Effective Matrix Factorization for Online Rating Prediction

Bowen Zhou
Computer Science and Engineering
University of New South Wales
Kensington, NSW, Australia 2052
bowenzhou222@gmail.com

Raymond K. Wong
Computer Science and Engineering
University of New South Wales
Kensington, NSW, Australia 2052
wong@cse.unsw.edu.au

Abstract

Recommender systems have been widely utilized by online merchants and online advertisers to promote their products in order to improve profits. By evaluating customer interests based on their purchase history and relating it to commodities for sale these retailers could excavate out products which are most likely to be chosen by a specific customer. In this case, online ratings given by customers are of great interest as they could reflect different levels of customers' interest on different products. Collaborative Filtering (CF) approach is chosen by a large amount of web-based retailers for their recommender systems because CF operates on interactions between customers and products. In this paper, a major approach of CF, Matrix Factorization, is modified to give more accurate recommendations by predicting online ratings.

1. Introduction

Most online retailers provide the public, who have different interests, with a great variety of products and services. Therefore, in order to cater for customers interests by helping them in making choices from the large number of various items, recommender systems are increasingly used to shrink the range of products that might be attractive to a specific customer.

Based on accurate recommendations, sellers could gain their reputation among customers and increase the possibility that users make a purchase on recommended products. It has been shown by other researchers that movie recommender systems could be used to maximize revenues by applying effective algorithms [1].

Recommender systems usually take into consideration profiles of users and items respectively and give recommendations by applying algorithms on the information of attributes and/or users purchase history retrieved from these profiles and evaluating users po-

tential preferences of each item. There are two main approaches that could be used for the purpose of recommending products effectively. One is called Content-Based filtering, which creates a profile describing the types of items the user is interested in and a profile containing descriptions of items, and generates recommendations based on these two profiles [2].

Another approach called Collaborative Filtering generates recommendations based on users behaviour [3]. This approach focuses on a group of users or items that have similar behaviour or attributes and establishes relationships between each user (user-based filtering) or each item (item-based filtering) by analysing responds (e.g. movie ratings) each user gives to each item. In this case, responses from users to items are not only restricted to what users prefer, implicit feedbacks such as what users dislike are also of great significance [4].

One of CF approaches, Matrix Factorization (MF), has been popular in recent years. Most MF researches improve the accuracy by taking the data (e.g. movie ratings) as the point of penetration without considering the impact of item attributes [5], [6], [7]. Although these new MF approaches can provide more accurate ratings, an improvement from different perspective will be necessary when they reach a bottleneck.

Also, a qualified MF-based approach developed by Koren in [8] shows a problem of extreme long training time. His approach provides precise predictions for movie ratings while runs for 17 minutes for each iteration when there are 50 factors. In this case, if the practical recommender system updates the models frequently after receiving feedbacks from users and the number of active users is high, the server might suffer from the operation time.

Another trouble most CF approaches (not only MF) face is the cold-start problem. If a user has only few historic data available to the recommender system, it would be difficult to generate accurate recommendations or give accurate predictions to this user. A free software called MyMediaLite, developed by Zeno,

HICSS

Steffen, Christoph and Lars [9], gives predictions to new users by simply returning the global average of all the ratings. More precisely, authors of [10] provides a way to reduce the prediction error a little for those who have rated for few items.

To address these issues, this paper firstly takes attributes into consideration to enhance the accuracy of the original MF approach. Then the theory of k-nearest-neighbour approach will be implemented along with the improved MF approach to give more accurate predictions. Also, an effective method is used deal with cold-start problem and the effect is compared with [10].

In summary, the contributions of this paper can be summarized as follows:

- Item attributes are taken into consideration to enhance the accuracy of the MF approach;
- Ratings of k nearest neighbours of predicted items are used to modify the predictions for more accurate recommendation;
- An improved MF approach mines the most significant data to deal with cold-start problem.

The rest of this paper is organized as follows. Section 2 summarizes related work in Content-Based recommendations, Collaborative Filtering and Matrix Factorization (MF) based approaches. Details of MF are provided in Section 3. After that, we present our MF based approach in Section 4. Section 5 describes the experiments that show the effectiveness of our approach and Section 6 discusses the result. Finally, Section 7 concludes the paper.

2. Related work

Many recommendation methods have been studied due to the great demand in the current age of big data. Recommendation approaches are split into two types: Content-Based (CB) and Collaborative Filtering (CF).

2.1. Content-Based Recommendation

CB consists of three steps: Information Retrieval, Profile Learning, and Recommendation Generation.

2.1.1. Information Retrieval: In real applications the attributes of items can be distinguished as structured or unstructured information, as stated in [2]. Structured attributes often have explicit meaning and the values are within a certain range (e.g. age, gender), while unstructured attributes are often implicit and there are no restrictive range for their values (e.g. purchase history). Different from structured attributes which can be directly used in recommendation systems, unstructured information needs to be extracted from a fragment of contents in order to be quantized. An example of Vector

Space Model (VSM) is given in [2], which is the most commonly used approach to retrieve information from a section of text.

