Collaborative Filtering with Collective Training

Yong Ge[1], Hui Xiong[1], Alexander Tuzhilin[2], Qi Liu[3]

[1] Rutgers Business School, Rutgers University
hxiong@rutgers.edu,yongge@pegasus.rutgers.edu
[2] Leonard N. Stern School of Business, NYU, atuzhili@stern.nyu.edu
[3] University of Science and Technology of China, feiniaol@mail.ustc.edu.cn

ABSTRACT

Rating sparsity is a critical issue for collaborative filtering. For example, the well-known Netflix Movie rating data contain ratings of only about 1% user-item pairs. One way to address this rating sparsity problem is to develop more effective methods for training rating prediction models. To this end, in this paper, we introduce a *collective training* paradigm to automatically and effectively augment the training ratings. Essentially, the *collective training* paradigm builds multiple different Collaborative Filtering (CF) models separately, and augments the training ratings of each CF model by using the partial predictions of other CF models for unknown ratings. Along this line, we develop two algorithms, Bi-CF and Tri-CF, based on *collective training*. For Bi-CF and Tri-CF, we collectively and iteratively train two and three different CF models via iteratively augmenting training ratings for individual CF model. We also design different criteria to guide the selection of augmented training ratings for Bi-CF and Tri-CF. Finally, the experimental results show that Bi-CF and Tri-CF algorithms can significantly outperform baseline methods, such as neighborhood-based and SVD-based models.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms, Experimentation

Keywords

Collaborative Filtering, Collective Training

1. INTRODUCTION

Recommender systems [1] provide personalized suggestions by identifying user interests from user behavior data. As a major recommendation technique, collaborative filtering (CF) aims at predicting the preference of a user by us-

Table 1: A Sample Data Set.

	$User_1$	$User_2$	$User_3$	$User_4$	$User_5$
$Item_1$	NaN	NaN	2	3	NaN
$Item_2$	1	2	NaN	2	3
$Item_3$	2	4	2	4	5
$Item_4$	1	2	NaN	2	3
$Item_5$	1	2	1	NaN	4
$Item_6$	1	2	NaN	5	7
$Item_7$	NaN	NaN	5	NaN	NaN

Note: NaN indicates unknown rating.

ing available ratings or taste information from many users. Specifically, given N users, M items and a $M \times N$ preference matrix R, CF is typically to predict the unknown ratings in R by using the available training ratings. Many CF algorithms, which can usually be categorized into two groups: memory-based and model-based methods [1], have been proposed to address this prediction problem.

The prediction performance of most CF methods strongly depends on the available training ratings. In other words, better prediction can usually be expected if more training ratings become available. However, rating data are usually very sparse because it is expensive to obtain more training ratings from users or experts. Consequently, the unknown ratings usually significantly outnumber the available ratings. Then, the question is whether it is possible to leverage the abundant unknown ratings in addition to training ratings to improve the performance of CF methods. Through the sample data in Table 1, we demonstrate the feasibility to exploit unknown ratings to improve CF methods.

In Table 1, we have an item-user matrix (R), where there are 7 items and 5 users. For example, with an item-oriented KNN method (iKNN) [1], we can predict the rating $R(4, 3)$ and $R(2, 3)$ as around 1. Note that KNN is the acronym of K-Nearest Neighbors and is also known as neighborhood method [1]. With these two predictions, we can better measure the similarity between $User_3$ and $User_5$, thus we can better predict rating $R(1, 5)$ via using user-oriented KNN method (uKNN) [1]. In contrast, if we only use known ratings to predict $R(1, 5)$ with user-oriented KNN method [1], we are not able to obtain reliable similarity value among $User_3$ and $User_5$ because the support (i.e., the number of common ratings by $User_3$ and $User_5$) is too low. Therefore we are not able to predict $R(1, 5)$ well. Through this illustrative study, we show that the performance of one CF model can be improved by leveraging partial predictions of other CF models for unknown ratings.

To that end, in this paper, we introduce the *collective*

training paradigm to improve CF methods. Essentially, the *collective training* paradigm iteratively augments the training ratings of one model by using the partial predictions of other CF models, then re-trains all CF models again. Along this line, we first develop a Bi-CF algorithm based on *collective training* among two CF models, which iteratively augments the training ratings for one model by leveraging the partial predictions of the other model, and re-trains the two CF models and re-makes predictions. The final prediction is based on the ensemble of the two CF models. Furthermore, to exploit the advantage of different CF models, we collectively train three CF models and develop a Tri-CF algorithm. For both Bi-CF and Tri-CF algorithm, one essential challenge is how to select augmented training ratings. In this paper, we design two different criteria to guide the selection for Bi-CF and Tri-CF. Finally, the experimental results on MovieLens data show that both Bi-CF and Tri-CF models could result in better performance than several traditional methods, such as KNN and SVD methods.

