the ordered dataset are used as the training set and the remnant 40% access records are used as the test set.

Table 1. The characteristics of the dataset used in the experiments

Number of users	Number of items	Number of transactions	Density
1980	2931	40445	0.698%

There are four approaches for evaluation of recommender systems including: a real environment, an evaluation environment, the logs of the system and a user simulator. By using an evaluation environment, we let a set of users to interact with the system over a period of time. In this approach, usually, the results are not reliable enough because the users often know the purpose of the evaluation. A few systems use simulated users to evaluate their performance. This approach can enable large-scale experiments to be implemented quickly repeatedly and perfectly controlled. However, the main drawback of this system is that it cannot simulate the real behavior of a user. Users are too complicated to predict and also their feelings, their emotions, and therefore, their actions change dynamically. Analysis of the log files of real users obtained in a real or evaluation environment is also a common technique for evaluation of recommender systems. In this method, the cross-validation technique is often used for evaluation of recommender system. Results obtained in a real environment with real users are the best way to evaluate a recommender system. But the main problem of the real and the evaluation environments is repetition of the experiments. Therefore, in this research we use log files of real environment that enables us to repeat the experiments and implement the cross-validation technique.

4.2 Evaluation metrics

In this paper, the evaluation metrics of recommendation algorithm are divided into two categories:

Classification Accuracy Metrics: these metrics assume the prediction process as a binary operation, either items are predicted (good) or not (bad). The precision and

[1] Modular Object-Oriented Dynamic Learning Environment

the recall are the most popular metrics in this category. They have been used by various researchers [19,20]. When referring to recommender systems, the recall and precision can be defined by Eq. (12) and Eq. (13):

$$\text{Recall} = \frac{tp}{tp + fp} \tag{12}$$

$$\text{Precision} = \frac{tp}{tp + fn} \tag{13}$$

Where *tp* stands for true positive, *fp* stands for false positive, and *fn* stands for false negative. The threshold for determining true positive is set to 3.5 meaning that if an item is rated 3.5 or higher, it is considered to be accepted by the user.

Since increasing the size of the recommendation set leads to an increase in the recall but at the same time a decrease in the precision, we can use F_1 measure [21,22] that is a well-known combination metric with the following Eq. (14):

$$F_1 = 2.\frac{\text{Precision.Recall}}{\text{Precision+Recall}} \tag{14}$$

Predictive Accuracy Metrics: these metrics measure how close predicted ratings of the recommender system are to the true user ratings. We use the MAE, a statistical accuracy metric [23] in this category which is computed with the following Eq. (15):

$$MAE = \frac{\sum_{x=1}^{x=N} \left| r_x^i - p_x^i \right|}{\left| N \right|} \tag{15}$$

Where P_x^i is the predicted rating for material *i* by learner *x*, r_x^i is real rating for material *i* by learner *x*, and *N* is the total number of learners.

4.3 Parameters setting

In the proposed feature-based recommendation method, we must run a set of experiments for adjusting parameters. The main parameters in GA comprise population size (PS), number of generations (NG), crossover probability (CP) and mutation probability (MP).

Since the results of genetic algorithms have stochastic nature, we have developed the experiments in a way that each set of parameters has been run for 50 times to reach a reliable conclusion. In this stage, we consider number of recommendation, RN=20, number of latent features, K=8, minimum number of rating required for test learners, M=100 and the number of learner, N=800.

Population size (PS): To be able to compare the effect of changing the initial size of the population on genetic algorithm efficiency and results, all of the parameters are fixed except the population. crossover probability would be set to (CP)=0.9 and mutation probability to (MP)=0.1. Generation numbers are chosen from 0 (initial generated data without running the algorithm) to 800 generations with the step size of 50 generations. The algorithm has been run 50 times for each population size and each generation value. Fig. 2 shows the results. This Figure compares only the average of best found solutions (in population).