2.1.2. Profile Learning: This step is to build a model for a user according to his preference in the past. Based on this model, recommendation systems can infer whether this user will like/dislike a new item. This step can be basically considered as a supervised classification learning step. A typical profile learning approach in CB is the k-nearest-neighbour approach (KNN) which evaluates most similar items according to their attributes. Naive Bayes (NB) is also commonly implemented for profile learning, particularly for document classification. By calculating posterior probabilities, NB will assign an item to one of the two classes: user likes or user dislikes.

2.1.3. Recommendation Generation: This step simply returns the top-N items which attracts user the most according to results in profile learning.

2.2. Collaborative Filtering

There are two main types of CF: memory-based CF (user-based and item-based) and model-based CF [11]. Examples of memory-based CF approaches are given in [12], [13]. Memory-based CF finds nearest neighbours of a user by searching through the entire user database and generates a list of recommended items. Ka, Anto, Volker, Xiaowei and Hans-Peter developed their probabilistic memory-based CF approach and they stated that the approach needs to scan the whole database to construct the profile space of users [14]. In order to deal with the computation complexity and memory bottleneck problem, their *shift of workload* provides an efficient way, in which construction of the profile space can be considered as a backstage operation and will be conducted only when there is available computing power. Also, in order to deal with the long computation time, an item-based CF, which can obviously shorten the operation time without negatively affecting accuracy of prediction, is provided in [15].

Model-based CF uses user database to build a model to store ratings, which could alleviate scalability and sparsity issues [10]. Although model-based CF can provide predictions of similar quality to memory-based CF, it often takes long time to build and train the offline model, and it is difficult to update models. There are some commonly implemented model-based CF such as clustering algorithm, singular value decomposition (SVD), and Bayesian network. Clustering algorithm treats recommendation as a classification problem, which means that it iteratively computes the center of clusters and classify which cluster a user

should be assigned into based on the distance learning approach, and then give recommendation according to components in nearest clusters [10], [16]. SVD is closely related to the Matrix Factorization method that will be discussed later in this paper. It decomposes the original rating matrix R by $A = U \times S \times V^T$, where U and V are the left and right singular vectors and S is the diagonal matrix, and then chose the first k singular values to reduce the dimension of the diagonal matrix S. The reconstruction of $R_k = U_k \times S_k \times V_k^T$ gives the *rank-k* matrix which is the closest approximation of the original rating matrix [17]. Also, SVD is commonly used together with principal component analysis (PCA) to reduce the dimension of the original database [10], [18]. Bayesian network model, which is also called probabilistic directed acyclic graphical model, operates with conditional probability. In the network, probability of a variable influencing others is evaluated and then recommendation will be given based on dependencies between pairs of variables [11], [19], [20].

2.3. Matrix Factorization

There are some other researchers who improved the MF approach for more accurate predictions. Yifan, Yehuda and Chris introduced a model with probability and confidence to obtain more accurate values of user factors and item factors [5]. Hanhuai and Arindam used a Gaussian prior with an arbitrary mean over **u** and **v**, and combined the correlated topic model and latent Dirichlet allocation work with their matrix model [6]. Vikas, Serhat, Jianying and Aleksandra introduced a new binary matrix, in which 0 means 'not purchased' and 1 means 'purchased', into the factor-learning process in non-negative Matrix Factorization (NMF) approach [7]. Compared to the above algorithms, the proposed new MF approach in this paper concentrates on transforming the original rating matrix and modifying already-predicted ratings using item attributes. It might be possible to combine the proposed MF with previous approaches to give more accurate predictions since it achieves improvement from different perspective.

The k-nearest-neighbour (KNN) approach (different from KNN in Content-Based approaches) which evaluates neighbours based on ratings given by users, has been studied to be applied in recommender systems [21], [22]. Compared to the classic KNN technique, Matrix Factorization (MF) method proved to be more accurate when it is used to deal with datasets such as the one provided by the online DVD rental company Netflix, as stated by the winner of Netflix prize [23], so it has become a dominant approach of Collaborative Filtering recommenders and is decided to lead to the solutions to MovieLens dataset in this paper.

3. Matrix factorization with stochastic gradient descent

MF is basically based on matrix R containing ratings given by a user u_i to an item v_j. The total number of users and items is M and N respectively.

3.1. Matrix factorization

$$
R = \begin{pmatrix}
r_{1,1} & \cdots & r_{1,j} & \cdots & r_{1,n} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
r_{i,1} & \cdots & r_{i,j} & \cdots & r_{i,n} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
r_{i,1} & \cdots & r_{i,j} & \cdots & r_{i,n}
\end{pmatrix}
$$

$$
R = \begin{pmatrix}
r_{1,1} & \cdots & ? & \cdots & ? \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
? & \cdots & r_{i,j} & \cdots & ? \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
? & \cdots & r_{i,j} & \cdots & r_{i,n}
\end{pmatrix}
$$