2. RELATED WORK

First, the idea of *collective training* has been studied for classification and regression problems [5, 10, 12] in machine learning community. But, in this paper, we adapt *collective training* to collaborative filtering and propose two algorithms to deal with the arisen challenge, which is how to iteratively augment the training set for individual CF method. Second, in the field of collaborative filtering [4, 8], there are some research papers [7, 3, 6], which have already explored unknown ratings to improve collaborating filtering methods and are known as active collaborative filtering. However, most of these methods need to query users with a small amount of unknown ratings. Then these supplemental training samples are included and used to build the CF models again together with original training samples. In other words, user's or expert's interaction is still needed to exploit the unknown ratings. In addition, Zhang and Pu [11] introduced a specific recursive method to iteratively use some predicted ratings for predictions of other unknown ratings. However, this method is specifically designed for the user-based CF approach. In contrast, our *collective training* is developed to automatically exploit unknown ratings without user's interaction and collectively train and boost different CF approaches.

3. COLLECTIVE TRAINING

In the context of collaborative filtering, *collective training* is to boost one CF model by the predictions of other CF models. The diversity of these CF models is needed for *collective training* because the estimations of unknown ratings by these CF methods will be the same if all these CF models are identical. Then, the mutual boost effect among these CF models will disappear. With different CF methods, different algorithms can be developed to perform *collective training*. The methods to select augmented training ratings may also vary among different algorithms.

Though *collective training* can be generally adapted to various combinations of multiple CF methods, we focus on three CF methods, i.e., uKNN [2], iKNN [2] and SVD [9], and develop two specific examples of *collective training* among them. Specifically, we design Bi-CF and Tri-CF algorithms based on these three CF models. Before introducing Bi-CF and Tri-CF, we briefly review these three CF methods.

Suppose we have N users and M items, and a set of available ratings. To estimate the unknown rating r_{ji} to item j by user i, item-oriented KNN method makes the prediction for r_{ji} as: $r_{ji} = \frac{\sum_{v \in N(j)} s_{vj} r_{vi}}{\sum_{v \in N(j)} s_{vj}}$. $N(j)$ is a set of neighboring items that are also rated by user i. s_{vj} is similarity between item j and item v, which is often computed with traditional correlation measurement, e.g. Pearson Correlation or Cosine Correlation. The analogous user-oriented KNN make the prediction for r_{ji} as: $r_{ji} = \frac{\sum_{u \in N(i)} s_{ui} r_{ju}}{\sum_{u \in N(i)} s_{ui}}$, where $N(i)$ is a set of neighboring users who also rate item j. s_{ui} is similarity between user u and user i. SVD models a user's preference to a item as dot product of the user latent factor and the item latent factor [9]. Given observed training rates, user and item latent factors are learned by minimizing the objective function as:

$$E = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ji}(r_{ji} - U_i^T V_j)^2 + \alpha_U \sum_{i=1}^{N} ||U_i|| + \alpha_V \sum_{j=1}^{M} ||V_j||,$$

where I_{ji} is 1 if r_{ji} is observed, and 0 otherwise. α_U and α_V are parameters.

3.1 The Bi-CF Algorithm

Bi-CF algorithm is based on two different CF methods: SVD and user-oriented KNN (uKNN). Specifically, we provide the pseudo code of Bi-CF as shown in Figure 1. As we can see, we first make prediction for the unknown ratings by using uKNN and SVD methods (step 2-3). Secondly we select partial predictions yielded by individual model (step 6-7). Then we re-predict the unknown ratings by uKNN model (SVD-based model) with the selected predictions by SVD-based model(neighborhood-based model) in addition to the original training ratings (step 8-10). This process is iteratively performed until a stop criterion is satisfied, which is that both SVD and uKNN models do not change much. Specifically, the changing of SVD model is reflected by U and M in latent feature space. And the changing of uKNN can be reflected by the predictions for unknown ratings.

