2015 IEEE 29th International Conference on Advanced Information Networking and Applications

Item Similarity Learning Methods for Collaborative Filtering Recommender Systems

Feng Xie[§†], Zhen Chen[†‡], Jiaxing Shang[§], Wenliang Huang[*] and Jun Li[†‡]

[§]Department of Automation, Tsinghua University, Beijing, 100084, China
[†]Research Institute of Information Technology, Tsinghua University, Beijing, 100084, China
[‡]Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing, 100084, China
[*]China Unicom Groups, Beijing, 100140, China
{xief10, shangjx06}@mails.tsinghua.edu.cn, {zhenchen, junl}@tsinghua.edu.cn

***Abstract*—As one of the most popular recommender technologies, Collaborative Filtering (CF) has been widely deployed in industry due to its simplicity and interpretability. However, it is facing great challenge to generate accurate similarities between users or items because of data sparsity. This will cause second-order error in the process of using weighted sum as prediction. To alleviate this problem, we propose several methods to learn more accurate item similarities by minimizing the squared prediction error. This optimization problem is solved using Stochastic Gradient Descent. A comprehensive set of experiments on two real-world datasets at error and classification metrics indicate that the proposed methods can achieve comparable or even better performance than other state-of-the-art recommendation methods of Matrix Factorization, and greatly outperform traditional item based CF method. Besides, the proposed methods inherit the interpretability of item based CF, which makes the recommended results more accessible compared to competing methods of Matrix Factorization.**

***Keywords*—Recommender Systems, Collaborative Filtering, Similarity Measurement, Matrix Factorization, Stochastic Gradient Descent.**

I. Introduction

Recommender systems (RS) bring great convenience to people's life by helping them find the most relevant content from seriously overloaded Web. Nowadays, RSs have been successfully used in many fields, such as video (e.g., Netflix, YouTube), music (e.g., Pandora), book (e.g., Amazon), news (e.g., Digg) etc. Over the years, massive algorithms have been developed to address the recommendation problem [1][2]. These algorithms make use of the user explicit (e.g. rating) or implicit (e.g. click-through, purchase and review) feedback to construct the user interest model and then make recommendations.

As one of the most promising recommender methods [3], collaborative filtering (CF) anticipates user's interests by considering the opinions of those who have similar preferences. Compared to other techniques (e.g., content based methods [4][5]), typically, CF based methods act only on a user-item rating matrix which is represented by the feedback information. Besides, CF based methods have the capability to expose unexpected items to users, which are not similar to those they have chosen before. This makes them work well in domains where the attribute content of items is difficult to parse, such as musics and videos.

CF based methods can be further classified into two classes: memory based CF methods [6] and model based CF methods [7][8][9][10]. The former finds similar users (or items) for the active user (or item) using similarity measurement methods, and then aggregate the ratings of these neighbours as the prediction. Memory based CF contains two popular methods, user based CF [11] and item based CF [12], depending on whether the neighbours are derived by identifying similar users or items. Due to its simplicity and reasonably accurate recommendations, memory based CF has been widely used in industry. However, it suffers from several problems, including data sparsity [13], cold start [14] and data correlation [15], where each user express preference to only a small subset of the available items, and users tend to rate similar items closely. Therefore, the similarities between users or items cannot be accurately measured by the existing similarity measurement methods, such as Cosine and Pearson Correlation, which will result in inaccurate predictions. To alleviate this problem, many model based methods are proposed, such as Bayesian belief nets CF models [7], clustering CF models [8], Markov decision process based CF models [9] and latent semantic CF models [10]. However, some of these models are not applicable because of their complexity, such as the estimation of multiple parameters and sensitivity to the statistical dataset properties. In addition, matrix factorization, such as [16], aims to alleviate this problem by reducing the dimensions of user-item rating matrix, then the implicit relationships between items (even those have not been co-rated by one user) can be captured. The outstanding performance makes matrix factorization be considered a state-of-the-art method in rating prediction, but it introduces high computational complexity and also faces the problem of uninterpretable recommendations.

To improve the performance of memory based CF without discarding its advantages, new similarity measurement method based on user or item rating statistics was proposed in [17] and Grey Forecast model was used for rating prediction regardless of the similarities [18].

In this paper, we propose several methods to learn more accurate item similarities by minimizing the squared prediction error. Then, the prediction is made by the weighted sum of the active user's ratings to similar items with learned similarities. The extensive experiments are conducted on two real-word datasets, and the results demonstrate that the proposed methods can greatly outperform traditional item based CF and achieve

1550-445X/15 $31.00 © 2015 IEEE
DOI 10.1109/AINA.2015.285

comparative or even better performance than the state-of-the-art method, matrix factorization. Besides, the proposed methods inherit the interpretability from item based CF, which make the recommendations more accessible to users.

The rest of the paper is organized as follows. Section II introduces the notations, the existing similarity measurement methods and motivation. In Section III, the proposed item similarity learning methods are presented detailedly. Sections IV and V provides the experiment design and result analysis respectively. Followed by the final section, which provides some concluding remarks.