As shown by the first $M \times N$ matrix, each entity, denoted by $r_{i,j}$, in the matrix refers to a rating user u_i gives to the item v_j, and the second matrix demonstrates that in most cases a majority of ratings are not available and need to be predicted. Therefore, the aim of MF algorithm is to predict all of the unknown ratings. Let U denote a $M \times K$ matrix and V denote a $N \times K$ matrix respectively, where K is the number of factors which need to be iteratively learned by the MF algorithm to generate a matrix such that $R \approx \hat{R}$, meaning that each entity in $\hat{R}$ represents a prediction of a rating. In this case, a specific rating $r_{i,j}$ can be evaluated by calculating the dot product of relevant rows in U and V respectively. Let $\mathbf{u_i}$ denote the *i-th* row of U and $\mathbf{v_i}$ denote the *i-th* row of V, and the predicted rating $r_{i,j}$ can be simply obtained via Equation (1). $\hat{u_{i,k}}$ and $\hat{v_{j,k}}$ in the equation are the approximation of the factors of users and items, which are not given, so they need to be learned by recommender systems to implement this formula.

$$
\hat{r_{i,j}} = \sum_{k=1}^{K} u_{i,k} \times v_{j,k} = \sum_{k=1}^{K} \hat{u_{i,k}} \times \hat{v_{j,k}} = \mathbf{u_i} \cdot \mathbf{v_j^T} \quad (1)
$$

3.2. Stochastic Gradient Descent

A problem of Equation (1) is that factors of users and items are undiscovered so that they need to be learned

by minimizing prediction errors obtained through the training set $\mathbf{T}$ which contains given ratings (2).

$$err_{i,j} = \hat{r_{i,j}} - r_{i,j} \qquad \forall r_{i,j} \in \mathbf{T} \qquad (2)$$

First of all, normal distribution is used to initialize factors in U and V. It is too restrictive to require only local minimization for some problems including [24], so stochastic gradient descent (SGD) algorithm, which iteratively modifies parameters to converge to a stationary point after some number of iterations [25], is chosen to be implemented iteratively through the training dataset to find the best-fit factors. Throughout SGD, each user factor will be updated as shown in (3), where η is the learning rate/step size (usually set to less than 0.2) and E is the error denoted by squared error $\frac{1}{2}(\hat{r_{i,j}} - r_{i,j})^2$ so that $u_{i,k}$ will be increased by the gradient of E with respect to $u_{i,k}$ itself. Equation (4) shows the update for item factor $v_{j,k}$.

$$u_{i,k} = u_{i,k} + \eta \frac{\partial E}{\partial u_{i,k}} \qquad (3)$$

$$v_{j,k} = v_{j,k} + \eta \frac{\partial E}{\partial v_{j,k}} \qquad (4)$$

By further calculation, the partial differential components in (3) and (4) can be simplified into (5) and (6), where n is the number of factors

$$\frac{\partial E}{\partial u_{i,k}} = \frac{\partial \frac{1}{2}(\hat{r_{i,j}} - r_{i,j})^2}{\partial u_{i,k}} = \frac{(\hat{r_{i,j}} - r_{i,j}) \cdot \partial \hat{r_{i,j}}}{\partial u_{i,k}}$$
$$= (\hat{r_{i,j}} - r_{i,j}) \frac{\partial (u_{i,1}v_{j,1} + \cdots + u_{i,n}v_{j,n})}{\partial u_{i,k}} \quad (5)$$
$$= (\hat{r_{i,j}} - r_{i,j}) \cdot v_{j,k}$$

$$\frac{\partial E}{\partial v_{j,k}} = (\hat{r_{i,j}} - r_{i,j}) \cdot u_{i,k} \qquad (6)$$

Below is the pseudo code for SGD algorithm used for MF. All the ratings in the training set R are sorted in random order, and within each iteration the randomly ordered rating will be used in sequence to implement (5) and (6). After iterating for $numIter$ times, only factors related to given ratings will be updated to give the minimum error. For example, if $r_{5,7}$ is not given in R, $u_{5,1\cdots numFactors}$ and $v_{7,1\cdots numFactors}$ will remain the same value as generated by normal distribution at the start of SGD procedure.

Input: training set R, number of factors numFactors, number of iteration numIter
Output: U and V with learned entities (factors)
For $n = 1 : numIter$
  For each rating $r_{i,j}$ in R, in random order
    For each factor $u_{i,k}, v_{j,k} \in [1, numFactors]$
    $u_{i,k} = u_{i,k} + \eta \frac{\partial E}{u_{i,k}}$, $v_{j,k} = v_{j,k} + \eta \frac{\partial E}{v_{j,k}}$

3.2.1. Overfitting: Since in the learning procedure the update of each user factor is only affected by item factor, and the item factor is changed only according to user factor, there exists a possibility that the learned factors are overfitting the training dataset, as illustrated in Figure 1. So a regularization process is introduced, which substitutes the user factors into the update of user factors, and the action for item factors is the same, as shown in euqation (7). λ is the regularization coefficient and there is no need to find a precise value of it because it is only used to prevent $u_{i,k}$ from "over oscillating".
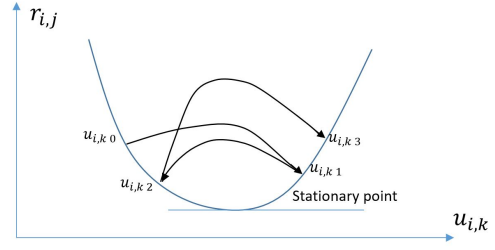

Figure 1: $u_{i,k}$ goes away from the stationary point from step 2 to step 3

$$u_{i,k} = u_{i,k} + \eta (\frac{\partial E}{\partial u_{i,k}} - \lambda \cdot u_{i,k}) \qquad (7)$$

4. Proposed Approaches

In this section, the proposed approaches will be discussed in detail.