Furthermore, instead of selecting the augmented ratings from all unknown ratings, we select these augmented ratings from a pool of unknown ratings for each iteration (step 4 and 11). This strategy can significantly decrease the probability that the same set of predicted ratings is selected at different steps of iteration. And this mechanism could save much time for the selection, which is actually quite time-consuming due to the large number of unknown ratings. Also the selected augmented predictions (H_1 and H_2) are put back into W after each iteration. After all iterations, we make predictions for unknown ratings by combining the results of uKNN and SVD models (step 15). Also note that, Θ in Algorithm 1 represents all parameters for uKNN and SVD, including the number of neighbors for uKNN, the number of latent factor and penalty parameters for SVD.

Confidence Measurement. One critical challenge of Bi-CF algorithm is how to effectively and efficiently select the partial predictions from all unknown ratings. On one hand, if many inaccurate predictions are augmented, the CF model may be degraded, but not boosted. On the other hand, the overall iteration will be very time-consuming if the selection takes much time. To this end, we propose one criterion to efficiently estimate the confidence of prediction.

Since there is no benchmark for an unknown rating, we turn to consulting available ratings, which are neighboring to the unknown rating in terms of users or items, to estimate the confidence of prediction for the unknown rat-

```
ALGORITHM 1 Bi-CF(R, W, T, Θ, K)
Input:
        R: the set of known ratings
        W: the set of unknown ratings
        T: the set of testing ratings
        Θ: the set of parameters
        K: the number of augmented ratings
Output:
        P_t: the predictions on testing set.

1.    R_1 ← R; R_2 ← R
2.    Make prediction for W with uKNN and R_1
3.    Make prediction for W with SVD and R_2
4.    Generate pool W̃ by randomly selecting from W
5.    Repeat
6.       Select K predictions (denoted as H_1) from all
            predictions for W̃ by uKNN
7.       Select K predictions (denoted as H_2) from all
            predictions for W̃ by SVD
8.       R_1 ← R_1 ∪ H_2; R_2 ← R_2 ∪ H_1
9.       Make prediction for W with uKNN and R_1
10.      Make prediction for W with SVD and R_2
11.      Generate W̃ by randomly selecting from W
12.   End of Repeat until a stopping criterion is satisfied
13.   Get the predictions P_t^1 for test set T with uKNN
14.   Get the predictions P_t^2 for test set T with SVD
15.   Output: P_t ← Average(P_t^1, P_t^2);
```

Figure 1: The Bi-CF Algorithm

ing. For the prediction of one unknown rating, if these neighboring available ratings are predicted well with the CF model, we think the prediction of this unknown rating is high-confident. Thus, the average/overall deviation between ground truth ratings and predictions of these neighboring available ratings should be evaluated first. Specifically, given one unknown rating r_{ji}, we first find a set of items ($\tilde{N}(j)$), which are neighboring to item j and rated by user i. Also we find a set of users ($\tilde{N}(i)$), who are neighboring to user i and rate item j. With $\tilde{N}(j)$ and user i, we have a set of known ratings $\{r_{vi}\}, v \in \tilde{N}(j)$. And we have a set of known ratings $\{r_{ju}\}, u \in \tilde{N}(i)$, with item j and $\tilde{N}(i)$. We still can get the prediction for each one in these two sets of known ratings with CF models. Accordingly we denote these two sets of predictions as $\{p_{vi}\}$ and $\{p_{ju}\}$. Note that elements in $\{p_{vi}\}$ ($\{p_{ju}\}$) have one-to-one correspondence with elements in $\{r_{vi}\}$ ($\{r_{ju}\}$). Then we use RMSE (Root Mean Squared Error) to evaluate the average deviation between ground truth ratings and predictions as follows:

$$\sqrt{\frac{\sum_{u \in \tilde{N}(i)}(r_{ju} - p_{ju})^2 + \sum_{v \in \tilde{N}(j)}(r_{vi} - p_{vi})^2}{|\tilde{N}(i)| + |\tilde{N}(j)|}} \quad (1)$$

$|\tilde{N}(i)|$ or $|\tilde{N}(j)|$ is the number of neighboring users or item. We specify this parameter as the same as the number of neighbors in KNN methods. Since the confidence of prediction for r_{ij} is inversely proportional to RMSE, we select top-K predictions, which are associated with lowest RMSE in equation 1, for each iteration.