II. Related Work

In the typical item based CF recommendation scenario, user's implicit or explicit preferences to certain item can be modelled as a rating. High rating suggests the user is satisfied with this item. If we denote m as the number of users and n as the number of items, the ratings of these users on items can be integrated into a user-item rating matrix $\boldsymbol{R}^{m\times n}$, where the rating of user $u(u = 1, 2, \ldots, m)$ on item $i(i = 1, 2, \ldots, n)$ can be denoted by $r_{u,i}$. $r_{u,i}$ is set to 0 when user u has not yet rated item i. Further, $U(i) = \{u|r_{u,i} \neq 0; u = 1, 2, \ldots, m\}$ is the set of users who have given ratings to item i. Similarly, $I(u) = \{i|r_{u,i} \neq 0; i = 1, 2, \ldots, n\}$ contains all items rated by user u. The critical step of item based CF is to calculate similarity between two items i and j with notation of $sim_{i,j}$, thus, all item pairs' similarities produce a item similarity matrix $\boldsymbol{Sim}^{n\times n}$. Then, the $k(k < n)$ most similar items for each item i are chosen as its k nearest neighbourhood $N_k(i)$. The k nearest neighbourhoods for all items will form the item neighbourhood matrix $\boldsymbol{Sim}^{n\times k}$. Finally, the prediction of the given user u on unrated item i is the weighted sum of the ratings of u on the items in $N_k(i)$ [19]. That is

$$\hat{r}_{u,i} = \frac{\sum\limits_{j\in N_k(i)\cap I(u)} sim_{i,j} \cdot r_{u,j}}{\sum\limits_{j\in N_k(i)\cap I(u)} |sim_{i,j}|} \tag{1}$$

where $\hat{r}_{u,i}$ is the prediction. Therefore, the accuracy of item based CF depends greatly on the quality of the similarity measure. Nowadays, many different methods have been proposed to compute similarity between items.

A. Cosine based Similarity

The basic idea to accurately measure the similarity between item i and item j is first to find out the users who have rated both of these items, and then the ratings for item i or j given by these users form an item rating vector. The vector cosine between two item rating vectors is applied to determine the similarity. Therefore, the cosine based similarity [13] (denoted by $cos_{i,j}$) between item i and item j is

$$cos_{i,j} = \frac{\sum\limits_{u\in U(i)\cap U(j)} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum\limits_{u\in U(i)\cap U(j)} r_{u,i}^2}\sqrt{\sum\limits_{u\in U(i)\cap U(j)} r_{u,j}^2}} \tag{2}$$

B. Pearson Correlation based Similarity

Since some items tend to get higher ratings than others, it is necessary to eliminate item means before computing similarity. Therefore, following cosine based similarity, Pearson correlation [12] measures the extent to which two item rating vectors linearly relate with each other

$$pear_{i,j} = \frac{\sum\limits_{u\in U(i)\cap U(j)} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum\limits_{u\in U(i)\cap U(j)} (r_{u,i} - \bar{r}_i)^2}\sqrt{\sum\limits_{u\in U(i)\cap U(j)} (r_{u,j} - \bar{r}_j)^2}} \tag{3}$$

where $\bar{r}_i$ and $\bar{r}_j$ are the average ratings of items i and j, respectively. $pear_{i,j}$ is the Pearson correlation based similarity between item i and item j.

C. Jaccard Coefficient based Similarity

In contrast with the aforementioned methods, Jaccard Coefficient [20] does not care about the exact ratings. It measures the number of users who have given ratings to both items compared to the number of users who have rated either of them

$$Jaccard_{i,j} = \frac{U(i) \cap U(j)}{U(i) \cup U(j)} \tag{4}$$

where $Jaccard_{i,j}$ is the Jaccard coefficient based similarity of items i and j.

D. Motivation

In real world scenarios, users typically express preference (click-through, purchase or rating) to only a handful of items out of thousands of items. Such sparse user-item rating matrix makes traditional methods like item based CF fail to measure the similarities between items that have not been co-rated by at least one user. Intuitively, the similarity between item i and item j will have $sim_{i,j} = 0$, if they have not been co-rated by at least one user. However, firstly, these two items can be similar to each other by the third item which is similar to both of them due to similarity transitivity [21]. Secondly, the similarity is not accurate when there are not enough co-rated users. Besides, due to systematic tendencies for some users like to give higher ratings than others, and for some items analogously (i.e. Pearson correlation attempts to overcome this advantage by subtracting user or item mean), the inherent error will exist using aforementioned similarity measurement methods. Consequently, the item similarity matrix generated will be inaccurate. Therefore, using these similarities as weight for prediction (see Eq.(1)) will bring second-order error.

However, due to its intuitive nature, item based CF method is amenable to explaining. This is because users are easier to accept items which are similar to those previously preferred by them. In addition, item based CF method is relatively simple to implement, which makes it favourable in many cases despite mediocre accuracy. This leads us to improve item based CF method by proposing item similarity learning methods which can overcome the aforementioned advantages.