4.1. MFA: Inserting movie attributes into dataset

Suppose that each movie has h manually defined attributes, so these attributes could be utilized to revise item factors. Firstly h virtual users will be created, and for each of virtual users, a spurious rating of each movie, the value of which would whether be min and max, will be assigned according to whether this specific movie has this specific attribute or not, where min and max are the minimum and maximum rating values defined in the dataset. For example, if the $10th$ movie has the $5th$ attribute, max will be assigned to the virtual rating given by the $(numUsers + 5)th$ to the $10th$ movie, and if the $11th$ movie does not have the $5th$ attribute, min will be assigned to the virtual rating given by the $(numUsers + 5)th$ to the $10th$ movie, where $numUsers$ is the total number of users existing in the dataset. In this case, when SGD operates on the newly inserted virtual ratings, item factors of corresponding movies in V will be modified and the predicted rating $\hat{r_{i,j}}$ will then be affected. The theory behind is that if two movies A and B have similar

attributes (e.g. both of them are romantic movies), during SGD item factors of A and B will similarly be increased or decreased, and this leads to the fact that item factors would be altered not only according to interactions between users and movies, but also in accordance with relationship between movies. In this stage, the proposed approach is named MFA.

4.2. MFA_KNN: Combine MF with K nearest neighbour

The k-nearest-neighbour method concentrates on the relationship between pairs of items, and then give recommendation based on the similarity between items. For example, if a user gives full mark to movie A, and movie B has a high similarity with A, the recommender system will generate a result that this user will enjoy B. Also, similar users might give close ratings to a specific movie. Different from the similarity estimation described in Section 2.2, the relationship between items can be evaluated by distance learning, as shown in Equation (8). $Dis_p(\mathbf{x}, \mathbf{y})$ is called the *MinKowski distance*, and $\mathbf{x}$ and $\mathbf{y}$ are the attribute vectors of item x and item y respectively. The most commonly used "distance" are *Manhattan distance* and *Euclidean distance*, which set p to 1 and 2 respectively.

$$Dis_p(\mathbf{x}, \mathbf{y}) = (\sum_{j=1}^{d} |x_j - y_j|^p)^{\frac{1}{p}} = ||\mathbf{x} - \mathbf{y}||_p \quad (8)$$

If two movies m_1 and m_2 both have h attributes, then m_1 and m_2 can be respectively denoted by $\mathbf{m_1}$ and $\mathbf{m_2}$, each of which is a vector with h elements. Applying the above theory will give Equation (9).

$$Dis_p(\mathbf{m_1}, \mathbf{m_2}) = (\sum_{j=1}^{h} |m_{1,j} - m_{2,j}|^p)^{\frac{1}{p}} = ||\mathbf{m_1} - \mathbf{m_2}||_p$$
$$(9)$$

Theoretically, if some of the nearest neighbours of an item v to be predicted for a user have already been rated by him/her, the accuracy of prediction of v can be improved by the difference between its neighbours' ratings and the average rating. This will be proved in the experiment. Also, it needs to be noticed that combining KNN with MF will not influence item factors so that it can be combined with other CF approaches.

4.3. Bias in prediction

Equation (1) simply takes the summation of product of each user factor and each item factor. It focuses on interactions between users and items but ignores the effects caused by users themselves. For example,

some users with rigorous critical thinking prefer to give lower ratings to each movie than the public, while some lenient users always give inspiring ratings (e.g. over 4 out of 5) to all of the movies. A bias is introduced to classify users of different tolerance.

$$\hat{r_{i,j}} = \bar{r_i} + \sum_{k=1}^{K} \hat{u_{i,k}} \times \hat{v_{j,k}} = \bar{r_i} + \mathbf{u_i} \cdot \mathbf{v_j^T} \quad (10)$$

$$\hat{r_{i,j}} = \bar{r} + \sum_{k=1}^{K} \hat{u_{i,k}} \times \hat{v_{j,k}} = \bar{r} + \mathbf{u_i} \cdot \mathbf{v_j^T} \quad (11)$$

Equation (10) introduces a bias $\bar{r_i}$ into the original evaluation of predicted ratings in Equation (1). In this case, predictions of users who tend to give low/high ratings will be lowered/increased by the factor $\bar{r_i}$, which is the average rating of user i, regardless of user factors and item factors obtained through SGD. Also, the increment, $\frac{\partial E}{\partial u_{i,k}}$, will not change due to the property of partial derivatives. Similarly, Equation (11) uses the global mean $\bar{r}$ as the bias, which can possibly avoid the problem caused by lack of ratings of some particular users who have rated only digital number of movies. The accuracy of these two biases needs to be tested.