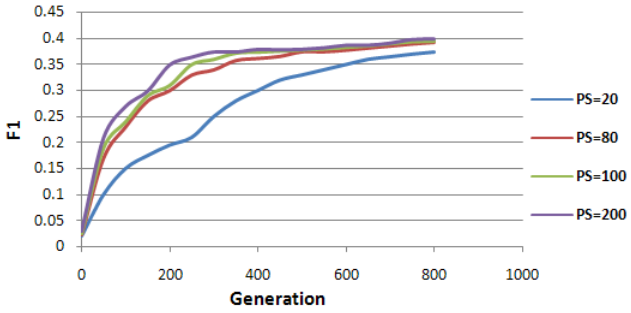

Fig. 2. Performance of the proposed method with respect of population Size

Results show that higher population size provides a high diversity and as a result, converging to better solutions happens sooner than smaller population sizes. On the other hand, higher population size needs more time for the algorithm to run. In this experiments, the population size of 20 lost diversity before reaching an acceptable solution. However, a population size of 200 does not provide much benefit over the population size of 100. Therefore, we will use a population size of 100 in our experiments.

Mutation probability (MP) and crossover probability of (CP): The optimal values of crossover and mutation probabilities are problem specifics that often are obtained by trial and error. Table 2 indicates the amount of F_1 for different value of MP and CP while NG=500, PS=100 and other parameters are similar to the previous experiment. According to experiment, CP=0.9 and MP=0.1 lead to good result for our problems.

Table 2. Performance of the proposed method with respect of MP and CP

MP	CP	F1
0	1	0.362
0.05	0.95	0.375
0.1	**.9**	**0.381**
0.15	.085	0.379
0.2	0.8	0.371
0.3	0.7	0.368
0.4	0.6	0.301

A Final parameter that must be adjusted is number of latent features. Fig. 3 shows the results obtained for the feature based approach with different number of latent features while NG=500, PS=100, CP=0.9 and MP=0.1. It can be seen that the performance improves steadily with increasing the number of features. To have high efficiency in the computation, we set K=8.

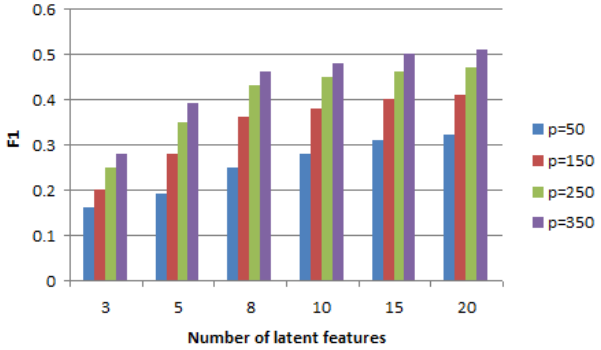

Fig. 3. Performance of the proposed method with respect of K

4.4 Comparative studies

4.4.1 Comparison of proposed method with Other algorithms

Table 3 presents a comparative study on recommendation quality between the proposed method and optimal state of five different algorithms in order to probabilistic recommendations: user-based CF using Pearson correlation with default voting (DV) [24], item-based CF using adjusted cosine similarity [25], two hybrid recommendation algorithms used by Pazanni [26] and Melville et al. [27], the personality diagnosis algorithm [28]. Comparisons were produced for N=800 learners with the average number of ratings equal to 100, M=80, CP=0.9, MP=0.1, NG=400 and PS=100.

As can be seen, linear combination of the proposed method (feature based method) with item based recommendation method generates better recommendations than other algorithms. Unlike individual feature based or item based method, the weighted method applies the results which are generated from two methods together and generates more qualified recommendation results.

Table 3. A comparison of prediction accuracy of various methods

System	Recall	Precision	F1
Linear combination of FBR with item based	**0.377**	**0.623**	**0.470**
FBR	0.373	0.583	0.447
User-based with DV	0.354	0.561	0.434
Item-based	0.321	0.53	0.400
Pazzani [24]	0.3622	0.601	0.452
Melville et al. [25]	0.373	0.619	0.465
Personality diagnosis	0.353	0.602	0.445

4.5 Performance evaluation for different sparsity levels

To illustrate that the proposed method can alleviate the sparsity problem, we increased the sparsity level of the training set by dropping some randomly selected entries. However, we kept the test set unchanged for each sparse training set. The performance of the proposed method algorithm was compared with other algorithms. Fig. 4 shows that the performance does not degrade rapidly in the case of proposed algorithm. It is because; features of an item can still be used for finding similar items. Furthermore, this algorithm enriches item and user profiles with combining latent features in recommendation process.