Algorithm 1 Learning method with similarity as initial weight

1: **procedure** SIMCF($\boldsymbol{Sim}^{n\times k}$)
2: $iter \leftarrow 0$
3: $\gamma \leftarrow$ learning rate
4: $\lambda \leftarrow \ell_F$ regularization weight
5: $\boldsymbol{R}^{m\times n} \leftarrow$ Training Set
6: Init $\boldsymbol{\Omega}^{n\times k}$ with $\boldsymbol{Sim}^{n\times k}$
7:
8: **while** $iter < maxIter$ or error on training set decreases **do**
9: **for all** $r_{u,i} \in \boldsymbol{R}^{m\times n}$ **do**
10: $\hat{r}_{u,i} \leftarrow \sum_{j\in N_k(i)\cap I(u)} \omega_{i,j} \cdot r_{u,j}$
11: $e_{u,i} \leftarrow r_{u,i} - \hat{r}_{u,i}$
12: **for all** $j \in N_k(i) \cap I(u)$ **do**
13: $\omega_{i,j} \leftarrow \omega_{i,j} - \gamma(-e_{u,i} \cdot r_{u,i} + \lambda\omega_{i,j})$
14: **end for**
15:
16: **end for**
17:
18: $iter \leftarrow iter + 1$
19: **end while**
20:
21: **return** $\boldsymbol{\Omega}^{n\times k}$
22: **end procedure**

III. Item Similarity Learning Methods

In the item based CF, the prediction $\hat{r}_{u,i}$ for a user u on an unrated item i is calculated as an aggregation of the items that have been rated by u (see Eq.(1). If $\omega_{i,j}$ is assigned to be the normalization of similarity with $\omega_{i,j} = sim_{i,j} / \sum_{j\in N_k(i)\cap I(u)} |sim_{i,j}|$, this equation is equivalent to following one:

$$\hat{r}_{u,i} = \sum_{j\in N_k(i)\cap I(u)} \omega_{i,j} \cdot r_{u,j} \tag{5}$$

To obtain more accurate prediction, $\omega_{i,j}$ will be optimized by the proposed item similarity learning methods. After learning the similarity for each item pair, an improved item similarity matrix will be generated (denoted by $\boldsymbol{\Omega}$). Since $N_k(i) \cap I(u)$ is k nearest neighbours of item i that have been rated by user u, $\boldsymbol{\Omega}$ is a $n \times k$ matrix which can be further denoted by $\boldsymbol{\Omega}^{n\times k}$.

The squared prediction error is used as loss function to compute the loss, which is given by

$$\mathcal{L}(\cdot) = \sum_{r_{u,i}\in \boldsymbol{R}^{m\times n}} \|r_{u,i} - \hat{r}_{u,i}\|_F^2 \tag{6}$$

where $r_{u,i}$ and $\hat{r}_{u,i}$ are the ground truth value and the prediction respectively. The prediction $\hat{r}_{u,i}$ for a given user u on item i is estimated as Eq.(5). Therefore, the item similarity matrix $\boldsymbol{\Omega}^{n\times k}$ is learned by minimizing the following regularized optimization problem

$$\underset{\boldsymbol{\Omega}^{n\times k}}{minimize} \quad \frac{1}{2} \sum_{r_{u,i}\in \boldsymbol{R}^{m\times n}} \|r_{u,i} - \hat{r}_{u,i}\|_F^2 + \frac{\lambda}{2} \left\|\boldsymbol{\Omega}^{n\times k}\right\|_F^2 \tag{7}$$

where the regularization term is used to prevent overfitting and λ is the regularization weight for learned item similarity matrix. The Stochastic Gradient Descent (SGD) algorithm [22] is applied to solve the optimization problem of Eq.(7). We developed three item similarity learning methods that use different initial values for $\boldsymbol{\Omega}^{n\times k}$.

A. SimCF-Similarity as Initial Weight

Following the common practices for top-N recommendation [23][24], the loss function in Eq.(6) is computed over only all rated entries of user-item rating matrix $\boldsymbol{R}^{m\times n}$. Note that before the learning procedure, one of the similarity measurement methods introduced in section (II) is used to roughly generate the item similarity matrix $\boldsymbol{Sim}^{n\times n}$. Then, in order to reduce the computational requirements for optimization, k nearest neighbours are chosen for each item to form the k nearest item neighbour similarity matrix $\boldsymbol{Sim}^{n\times k}$. Algorithm 1 provides the detailed gradient update rules. $\boldsymbol{\Omega}^{n\times k}$ is initialized with the k nearest item neighbour similarity matrix $\boldsymbol{Sim}^{n\times k}$ as the initial prediction (see line 6). The iteration is repeated until the number of iterations has reached the predefined threshold or the error decrease on the training set reaches the predefined resolution.

B. RanCF-Random Value as Initial Weight

Following the procedure of SimCF, RanCF also calculates the item similarity matrix firstly. With the same purpose to reduce the computation overhead for optimization, only k nearest neighbours for each item are chosen to generate $\boldsymbol{Sim}^{n\times k}$. $\boldsymbol{\Omega}^{n\times k}$ initialized with $\boldsymbol{Sim}^{n\times k}$, but the values are replaced by random values in (0,1) as the initial prediction (see line 6). This is in contrast with SimCF, which use the similarities as weights for initial prediction. Algorithm 2 provides the detailed procedure of RanCF.

C. ConCF-Constant Value as Initial Weight

The SGD algorithm may get local optimums due to inappropriate initial values. Therefore, we developed ConCF with constant value 0.1 as the initial prediction. That is, once $\boldsymbol{\Omega}^{n\times k}$ is initialized with $\boldsymbol{Sim}^{n\times k}$, the values are replaced by constant value 0.1 (see line 6). Other process is the same to SimCF and RanCF. Algorithm 3 illustrates the whole procedure and the gradient update rules.