4.4. MFAR: Setting threshold for cold-start problem

As most other Collaborative Filtering approaches [27], cold-start is a common problem for MF. It means that if a user has rated only few movies, recommender systems might not give predictions accurately due to lack of information of this user. To deal with this problem, a threshold th will be used to filter out most misleading ratings, meaning that ratings whose prediction error exceeds the threshold will be considered to negatively affect the accuracy of prediction. If a rating given by a cold-start user to movie is filtered out, it means that this rating is considered to mislead the user factors of this cold-start user, and if a rating of a movie which is not given by a cold-start user is filtered out, this rating is considered to mislead the item factors of this movie. More specifically, during SGD, after some number of iterations, if the absolute error of the prediction of an instance (ratings or attributes) in training dataset is still greater than th, this instance will not be used in the prediction.

5. Experiment

5.1. Parameter setting for MF

In this experiment, a dataset provided by Movielens is used. This dataset consists of 100,000 instances of

form $(i, j, r_{i,j})$, where i and j represent user ID and item (movie) ID respectively, and each instance refers to a rating ranging from 1 to 5 given by a specific user out of 943 to a specific movie out of 1682, so that the sparsity of this dataset is $1 - \frac{100000}{943 \times 1682} = 93.7\%$. The first part of the experiment will be testing how the accuracy of MF will be affected by two dominant parameters, number of iterations and number of factors. The test will be implemented using *all but n* method, in which n ratings of each user will be extracted as a testing dataset, and the remaining data will be used as the training set. The number of extracted ratings n is set to 5, 10, 15, and 20 respectively. Additionally, running time with respect to different parameter values will be measured to obtain a trade-off between algorithms complexity and accuracy. In order to show results, root mean square error (*RMSE*) and mean absolute error (*MAE*) will be used to illustrate how the predicted ratings differ from the true value. During the experiments, the parameter not to be tested will be fixed when the measurement is to be taken for the other.

$$RMSE = \sqrt{\frac{\sum_{i,j}(r_{\hat{i,j}} - r_{i,j})^2}{N}}$$

$$MAE = \frac{\sum_{i,j}|r_{\hat{i,j}} - r_{i,j}|}{N}$$

5.2. MF with movie attributes

In order to improve the accuracy of the traditional MF approach, attributes of movies will be inserted into the original dataset. 19 attributes of movies are provided by MovieLens of the form $(movieID, a_1, a_2 \cdots a_i \cdots a_{19})$, where $a_i = 0$ if this movie does not have the *i-th* attribute and $a_i = 1$ otherwise. The new MFA approach will treat each attribute as a new user, as shown below. Theoretically, MFA will learn factors more accurately due to similarity and difference among 1682 movies, with the premise that the number of newly inserted attributes are not much larger than the number of ratings, otherwise the overfitting problem might be caused.

Input: 19 attributes of 1682 movies, Database D
Output: New database D with movie attributes
For each movie m
    For each attribute $a_{m,i}$ of m
        If $a_{m,i} = 0$, Insert$(943 + a_{m,i}, m, 1)$ into D
        If $a_{m,i} = 1$, Insert$(943 + a_{m,i}, m, 5)$ into D

5.3. Bias

In previous sections, the global mean of the entire training dataset is used as the bias in the SGD and prediction generation process. In order to show the influence of using each user's average rating to learn factors and give predictions, a test will be taken for 1000 times and t-test will be used to show whether the accuracy of these two settings differ. In this section, 5 ratings of each user will be extracted out as the test data, and the remaining will be the training data. The reason is that some users in the Movielens dataset have rated for only 20 movies, so that if too many ratings of these users are extracted it will be harder to explain the prediction evaluated based on their average ratings.

5.4. Combine MF with KNN

The KNN approach finds k neighbours of an object by calculating the similarity between all of the other objects with it. For example, the traditional KNN approach takes an average of the values of an objects k nearest neighbours to give a prediction. A more accurate KNN approach has been shown by another researcher in [28]. This part will modify each predicted rating $r_{i,j}$ by k nearest movies of movie v_j which have been rated by user u_i, as illustrated in Equation (12).

$$r_{\hat{i,j}} = r_{\hat{i,j}} + \sigma \cdot \sum (r_{i,n} - \bar{r}_i) \qquad \forall n \in KNN(j) \quad (12)$$

KNN of a movie will be determined by the distances between it and other movies. The calculation of distance is shown in Equation (13), where a_i and b_i are the attribute vectors of movie A and B respectively.

$$dist(A, B) = \sum_i |a_i - b_i| \qquad (13)$$

The theory behind is that if the similarity between movie A and B is relatively high, the prediction of B given by a user should be affected by the true rating of A if the same user has rated for it. The factor $(r_{i,n} - \bar{r}_i)$, which represents the user preference of movie n, is negative if the users rating for n is below his/her average rating and is positive in another case. Value of σ should be carefully chosen, otherwise the modification would increase error of the prediction.