3.2 The Tri-CF Algorithm

In this subsection, we introduce Tri-CF algorithm, which is based on item-oriented KNN (iKNN), uKNN and SVD and boosts one CF model with the augmented ratings generated from the predictions of the other two CF models. As

shown in Figure 2, Tri-CF has a similar interactive process as Bi-CF. But different from Bi-CF model, Tri-CF evaluate the confidence of predictions for each unknown rating via analyzing the consistency of predictions by two models. In other words, if two CF motheds make consistent predicitons for one unknown rating, the predictions are considered as high-confident and will be added to the training set for the third CF model. Specifically, for one unknown rating, we can obtain three predictions p_1, p_2 and p_3 by uKNN, SVD and iKNN respectively. The consistency and confidence of two predictions p_1 and p_2 are inversely proportional to $|p_1 - p_2|$. Thus, among all unknown ratings, we select top-K unknown ratings, which are associated with lowest values of $p_1 - p_2$. And for each selected unknown rating, we calculate the average of p_1 and p_2, and add it into the training set for iKNN. For uKNN and SVD we use the same way to obtain the augmented training ratings.

However, the consistent predictions by two models (e.g., uKNN and SVD) still may be inaccurate, and consequently will degrade the third model (e.g., iKNN) if such predictions are augmented into training set of the third model (e.g., iKNN). Thus, inspired by [12], we heuristically put a constraint condition in order to select effective predictions and overcome the argmented noisy ratings. Specifically, we first evaluate the confidence of the predictions for each known rating with the the same method as in the above paragraph. Note that we consider the two predictions p_1 and p_2 as confident if $|p_1 - p_2|$ is lower than 0.5 in the experiment. Then, we count the number (denoted as c) of confident predictions for the known ratings. Among these c confident predictions, we count the number (denoted as c') of predictions, which are almost the same as the ground truth ratings. Finally we estimate the noise rate of the found high-confident estimations as $\frac{c-c'}{c}$. This noise rate is estimated for the augmented rating of individual CF model. Therefore, during each iteration, we estimate the noise rate of potential augmented ratings for each CF model. And we augment the training rating, if it is lower than certain threshold Nr. In addition, we use some other procedures for Tri-CF as mentioned in section 3.1, such as stopping criterion.

4. EXPERIMENTAL RESULTS

In this section, we empirically validate the performances of the proposed Bi-CF and Tri-CF models.

The Experiment Setup. We validate the proposed Bi-CF and Tri-CF models on the MovieLens dataset [1] which contains 100000 discrete ratings (on a 1-5 scale) from 943 users for 1682 movies. In this paper, 80% of known ratings are used as training ratings and 20% are used as the testing set. The parameters for SVD are specified as $\alpha_U = 0.05$, $\alpha_V = 0.05$, and the learning rate $\gamma = 0.003$ as suggested in [9]. And we represent the number of neighbors for KNN and equation 1 as Nei, the number of latent factors as f. In our experiments, we show the performance with different values of Nei and f. Also the number of augmented ratings during each iteration is set as $K = 500$. The noise rate threshold is set as $Nr = 0.1$. Finally, we use the RMSE [2, 1] metric to evaluate different methods.

In Table 2, we show the performances of different methods with different values of Nei and f. Particularly, we also directly ensemble SVD and uKNN by averaging the final

---

[1]http://www.grouplens.org/node/73

```
ALGORITHM 2  Tri-CF(R, W, T, Θ, K, Nr)
Input:
        R: the set of known ratings
        W: the set of unknown ratings
        T: the set of testing ratings
        Θ: the set of parameters
        K: the number of top-K confident predictions
        Nr: the noise rate threshold
Output:
        P_t: the predictions on testing set.

1.      R_1 ← R; R_2 ← R; R_3 ← R
2.      Make prediction for W with uKNN and R_1
3.      Make prediction for W with SVD and R_2
4.      Make prediction for W with iKNN and R_3
5.      Generate pool W̃ by randomly selecting from W
6.      Repeat
7.          If Noise rate for iKNN is lower than Nr.
            Then Select Top-K confident predictions H_12 from
                    W̃ by uKNN and SVD; R_3 ← R_3 ∪ H_12
8.          If Noise rate for uKNN is lower than Nr.
            Then Select Top-K confident predictions H_23 from
                    W̃ by SVD and iKNN; R_1 ← R_1 ∪ H_23
9.          If Noise rate for SVD is lower than Nr.
            Then Select Top-K confident predictions H_13 from
                    W̃ by uKNN and iKNN; R_2 ← R_2 ∪ H_13
10.         Make prediction for W with uKNN and R_1
11.         Make prediction for W with SVD and R_2
12.         Make prediction for W with iKNN and R_3
13.         Regenerate W̃ by randomly selecting from W
14.     End of Repeat until a stopping criterion is satisfied
15.     Get the predictions P_t^1 for test set T with CF_1
16.     Get the predictions P_t^2 for test set T with CF_2
17.     Get the predictions P_t^3 for test set T with CF_3
18.     Output: P_t ← Average(P_t^1, P_t^2, P_t^3);
```

Figure 2: The Tri-CF Algorithm

predictions of these two models. We represent this method as Ensemble. As can be seen, Bi-CF and Tri-CF models can outperform the competing methods, including KNN, SVD and Ensemble, in the most cases. The results of Bi-CF and Tri-CF are obtained after a stop criterion is satisfied.