IV. Experimental Evaluation

A. Data Sets

The performance of the proposed item similarity learning methods is evaluated on two different real datasets, namely MovieLens100k and MovieLens1M. Both of them are the subsets of data collected by Grouplens research from the MovieLens Web site (http://movielens.umn.edu). MovieLens100k consists of 100,000 ratings (in the scale of 1-5 stars) of 943 users on 1682 movies. MovieLens1M is much larger than MovieLens100k, which consists of one million ratings given by 6040 users on 3952 movies. Table I summaries the characteristics of all the datasets.

Algorithm 2 Learning method with random value as initial weight

procedure RANCF($\boldsymbol{Sim}^{n\times k}$)
  $iter \leftarrow 0$
  $\gamma \leftarrow$ learning rate
  $\lambda \leftarrow \ell_F$ regularization weight
  $\boldsymbol{R}^{m\times n} \leftarrow$ Training Set
  Init $\boldsymbol{\Omega}^{n\times k}$ with $\boldsymbol{Sim}^{n\times k}$ and then replaced with random values in (0,1)

  while $iter < maxIter$ or error on training set decreases **do**
    for all $r_{u,i} \in \boldsymbol{R}^{m\times n}$ **do**
      $\hat{r}_{u,i} \leftarrow \sum_{j\in N_k(i)\cap I(u)} \omega_{i,j} \cdot r_{u,j}$
      $e_{u,i} \leftarrow r_{u,i} - \hat{r}_{u,i}$
      for all $j \in N_k(i) \cap I(u)$ **do**
        $\omega_{i,j} \leftarrow \omega_{i,j} - \gamma(-e_{u,i} \cdot r_{u,i} + \lambda\omega_{i,j})$
      end for

    end for

    $iter \leftarrow iter + 1$
  end while

  return $\boldsymbol{\Omega}^{n\times k}$
end procedure

Algorithm 3 Learning method with constant value as initial weight

procedure CONCF($\boldsymbol{Sim}^{n\times k}$)
  $iter \leftarrow 0$
  $\gamma \leftarrow$ learning rate
  $\lambda \leftarrow \ell_F$ regularization weight
  $\boldsymbol{R}^{m\times n} \leftarrow$ Training Set
  Init $\boldsymbol{\Omega}^{n\times k}$ with $\boldsymbol{Sim}^{n\times k}$ and then replaced with constant value 0.1

  while $iter < maxIter$ or error on training set decreases **do**
    for all $r_{u,i} \in \boldsymbol{R}^{m\times n}$ **do**
      $\hat{r}_{u,i} \leftarrow \sum_{j\in N_k(i)\cap I(u)} \omega_{i,j} \cdot r_{u,j}$
      $e_{u,i} \leftarrow r_{u,i} - \hat{r}_{u,i}$
      for all $j \in N_k(i) \cap I(u)$ **do**
        $\omega_{i,j} \leftarrow \omega_{i,j} - \gamma(-e_{u,i} \cdot r_{u,i} + \lambda\omega_{i,j})$
      end for

    end for

    $iter \leftarrow iter + 1$
  end while

  return $\boldsymbol{\Omega}^{n\times k}$
end procedure

TABLE I: Characteristics of datasets

Dataset	♯Users	♯Items	♯Ratings	Ruser	Ritem	Density
MovieLens100k	943	1682	100,000	106.04	59.45	6.30%
MovieLens1M	6040	3952	1,000,000	165.56	253.04	4.19%

The "♯Users", "♯Items" and "♯Ratings" are the number of users, items and ratings respectively included in each of the datasets. The "Ruser" and "Ritem" are the average number of ratings for each user and each item respectively. The "Density" measures the density of each dataset with Density = ♯Ratings/(♯Users × ♯Items).

B. Evaluation Methodology

To evaluate the performance of the proposed methods, the data set is randomly split into training set and test set. In this paper, training set contains nearly 80% ratings of the data set, while the rest of the data is used as the test set. The proposed methods learn better item similarity to improve the accuracy of item based CF by minimizing the rating prediction error. Therefore, two frequently used error metrics, Mean Absolute Error (MAE) [18] and Root Mean Square Error (RMSE) [18], are applied to evaluate the error between ground truth value and predicted value. MAE and RMSE are defined as follows

$$MAE = \frac{\sum_{(u,i)\in T} |r_{u,i} - \hat{r}_{u,i}|}{|T|} \tag{8}$$

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in T} (r_{u,i} - \hat{r}_{u,i})^2}{|T|}} \tag{9}$$

where T contains all user-item pairs in the test set. $|T|$ is the number of (u,i) pairs.