5.5. MFAR for cold-start problem

As most other Collaborative Filtering approaches [27], cold-start is a difficult but common problem for MF. It means that if a user has rated only few movies, recommender systems might not give predictions accurately due to lack of information of this user. To deal with this problem, a threshold t will be used to

filter out most useless data, meaning that data whose prediction error exceeds the threshold will be considered to negatively affect the accuracy of prediction. More specifically, during SGD, after some number of iterations, if the absolute error of the prediction of an instance (ratings or attributes) in training dataset is still greater than t, this instance will not be used in the prediction. In this part, 500 users will be randomly chosen, and 300 of them will be placed in the training set, and n ratings of each of the left 200 users will also be inserted into the training set. The remaining ratings of these 200 users will be used as the testing set. This approach is named as Given_n, where n is set to 5, 10, 15, and 20 respectively. This test method was also used in other researches including [10] and [29]. In all of the above experiments, tests will be operated 10 times in each case to give an average to avoid 'extreme' values.

5.6. Testing platform and device

Different from Hadoop, one of the distributed frameworks, which utilized HDFS to split large data file into several blocks with some replications to store them in different data-nodes (storage part) and uses MapReduce to process the data in parallel, this paper only focuses on single machine. During the experiment, the program is run on OS X EI Capitan. The machine used is Macbook with a 2.0 GHz quad-core Intel Core i7 processor (Turbo Boost up to 3.2 GHz) with 6 MB shared L3 cache. Memory is 8 GB of 1600 MHz DDR3L, and the machine has 256 GB flash storage.

6. Results and discussion
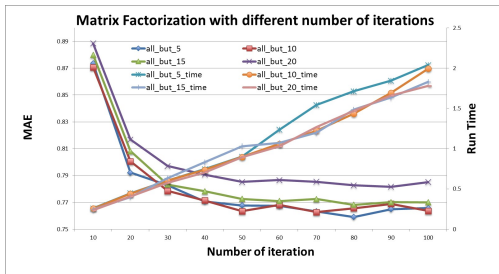
6.1. Number of iterations

Figure 2: Matrix Factorization performance with respect to number of iterations

The number of iterations is the first parameter tested to show the relationship with the recommendation quality. It is assigned from 10 to 100 with an increment of 10, and the test is based on the traditional MF approach. Figure 2 indicates that for all the number of ratings to be predicted (e.g. 5 to 20 with an increment

of 5), prediction error of MF keeps decreasing while number of iterations increases and it converges to a stationary point when number of iterations reaches 50. The operation time of MF has a steady growth rate during the rise of this parameter. Therefore, $numIter$ is set to 50 for the following experiments.
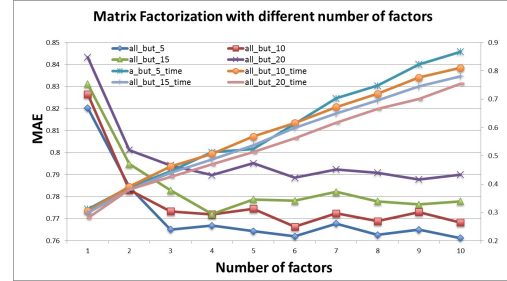

Figure 3: Matrix Factorization performance with respect to number of factors

Similarly, as illustrated in Figure 3, the accuracy of MF rises with the increase of number of factors, and running time is approximately proportional to it. According to this result, $numFactors$ is set to 5 for the following improved MF approaches. The above results show that the running time for different n values with the same parameter settings does not vary too much. It is because that the data in training set dominates the complexity of MF. If only $\frac{20 \times 943}{100000} = 18.86\%$ ratings are used as testing data, and the remaining data are used to learn factors iteratively (e.g. 50 times), there is no doubt that n value will only have insignificant influence on the operation time.

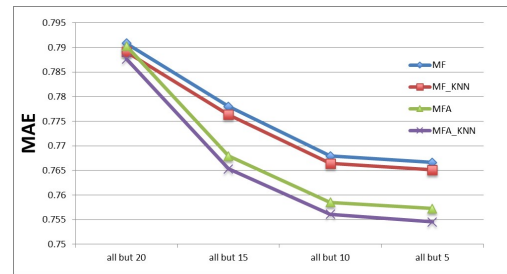### 6.2. MF with movie attributes MFA & use of KNN

Figure 4: Mean absolute error of improved Matrix Factorization approaches

After inserting attributes of movies into the training set, the prediction is supposed to be more accurate, and Figure 4 and Figure 5 display the prediction error before and after using attributes, where MF is the traditional Matrix Factorization method and MFA is the method using attributes. In all of the 4 cases where the number of data extracted for each user varies from 5
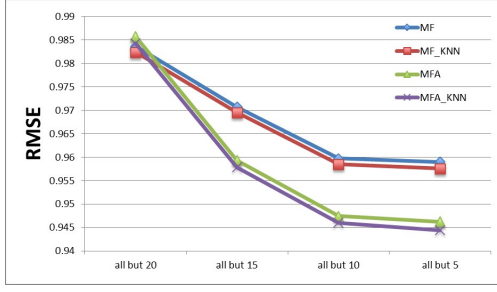
Figure 5: Root mean square error of improved Matrix Factorization approaches