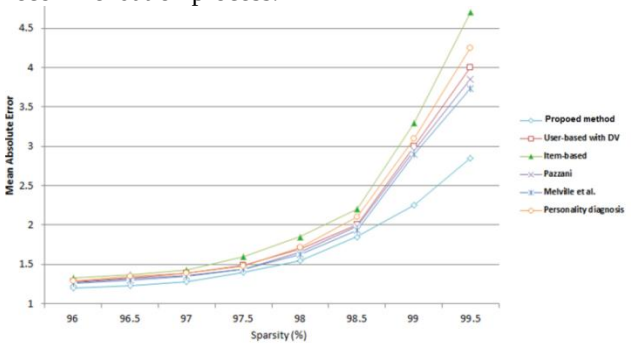

Fig. 4. Performance of algorithms under different sparsity levels

5. Conclusions

One of the most important applications of recommendation systems in the learning environment is personalization and recommendation of e-learning materials. However, since the repository of learning materials is very massive and these materials have several features, when applying the existing recommendation algorithms, there are some problems such as sparsity. To address sparsity problem and have better recommendations for learners, a hybrid recommender system is proposed to recommend learning items in users' learning processes. The proposed method discovers and optimizes latent features by GA and generates recommendation using collaborative filtering. The experiment results show that the proposed approach performs better than traditional approaches in the terms of accuracy measures measurements and also can alleviate sparsity problem. The main contribution of this paper is improving the quality of recommendations and addressing sparsity problem by considering latent features. For future researches, we can combine latent features with observed features to make a hybrid recommendation approach. In addition, we can make a comparison between meta-heuristic approaches for latent features optimization.

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, 2005.

[2] B. Mobasher, Data Mining for Personalization. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web: Methods and Strategies of Web Personalization, pp. 1–46, 2007.

[3] N. Manouselis, H. Drachsler, R. Vuorikari, H. Hummel, and R. Koper, Recommender systems in technology enhanced learning, Recommender Systems Handbook, pp. 387–415, 2011.

[4] L. Terveen, W. Hill, B. Amento, D. McDonald and J. Creter, "PHOAKS: a system for sharing recommendations" Communications of the ACM 40, pp. 59–62, 1997.

[5] T. Kerkiri, A. Manitsaris and A. Mavridou, "Reputation Metadata for Recommending Personalized learning