In the top-N recommendation scenario, we care more about suggesting a short list of items to the given user than predicting precisely. Therefore, two maturely used metrics in information retrieval, precision and recall [23], are applied to measure the classification accuracy of recommender systems. Since an item can be either interesting or uninteresting to the given user, there are four possibilities between the ground truth value and predicted value, namely, true positive (TP), false positive (FP), true negative (TN) and false negative (FN). If we define a rating $r_{u,i}$ is less than the set threshold t, it suggests that user u is not interested in item i, and vice versa. Then true positive can be denoted by $TP = \{(u,i)|(u,i) \in T, r_{u,i} \geq t, \hat{r}_{u,i} \geq t\}$, which contains all interesting (u,i) pairs in test set that are classified correctly, while true negative contains all uninteresting (u,i) pairs that are classified correctly, where $TN = \{(u,i)|(u,i) \in T, r_{u,i} < t, \hat{r}_{u,i} < t\}$. Similarly, false positive is the set of uninteresting (u,i) pairs that are classified into interesting ones, witch can be denoted by $FP = \{(u,i)|(u,i) \in T, r_{u,i} < t, \hat{r}_{u,i} \geq t\}$. The others are included in $FN = \{(u,i)|(u,i) \in T, r_{u,i} \geq t, \hat{r}_{u,i} < t\}$. Here, we set $t = 3$. Consequently, precision and recall can be defined as follows

$$precision = \frac{|TP|}{|TP| + |FP|} \tag{10}$$

$$recall = \frac{|TP|}{|TP| + |FN|} \tag{11}$$

where "| |" is the size of the set.

Often, the relationship between precision and recall is inverse. Therefore, a combination between precision and recall is usually used to better understand this type of recommendation quality, which is named F-measure [18]

$$F\text{-}measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{12}$$

C. Comparison Algorithms

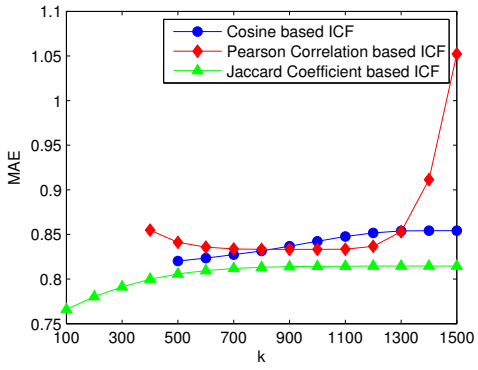
The performance of the proposed methods is compared against that achieved by item based CF (ICF) [12], unweighted ICF (UICF), SlopeOne [25], SVD+ [16] and SVD++ [16]. With different similarity methods to calculate similarity between items, there are several variants of ICF, namely Cosine based ICF, Pearson Correlation based ICF and Jaccard Coefficient based ICF. UICF uses similarities to find k most similar items for each item, and then takes the average as the prediction rather than the weighted average. Similarly, UICF also has three variants. SVD+ and SVD++ are two variants of matrix factorization, the former is closely related to PureSVD [23], but it uses SGD for minimizing the regularized squared error on the training set. SVD++ is an improved version of SVD+ by taking user and item biases into account. This set of methods constitute the current state-of-the-art for rating prediction task in recommendation. Therefore, they form a good set of methods to evaluate our proposed methods.

V. Experimental Results

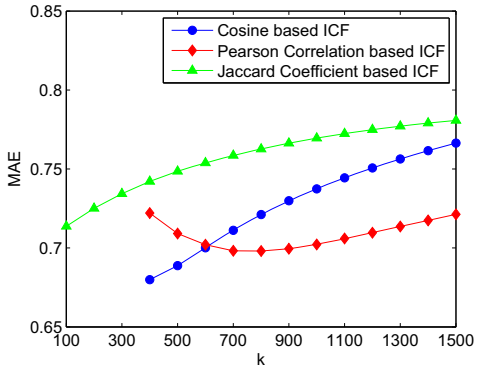
The experimental evaluation consists of three parts. First, we study the effect of the number of nearest neighbours and various similarity methods of our proposed approaches on the prediction accuracy. Second, the effect of various item similarity learning methods is studied. Finally, due to the lack of space and the same results and conclusions carried over to two datasets, MovieLens1M is chosen to represent the comparison results with other competing methods (Section IV-C) on both error and classification metrics.

A. Effect of the Number of Nearest Neighbours

When the number of nearest neighbours k is set to be small, the prediction may be failed or inaccurate due to the lack of items rated by the given user in this small neighbourhood. Particularly, if user u did not rate at least one item of i's k most similar items, (u, i) pair in test set can not be predicted. Here, we define the predictable pairs in proportion to all candidate pairs as the predict rate. Since the results are meaningful only when the predict rate is high enough, in this paper, the thresholds are 99% and 90% for MovieLens100k and MovieLens1M respectively due to the difference of data sparsity (see Table I). Fig. 1 shows the performance comparison of ICF with three similarity methods. The Jaccard Coefficient based ICF can achieve high predict rate with smaller k than the others. This is because the popular items always get the high Jaccard Coefficient similarities, and these items have high probabilities to be rated by the given user. Moreover, ICF performs difference with different similarity methods, and the results tend to be worse as k increases to a certain size since the rating with low similarity becomes noise data.



(a) MovieLens100k dataset



(b) MovieLens1M dataset

Fig. 1: The MAE comparison of ICF with different similarity methods.