Table I: t-test for KNN

t-Test: Two-Sample Assuming Unequal Variances		
	MF	MF with KNN
Mean	0.762332362	0.760927889
Variance	0.000135418	0.000139141
Observations	1000	1000
Hypothesized Mean Difference	0 (test whether two samples are equal)	
t Stat	2.680376141	
t Critical two-tail	1.961152015	

to 20, MFA always gives better prediction than MF. In addition, K = 10 and = 0.06 are the configurations for combining MF and MFA with KNN method, which use a specific movies most similar movies to modify predicted ratings. The figures indicate that the prediction error decrease by a small number when KNN is used for further modification. Table I shows the results of t-test which takes 1000 samples of MF and MF with KNN respectively. With a confidence level of 95%, t value (2.680376141) is greater than the critical t value (1.961152015), which means that the prediction error of MF 'really' exceeds the error of MF with KNN. It provides an evidence that the similarity of items will similarly attract a user to some extent.

6.3. Bias

In order to show whether biases in Equation (10) and (11) have different impacts on the accuracy, 1000 samples were used to perform t-test. With a confidence level of 95%, as shown in TABLE II, t value of the difference in two mean values is less than the negative critical t value (-1.9612), which means that

Table II: t-test for bias

t-Test: Two-Sample Assuming Unequal Variances		
	MF user mean in SGD	MF global mean in SGD
Mean	0.755682927	0.762332362
Variance	0.000134362	0.000135418
Observations	1000	1000
Hypothesized Mean Difference	0 (test whether two samples are equal)	
t Stat	-12.80206157	
t Critical two-tail	1.961152015	

using each user' mean rating in SGD and prediction generation can "really" improve the accuracy. However, one shortcoming of using each user mean is that when the test was performed for all but 20 and all but 15, the result was not as satisfactory as using global mean. So, due to the poor stability, whether user mean or global mean should be used needs to be carefully decided.
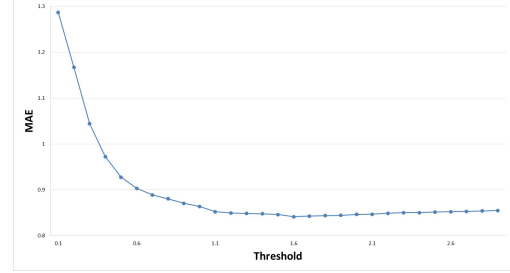
6.4. MFAR for cold-start problem

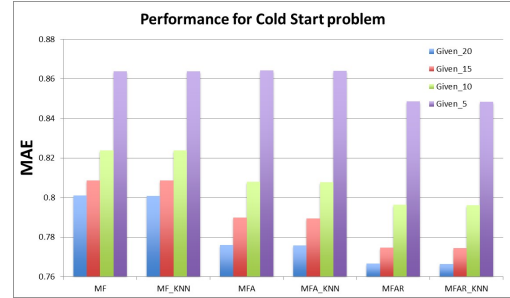Figure 6: Impact of different values of threshold

Figure 7: Matrix Factorization approaches dealing with cold-start problem, shown in MAE
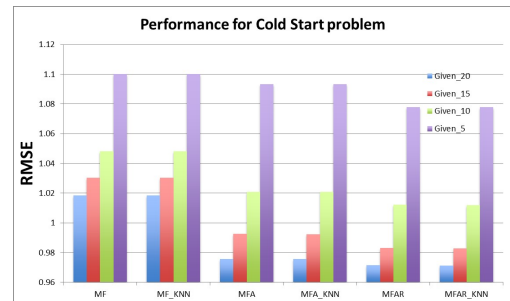

Figure 8: Matrix Factorization approaches dealing with cold-start problem, shown in RMSE

Firstly, in order to learn the influence of different threshold on prediction accuracy, the number of given ratings of each user is set to 5. During the test, threshold varies from 0.1 to 2.9 with an increment of 0.1, and the experiment is averaged over 10 iterations. As shown in Figure 6, MAE drops dramatically at the beginning and reaches the minimum value when threshold is between

Table III: Comparison with others' approaches

Approach	all but 10	training 80%
MFA with KNN	0.7560981	0.748411
PCA-GAKM [10]	0.7821	–
Pearson Correlation Similarity [18]	–	0.828
Euclidean Distance Similarity [18]	–	0.818
Log Likelihood [18]	–	0.817
Tanimoto Similarity [18]	–	0.794

Table IV: Comparison with others' approaches for cold-start problem

	Given_5	Given_10	Given_15	Given_20
MFAR	0.8485	0.7961	0.7743	0.7663
[10] approx.	0.90	0.82	0.79	0.77

1.5 and 2.0. The reason is that when threshold has small value (e.g. 0.1 to 1.0), most of the ratings in training dataset will be considered as *misleading*, and consequently there will not be enough training data to accurately learn the compulsory factors. Additionally, when threshold reaches large value (e.g. larger than 2.5), more *misleading* ratings will not be filtered out because the prediction error of these ratings will not exceed the threshold. Therefore, compared to the number of ratings of each user provided (i.e. 5), large amount of movie attributes will mislead the factors learned by SGD. According to the result shown in Figure 6, the threshold is set to 1.7 in the following experiments.