resources," Second International Workshop on Semantic Media Adaptation and Personalization, 2007, pp. 110–115.
[6] D. Lemire, H. Boley, S. McGrath and M. Ball, "Collaborative Filtering and Inference Rules for Context-Aware Learning Object Recommendation," International Journal of Interactive Technology and Smart Education, Vol. 2, No. 3, pp. 1–11, 2005
[7] J. Bobadilla, F. Serradilla, and A. Hernando, "Collaborative filtering adapted to recommender systems of e-learning." Artificial Intelligence (AI) in Blended Learning, Vol. 22, No 4, pp. 261–265, 2009.
[8] O.C. Santos, and J.G. Boticario, Recommendation strategies for promoting eLearning performance factors for all. In The 6th workshop on intelligent techniques for web personalization & recommender systems, 2008, pp. 89–98.
[9] N. Soonthornphisaj, E. Rojsattarat, and S. Yim-ngam, Smart E-learning using recommender system. Proceedings of the 2006 international conference on Intelligent computing, 2006, pp. 518–523.
[10] D. Aijuan, and W. Baoying, Domain-based recommendation and retrieval of relevant resources in e-learning. In IEEE international workshop on semantic computing and applications, 2008, pp. 103–108.
[11] N. Soonthornphisaj, E. Rojsattarat, and S. Yim-ngam, "Smart E-learning using recommender system", In Computational intelligence, Vol. 4114, pp. 518–523, 2006.
[12] G. Liang, K. Weining and L. Junzhou, "Courseware recommendation in e-learning system", In Advances in web based learning, pp. 10–24, 2006.
[13] X. Su and M. Khoshgoftaar, A Survey of Collaborative Filtering Techniques, Hindawi Publishing Corporation Advances in Artificial Intelligence, Volume 2009, Article No. 4, 19 pages, 2009.
[14] M. Salehi, I. Nakhai Kamalabadi M. B. Gaznavi Goushchi, A New Adaptive Hybrid Recommender Framework for Learning Material Recommendation, Vol. 5, No. 3, pp. 25-33, 2013.
[15] J. Zhong and X. Li, Unified collaborative filtering model based on combination of latent features, Expert Systems with Applications, Vol. 37, pp. 5666–5672, 2010.
[16] H. Drachsler H. G. K. Hummel, and R. Koper, "Personal recommender systems for learners in lifelong learning: requirements, techniques and model", International Journal of Learning Technology, Vol. 3, No. 4, pp. 404-423, 2008.
[17] M. Salehi, M. Pourzaferani, S. A. Razavi, Hybrid attribute-based recommender system for learning material using genetic algorithm and a multidimensional information model, Egyptian Informatics Journal, Vol.1, No. 14, pp. 67-78, 2013.
[18] D.E. Goldberg, Genetic algorithms in search, optomization, and machine learning. Reading, MA: addison-wesley, 1989.
[19] M. Pazzani, D. Billsus, Content-based recommendation systems. The adaptive web, pp. 325–341, 2007.
[20] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J. Riedl, "Evaluating Collaborative Filtering Recommender Systems", ACM Transactions on Information Systems 22(1) (2004) 5-53.
[21] Y.Y. Shih and D.R. Liu, "Product recommendation approaches: Collaborative filtering via customer lifetime value and customer demands", Expert Systems with Applications, Vol. 35, No. (1–2) pp. 350–360, 2008.
[22] Y.H. Cho, J.K. Kim, "Application of Web usage mining and product taxonomy to collaborative recommendations in e-commerce", Expert Systems with Applications, Vol. 26, pp. 233–246, 2004.
[23] P. Symeonidis, A. Nanopoulos, A.N. Papadopoulos, Y. Manolopoulos, Collaborative recommender systems: Combining effectiveness and efficiency. Expert Systems with Applications, 34(4) (2008) 2996–3013.
[24] J.S. Breese, D. Heckerman and C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, UAI'98 Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, pp. 43–52, 1998.
[25] B. Sarwar, G. Karypis, J. Konstan and J. Reidl, "Item based collaborative filtering recommendation algorithms", in Proceedings of the 10th international conference on World Wide Web., pp. 285–295, 2001.
[26] M.J. Pazzani, "A framework for collaborative, content-based and demographic filtering", Artificial Intelligence Review, Vol. 13, No. 5–6, pp. 393–408, 1999.
[27] P. Melville, R.J. Mooney and R. Nagarajan, "Content boosted collaborative filtering for improved recommendations". in Eighteenth National Conference on Artificial Intelligence, pp. 187–192, 2002.
[28] D. Pennock, E. Horvitz, S. Lawrence and C. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach", in Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, pp. 473–480, 2000.

Mojtaba Salehi received his B.Sc. degree from Shahid Bahonar University in 2004, M.Sc. degree from Tehran University in 2006 and Ph.D. degree from Tarbiat Modares University in 2013. During 2012, he was a researcher in Eindhoven University of technology in Netherlands, mathematics and computer science department. He was selected as the top student in Iran at 2012. He has about 20 journal papers. He is currently working as an Assistant Professor of K.N. Toosi University of Technology (KNUT). His research areas of interest include applied operation research, soft computing, data mining and recommender systems.