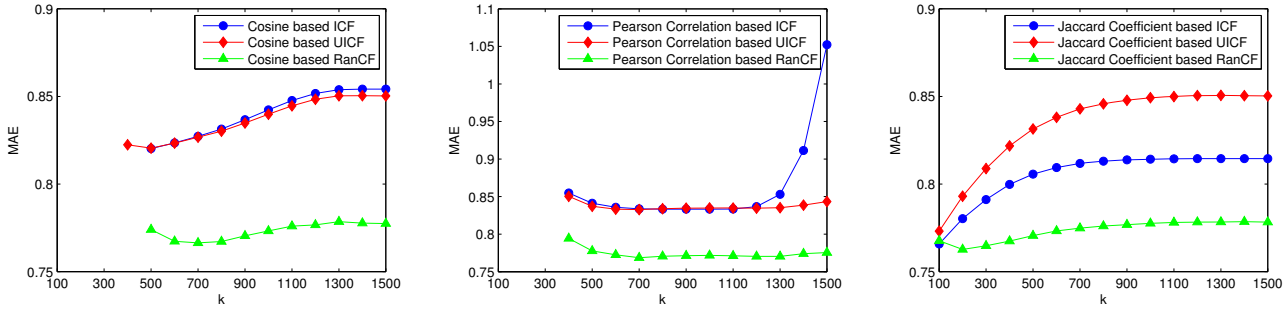
Take RanCF as an example, the performance compared against that achieved by ICF and UICF with different similarity methods are illustrated in Fig. 2 and Fig. 3. It shows that the proposed method can outperform the traditional ICF no matter which similarity method is based on. To reveal the improvement more clearly and fair, the optimum result for each method is chosen. For example, the optimum MAE value is 0.60 for Cosine based RanCF, and 0.68 for Cosine based ICF, on MovieLens1M dataset. Therefore, the improvement is over 10% ($(0.68 - 0.6)/0.68 = 11.76\%$) in terms of MAE. The results are illustrated in Fig. 4. The same results and conclusions are reached by SimCF and ConCF. Besides, we can find that UICF achieves comparatively even better performance than ICF, which suggests that the similarities calculated by these popular similarity measurement methods are inaccurate. This motivates us to improve the ICF by learning more accurate item similarities.

B. Effect of Similarity Methods

Since different similarity methods may generate different k nearest neighbours for each item, the initial weight matrix $\mathbf{\Omega}^{n \times k}$ for item similarity learning methods will not be the same (Because $\mathbf{\Omega}^{n \times k}$ is replaced by $\boldsymbol{Sim}^{n \times k}$ firstly.). Intuitively, the results achieved by the proposed approaches may be quite different. However, Table II shows that RanCF

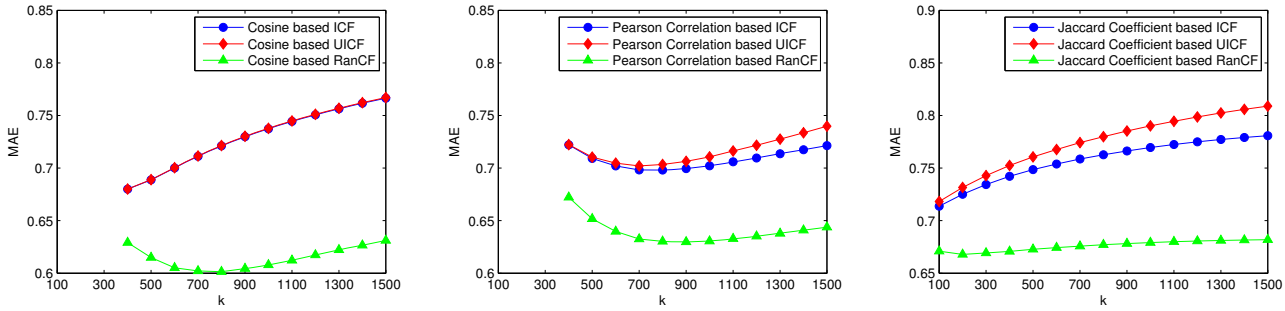
TABLE II: The MAE comparison of RanCF with three similarity methods generating k neaerest neighbours

Dataset	k	100	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400	1500
MoiveLens100k	Cosine					0.774	0.767	**0.766**	0.767	0.770	0.773	0.776	0.777	0.779	0.778	0.777
	Pearson				0.794	0.778	0.773	**0.769**	0.771	0.771	0.772	0.771	0.770	0.770	0.774	0.775
	Jaccard	0.768	**0.763**	0.765	0.767	0.771	0.773	0.775	0.776	0.777	0.778	0.778	0.778	0.778	0.779	0.778
MoiveLens1M	Cosine				0.629	0.615	0.605	0.602	**0.601**	0.604	0.608	0.612	0.617	0.622	0.627	0.631
	Pearson				0.672	0.652	0.640	0.632	**0.630**	0.630	0.631	0.633	0.635	0.638	0.641	0.644
	Jaccard	0.671	**0.668**	0.669	0.671	0.673	0.674	0.676	0.677	0.678	0.679	0.680	0.681	0.681	0.682	0.682



(a) Cosine based (b) Pearson Correlation based (c) Jaccard Coefficient based

Fig. 2: The MAE comparison of RanCF, ICF and UICF with different similarity methods on MovieLens100k dataset.



(a) Cosine based (b) Pearson Correlation based (c) Jaccard Coefficient based

Fig. 3: The MAE comparison of RanCF, ICF and UICF with different similarity methods on MovieLens1M dataset.

performs slightly difference (especially on MovieLens100k dataset) with different similarity methods generating k nearest neighbours. This is because the proposed methods aim to learn more accurate similarities by minimizing the prediction error regardless of the similarity values calculated using the similarity methods introduced in Section II.