When MF is used to deal with cold-start problems, the number of given ratings of each user is set to 5, 10, 15, and 20 respectively. As shown by Figure 7 and Figure 8, combination of MF and KNN fails to make contributions to the accuracy of prediction, which might be because that the extreme small number of given ratings in training set fail to provide nearest neighbours of predicted movies. For Given_20, Given_15, and Given_10, MAE of MF, MFA and MFAR decrease progressively, but for Given_5, MAE of MF and MFA is almost the same. RMSE has a similar trend, with MFAR providing the most accurate prediction.

6.5. Comparison with others' algorithm

It is convenient to compare the proposed MF algorithm with other researchers' approaches such as [10] and [18], as they also used the the Movielens 100K dataset and showed their results in RMSE and MAE.

As shown in Table III, authors of [10] provided the precise value of their PCA-GAKM method for *all but 10* test, and in comparison, the proposed MFA-KNN method has better performance. Dheeraj, Sheetal and Debajuoti measured their algorithms under the condition that the ratio of training set and testing set was 80%:20% [18], and MFA-KNN also has more qualified prediction quality. Table IV compares the accuracy of MFAR with PCA-GAKM. MFAR always has more accurate prediction than PCA-GAKM regardless of the number of ratings provided by each user.

7. Conclusion

In this paper improved Matrix Factorization methods are developed to give more accurate predictions. The traditional MF approach is firstly studied to show its performance with respect to number of iterations and number of factors. Attributes of movies are then inserted into the training set in order to help the algorithm in learning factors, and the results show that the insertion can obviously increase accuracy of rating prediction. Also, since similar movies might have close attractiveness to users, k-nearest-neighbour approach is implemented to further improve the prediction, and the experiment supports this hypothesis. Finally, to deal with cold-start problem, which is one of the most common problems for Collaborative Filtering, a reduction approach is used to filter out the most useless in the training set. The carefully selected threshold t proved to effectively reduce the prediction error for those who have insufficient ratings (5-20 during the experiment).

Our ongoing work includes adjustments (or extensions, if needed) to our proposed method to make it generic for a variety of prediction applications. We are also interested in combining our method with other work on non-negative Matrix Factorization to achieve more accurate predictions.

References

[1] A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub, and I. Netanely, "Movie recommender system for profit maximization," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 121–128.

[2] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*. Springer, 2007, pp. 325–341.

[3] D. Bokde, S. Girase, and D. Mukhopadhyay, "Matrix factorization model in collaborative filtering algorithms: A survey," 2015.

[4] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *AAAI/IAAI*, 2002, pp. 187–192.

[5] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 2008, pp. 263–272.

[6] H. Shan and A. Banerjee, "Generalized probabilistic matrix factorizations for collaborative filtering," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1025–1030.

[7] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic, "One-class matrix completion with low-density factorizations," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1055–1060.

[8] 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*. ACM, 2008, pp. 426–434.

[9] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Mymedialite: A free recommender system library," in *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 2011, pp. 305–308.

[10] Z. Wang, X. Yu, N. Feng, and Z. Wang, "An improved collaborative movie recommendation system using computational intelligence," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 667–675, 2014.

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

[12] J. Wang, A. P. De Vries, and M. J. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 501–508.

[13] H. J. Ahn, "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem," *Information Sciences*, vol. 178, no. 1, pp. 37–51, 2008.

[14] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, "Probabilistic memory-based collaborative filtering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 1, pp. 56–69, 2004.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[16] L. H. Ungar and D. P. Foster, "Clustering methods for collaborative filtering," in *AAAI workshop on recommendation systems*, vol. 1, 1998, pp. 114–129.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Fifth International Conference on Computer and Information Science*. Citeseer, 2002, pp. 27–28.

[18] D. kumar Bokde, S. Girase, and D. Mukhopadhyay, "An item-based collaborative filtering using dimensionality reduction techniques on mahout framework," *CoRR*, 2015.

[19] T. D. Nielsen and F. V. Jensen, *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[20] C. Ono, M. Kurokawa, Y. Motomura, and H. Asoh, "A context-aware movie preference model using a bayesian network for recommendation and promotion," in *User Modeling 2007*. Springer, 2007, pp. 247–257.

[21] T. Hong and D. Tsamis, "Use of knn for the netflix prize," *CS229 Projects*, 2006.

[22] P. G. Campos, A. Bellogín, F. Díez, and J. E. Chavarriaga, "Simple time-biased knn-based recommendations," in *Proceedings of the Workshop on Context-Aware Movie Recommendation*. ACM, 2010, pp. 20–23.

[23] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[24] T. S. Ferguson, "An inconsistent maximum likelihood estimate," *Journal of the American Statistical Association*, vol. 77, no. 380, pp. 831–834, 1982.

[25] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.

[26] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[27] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2002, pp. 253–260.

[28] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering," in *Proceedings of the fifth international conference on computer and information technology*, vol. 1. Citeseer, 2002.

[29] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 39–46.