To further study and compare the proposed two models, we compare the RMSEs on the testing set at different steps of iteration in Figure 3, where we obtain the RMSEs at each step of iteration by averaging the predictions of the two/three basic CF models. In Figure 3, we specify the number of neighbors as $Nei = 40$ and $f = 40$. As can be seen, the RMSEs of both Bi-CF and Tri-CF decrease significantly after several initial iterations. Also, the Tri-CF results show a little better performance during these iteration. Note that it takes much more steps to converge for both Bi-CF and Tri-CF models, but here we only show the first 10 iterations.

Table 2: RMSE Comparisons on MovieLens

Nei	f	uKNN	SVD	Ensemble	Bi-CF	Tri-CF
10	10	1.0401	0.987	0.9702	0.9581	0.9522
20	20	1.0207	1.0022	0.9702	0.9590	0.9535
30	30	1.0183	1.0162	0.9750	0.9600	0.9535
40	40	1.0181	1.0298	0.9795	0.9609	0.9580

5. CONCLUDING REMARKS

In this paper, we exploited the well-known concept of *collective training* for collaborative filtering and demonstrated its effectiveness for recommendation. Essentially, the *collec-*
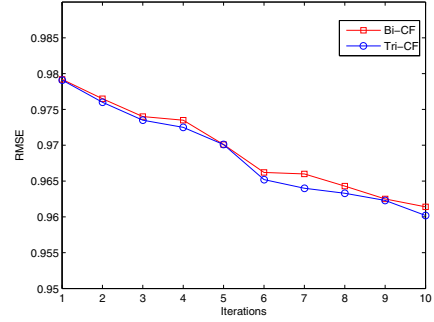
Figure 3: RMSEs at Different Iterations.

tive training paradigm builds multiple collaborative filtering models, and augments the training rating for one collaborative filtering model by leveraging the predictions of other collaborative filtering models. To demonstrate the usefulness and practicality of this powerful idea, we developed two specific examples of *collective training* of multiple CFs, i.e., Bi-CF and Tri-CF. Two different criteria are also designed to guide the selection of augmented training ratings. Finally, experimental results on the MovieLen data showed the advantages of both Bi-CF and Tri-CF by comparing with some baseline methods, such as KNN and SVD. As a future work, we would like to explore other possible combinations of *collective training*, in addition to Bi-CF and Tri-CF, and identify the most powerful combination methods. In addition, one limitation of Bi-CF and Tri-CF is that it takes many iterations before a stop criterion is satisfied. In the future, we will study the convergence of the iterations.

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Towards the next generation of recommender systems: A survey of the state-of-the art and possible extensions. *IEEE TKDE*, 17(6):734–749, 2005.

[2] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *IEEE ICDM*, pages 43–52, Omaha NE, US, 2007.

[3] C. Boutilier, R. S. Zemel, and B. Marlin. Active collaborative filtering. In *ACM SIGIR*, 2003.

[4] Y. Ge, Q. Liu, H. Xiong, A. Tuzhilin, and J. Chen. Cost-aware travel tour recommendation. In *SIGKDD*, San Diego, CA, 2011.

[5] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *ACM CIKM*, 2005.

[6] A. S. Harpale and Y. Yang. Personalized active learning for collaborative filtering. In *ACM SIGIR*, 2008.

[7] R. Jin and L. Si. A bayesian approach toward active learning for collaborative filtering. In *UAI*, 2004.

[8] Q. Liu, E. Chen, H. Xiong, and C. H. Q. Ding. Exploiting user interests for collaborative filtering: interests expansion via personalized ranking. In *ACM CIKM*, pages 1697–1700, Toronto, Canada, 2010.

[9] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD Cup and Workshop*, 2007.

[10] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-rad. Collective classification in network data articles. In *AI Magazine*, 2008.

[11] J. Zhang and P. Pu. A recursive prediction algorithm for collaborative filtering recommender systems. In *ACM RecSys*, 2007.

[12] Z.-H. Zhou and M. Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE TKDE*, 17(11):1529–1541, 2005.