Besides, as k increases, the MAE value falls down and then up. The reason is that small neighbourhood does not have enough rated items for prediction, on the contrary, big neighbourhood will bring noise data which may decrease the performance. The optimum k for Cosine and Pearson Correlation based RanCF are much bigger than the one of Jaccard Coefficient based RanCF. It is still because Jaccard Coefficient similarity method tend to give higher similarities to popular items which have higher probabilities to be rated by users. Therefore, Jaccard Coefficient based RanCF can easily get enough data to make optimum prediction even with small k. However, Jaccard Coefficient based RanCF performs worse than other two RanCF variants. This appears more obvious on the sparser dataset, MovieLens1M. The results of SimCF and ConCF are not given because of the same conclusion reached and lack of space.

C. *Effect of Initial Weight*

Since different similarity methods will generate different k nearest neighbours, the items used for prediction may not be the same. Despite this difference, they perform comparatively and carry out the same conclusions (Section V-B). Therefore, the Pearson Correlation based methods are only used for further analysis. To better understand the effect of initial weight for item similarity learning methods, Table III summarizes the MAE comparison of Pearson Correlation based SimCF, RanCF and ConCF. It shows that they perform about the same, and have the same trend (The MAE value goes down and then up). It concludes that the performance of the proposed methods depend less on the initial weight than the similarity

methods which decide the k nearest neighbours (Although the dependence is also not heavy).

D. Comparison with Other Approaches

To effectively evaluate the performance of the proposed methods, we compare them against several state-of-the-art CF methods, namely SlopeOne and two variants of PureSVD. The Cosine similarity measurement method is used to calculate the similarities between items and generate k nearest neighbours for each item. The RanCF is chosen for the comparison with the parameters $\gamma = 0.003$, $\lambda = 0.04$ and $maxIter = 30$. On the dataset of MovieLens1M, k is set 400 so that Cosine based CF can achieve the optimum performance, while k is 800 for Cosine based RanCF. The performance comparison is shown in Fig. 5.

Fig. 5(a) shows that Cosine based RanCF performs comparatively with the state-of-the-art methods, SVD+ and SVD++, in terms of error metrics. Moreover, RanCF greatly outperforms SlopeOne, ICF and UICF with more accurate prediction. In Fig. 5(b), it can conclude that the precision and recall have inverse relationship. The precision values of RanCF and SVD+, SVD++, are about the same, but RanCF can achieve higher recall than them. Therefore, RanCF outperforms SVD+ and SVD++ in terms of F-measure. Since more than 80% of ratings are not less than 3 of both datasets and the overall mean is bigger than 3.5, ICF will get few FN by using the weighted sum as the prediction even though the prediction is not accurate (High MAE and RMSE in Fig. 5(a)). This is more obvious for UICF, which directly uses the average as the prediction. Therefore, ICF and UICF always give prediction bigger than 3, which results in few FN and many FP. Consequently, the precision is low, but the recall is relatively high. However, the recall is meaningful only when precision is high enough, thus, RanCF achieves outstanding performance compared against that achieved by state-of-the-art methods.

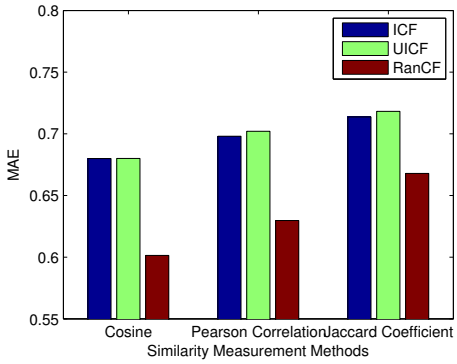
E. Computational Complexity Analysis

Matrix Factorization methods need to learn at least two factored matrices $\boldsymbol{P}^{m\times f}$ for users and $\boldsymbol{Q}^{n\times f}$ for items, where f is the number of latent factors. Therefore, the minimum number of parameters is $(m+n)\times f$. However, $n \times k$ is the maximum number of parameters required for the proposed methods, where k is the number of nearest neighbours for each item, and it is in the same order compared with f. In real-world systems, the number of users is far greater than the number of items, namely $m \gg n$. Therefore, the proposed methods require less parameters than required by Matrix Factorization methods. Besides, for each (u, i) pair in the training set, the computational complexity is $O(l)$ for the proposed methods, where l is the number of items which are rated by user u and are simultaneously the members of item i's k nearest neighbours, while it is $O(f)$ for Matrix Factorization methods since $\boldsymbol{P}^{m\times f}$ and $\boldsymbol{Q}^{n\times f}$ are dense. Generally, l is far less than k. Hence, $l \ll f$. This results in a much lower computational complexity for the proposed methods than Matrix Factorization methods.

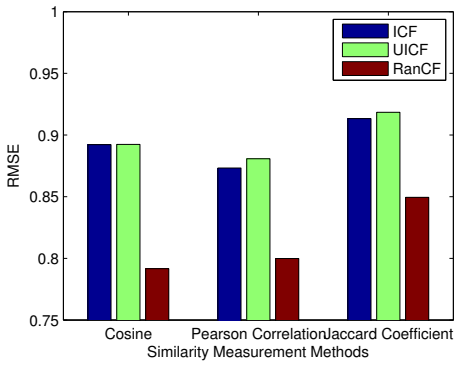
VI. Conclusion

Due to the inaccurate similarities calculated by current mainstream similarity measurement methods, which may bring second-order error in prediction, we proposed a series of item similarity learning methods to overcome this challenge. Experimental results show that the proposed approaches achieve comparable and even better performance against that achieved by the state-of-the-art methods, matrix factorization, and greatly outperform item based CF which has been widely deployed in industry. Besides, since the proposed methods inherit the interpretability from item based CF, their recommended results are more accessible than that provided by matrix factorization based methods.

