One-Class Recommendation System with PU learning

Yu-Chi Chen
Dept. of Computer Science and Information Engineering
National Cheng Kung University
Taiwan, R.O.C.
P76024295@mail.ncku.edu.tw

Hung-Yu Kao
Dept. of Computer Science and Information Engineering
National Cheng Kung University
Taiwan, R.O.C.
hykao@mail.ncku.edu.tw

ABSTRACT
With the explosion of e-commerce, recommendation systems are getting well-known. Many kinds of recommendation systems are used in various websites. The one-class problem in recommendation systems is also a kind of recommendation problems that we cannot ignore. In the traditional one-class classification problem, we usually come up with Positive and Unlabeled learning (PU learning) but we cannot apply PU learning directly because we only can get the user-item rating matrix, in other words, the information is not enough to be applied to PU learning. The rating matrix does not define features of each ratings. Without getting the outlier information of the users and the items, we have to define the features from the one-class rating matrix. In this paper, we proposed a PU learning framework which is focus on the one-class recommendation problem, called RPU (*Recommendation by PU learning*). We use MovieLens and TencentWeibo datasets to evaluate our methods. RPU can get the best accuracy in the baselines of one-class approaches. In RPU, we have a feature selection part. We also contribute a few kinds of features that are appropriate for RPU. The features not only perform great in our RPU framework but perform around 10% better by using OCSVM.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information filtering;

General Terms
Algorithms, Experimentation

Keywords
Recommendation System; PU learning; feature selection; Matrix Factorization

1. INTRODUCTION
As the rapidly increasing number of users of online-shopping, recommendation systems become much more popular. Recommendation systems can advise the customers some products they might purchase but never heard of. It is a great technique to increase revenue on commercial basis. The aim of recommendation systems [2, 9, 14, 18] is to recommend the customers some products by preferences of the customers. According to Collaborative Filtering (CF), users will be recommended items by someone else who has similar tastes and preferences which would react in the ratings. We can recommend items to the users according to the relationships between the historical rating-behavior of users and can extract the users' preferences and the items' features by CF. Although it is hard to know what the features really are, we can still use it to predict what users would like.

However, there is a common problem in CF recommendation systems. Traditional CF recommendation systems recommend the users some items by the historical 5 ranks ratings. The zero ratings in the rating matrix mean the unknown rating which we need to predict. Because we cannot force the users to give the ratings of the items, we need to face the data-sparse problem. If the system doesn't have enough ratings, recommending might be difficult for the system. From the view of customers, if they can use only LIKE to evaluate the items, recommending would be simpler. It might raise the users' interest and make the users intent to give the ratings to the items. Moreover, the click behavior on the websites can be a sign that points out the websites might be liked by the users. The websites recommendation can be regarded as a one-class problem, too. The relationship between followers and followees in a social network is also the problem that we can regard as the same circumstance. Having two classes in the answers of the problem, to recommend the item or not, we can regard the one-class problem as a classification problem. Nonetheless, it is hard to define the features of the rating matrix which only have users, items and ratings in the matrix.

Positive and Unlabeled learning (PU learning) is aim to deal with the one-class classification problems, that is, the data need to have real features because PU learning includes a classifier. However, it is not easy to define features from a user-item rating matrix which we usually take as training data. As the assumption of CF, it assumes that there have some relationships between users or items. In memory-based Collaborative Filtering, it assumes the rating behaviors of users are similar to each other of the same preference, and so do the items. In model-based collaborative filtering, we can extract the latent factor matrices as features from the user-item rating matrix. Although we cannot identify what the features really are, the features can still be tried to be embedded into the PU learning framework. Both of memory-based and model-based Collaborative Filtering methods inspire the ideas for us to identify the features from the user-item rating matrix.

Being inspired by the memory-based and model-based traditional Collaborative Filtering and PU learning, we propose *Recommendation by PU learning (RPU)*. Figure 1 is the framework of RPU. In RPU framework, we contribute three parts: First of all, we calculate the features from the training data. We propose a feature by using the model-based CF, called f*f. We also calculate the similarity between the users and items in the rating matrix, which is the memory-based CF, to produce the features, called similarity feature. The last feature mixes the two kinds of Collaborative Filtering up. We call the hybrid feature hybrid-MFsim.

The other point that we focus on in RPU is the strategy of negative selecting. Due to the defect of the distribution of found

negative data, we cannot only choose the negative by intuitive thinking. We will show the proof of this phenomenon and provide a strategy to solve the problem later.

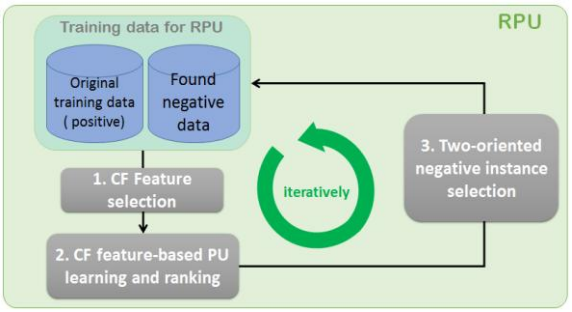

Figure 1. The framework of RPU

2. RELATED WORK

In this section, we review the previous works that related to our approach, including Collaborative Filtering, Positive and Unlabeled learning, One-class SVM [3, 4, 15], One-class collaborative filtering [1, 8, 10, 19], and their weakness on issues proposed in this paper.