These advantages drive us to deploy them in the real-world systems, in the future. Moreover, the Asynchronous Distributed Stochastic Gradient Descent technology will be adopted to learn item similarities so that the proposed methods are more applicable.



(a) MAE



(b) RMSE

Fig. 4: The optimum MAE and RMSE comparison of RanCF, ICF and UICF with different similarity methods on MovieLens1M dataset.

References

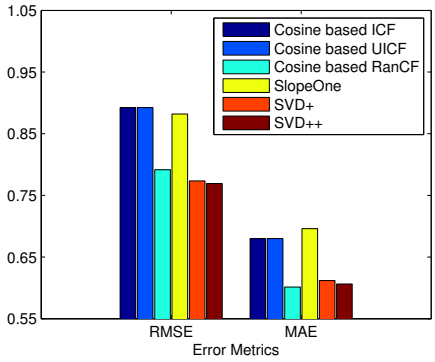
[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734–749, 2005.

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

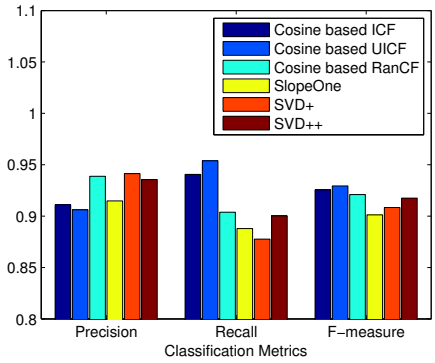
[3] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Foundations and Trends in Human-Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

TABLE III: The MAE comparison of three proposed algorithms with Pearson correlation generating k neaerest neighbors

Dataset	k	400	500	600	700	800	900	1000	1100	1200	1300	1400	1500
MoiveLens100k	SimCF	0.7985	0.7797	0.7731	0.7724	**0.7719**	0.7723	0.7733	0.7726	0.7728	0.7750	0.7766	0.7771
	RanCF	0.7943	0.7777	0.7725	0.7687	0.7706	0.7714	0.7718	0.7712	**0.7704**	0.7704	0.7740	0.7755
	ConCF	0.7970	0.7776	0.7723	**0.7702**	0.7713	0.7718	0.7720	0.7717	0.7717	0.7722	0.7735	0.7752
MoiveLens1M	SimCF	0.6737	0.6528	0.6399	0.6313	0.6286	**0.6274**	0.6278	0.6298	0.6316	0.6339	0.6365	0.6392
	RanCF	0.6723	0.6517	0.6398	0.6325	0.6303	**0.6297**	0.6306	0.6329	0.6352	0.6378	0.6409	0.6438
	ConCF	0.6649	0.6439	0.6319	0.6242	0.6222	**0.6219**	0.6229	0.6255	0.6279	0.6307	0.6338	0.6368



(a) MAE & RMSE



(b) Precision, Recall & F-measure

Fig. 5: The performance of RanCF compared against that achieved by state-of-the-art CF methods.

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

[5] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender systems handbook*. Springer, 2011, pp. 73–105.

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

[7] X. Su and T. M. Khoshgoftaar, "Collaborative filtering for multi-class data using belief nets algorithms," in *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*. IEEE, 2006, pp. 497–504.

[8] L. H. Ungar and D. P. Foster, "Clustering methods for collaborative filtering," in *AAAI Workshop on Recommendation Systems*, vol. 1, 1998.

[9] G. Shani, R. I. Brafman, and D. Heckerman, "An mdp-based recommender system," in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 453–460.

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

[11] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 1994, pp. 175–186.

[12] 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.

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

[14] N. N. Liu, X. Meng, C. Liu, and Q. Yang, "Wisdom of the better few: cold start recommendation via representative based rating elicitation," in *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 2011, pp. 37–44.

[15] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 1, p. 2, 2011.

[16] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[17] F. Xie, M. Xu, and Z. Chen, "Rbra: A simple and efficient rating-based recommender algorithm to cope with sparsity in recommender systems," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, 2012, pp. 306–311.

[18] F. Xie, Z. Chen, J. Shang, and G. C. Fox, "Grey forecast model for accurate recommendation in presence of data sparsity and correlation," *Knowledge-Based Systems*, 2014.

[19] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[20] Z. Huang, X. Li, and H. Chen, "Link prediction approach to collaborative filtering," in *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*. ACM, 2005, pp. 141–142.

[21] F. Xie, Z. Chen, H. Xu, X. Feng, and Q. Hou, "Tst: Threshold based similarity transitivity method in collaborative filtering with cloud computing," *Tsinghua Science and Technology*, vol. 18, no. 3, pp. 318–327, 2013.

[22] S. Kabbur, X. Ning, and G. Karypis, "Fism: factored item similarity models for top-n recommender systems," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 659–667.

[23] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 39–46.

[24] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 497–506.

[25] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *Society for Industrial Mathematics*, vol. 5, pp. 471–475, 2005.