2.1 Collaborative Filtering

Collaborative Filtering (CF), one of categories in a recommendation system, is known as a useable concept. There are two categories of CF, memory-based and model-based. In memory-based CF, we usually recommend item by calculating the similarity of users or items. The mathematical models are used to recommend in the model-based system.

The memory-based has two orientations, user-oriented and item-oriented. Both of two orientations calculate the similarity between users and items. If we want to know one of the unknown ratings, we can just predict it by voting or averaging the rating from other similar users or items.

Matrix factorization (MF) [7, 11] is another kind of CF and it is also a well-known method base on the mathematical model. MF factorizes a rating matrix into two latent factor matrices. Traditionally, the ratings range from one to five. After factorizing the rating matrix into two latent factor matrices, a user-feature matrix and a feature-item matrix, we can produce the predicted rating matrix by multiplying these two matrices. In algorithm of MF, we get two latent factor matrices P and Q randomly at first. Then, it takes plenty of iterations to update the latent factor matrices P' and Q' by the formulas Eq.(1-3). After lots of iterations, the difference e between the training rating matrix and the predicted rating matrix, which is the product of user-feature matrix and feature-item matrix, will be very small.

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

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

$$q'_{kj} = q_{kj} + \alpha(2e_{ij} p_{ik} - \beta q_{kj}) \quad (3)$$

2.2 Positive and Unlabeled Learning

Positive and Unlabeled learning (PU learning) [12] is introduced in Web Data Mining. It is a powerful strategy to solve the problem of one-class classification. One-class problem is hard to classify by general classifier because it has only one-class data in training set. PU learning can find out the reliable negative data from the unknown because some of the data in the unknown part might be very unlike positive training data. Then we assume the very-unlike-positive data in the unknowns are negative. There are many kinds of ways to implement PU learning. We can choose negative by the spy technique which is to label the instances as positive or negative by the positive probability [13]. However, there has another way. We can give a weight to express how much the probability that the instances to be negative [6]. The first method can be regarded as the special case of the second method, where the weight is only 0 and 1. We choose the negative instances by the spy technique because we think the spy technique is more pure to use which means we do not have to consider the degree of the instances. After lots of iterations, we can finally find out the considerable negative data from the unknowns. In the end, the traditional classifier can be used.

PU learning is largely applied in two-class classification. However, rating matrix does not have features that are needed in classifier. In this paper, we will propose some kinds of features from rating matrix.

2.3 One-Class SVM

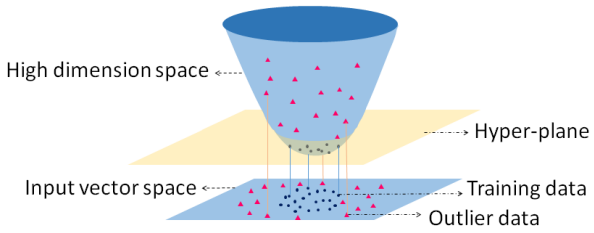

Figure 2. The way one-class SVM work.

In one-class SVM [21], it is hard to find a hyper-plane to separate data from training data and unlabeled data in input vector space because it is not a linear problem. When we want to classify data of the training class and outlier data with a hyper-plane, we need to project the data into feature space which is the higher dimension space. In the high dimension space, to find the hyper-plane to classify the data is becoming possible. Figure 2 shows how data project to the high dimension space and how the hyper-plane separates the training class and the other.

One-class SVM classify the data of training class from unlabeled testing data with a hyper-plane through mapping data into high dimension space.

2.4 One-Class Collaborative Filtering

2.4.1 All Missing Value as Unknown (AMAU)

Encountering the one-class problem that we only have rating matrix for training, we always mentioned All Missing Value as Unknown and All Missing Value as Negative strategies because these are the first two methods flashed into our mind. All Missing Value as Unknown is a method that we ignore the unknown part and just train the positive training data in MF. With the growing number of iterations of MF, the value of ratings is converging in the end. Eventually, we get the ratings almost the same and the predicted rating must be high. In this situation, it is hard to recommend.

2.4.2 All Missing Value as Negative (AMAN)

All Missing Value as Negative treats the missing values as negative data. With the product of two latent factor matrices gradually getting closer to the rating matrix, MF reaches the goal that extracts latent features from rating matrix iteratively.

However, some of the negative data that we use in MF are not negative. It is obviously not a very good assumption because there must have positive data in missing part of matrix.

2.4.3 sALS

sALS is proposed by Rong Pan et al. in ICDM 2008 [16]. It mentions that AMAN and AMAU are two extreme approaches to address one-class problem. They contribute a sampling approach, which is sampling some negative from unlabeled data by using the three concepts. It is an eclectic way to solve the one-class problem. Three strategies are there to be proposed.

sALS is going to sampling some negative data in the beginning. They give the probability for each missing value before three kinds of sampling strategies. After they sample some sets of negative, the sets of negative data will be used to be the training data of general matrix factorization respectively. After all, the prediction of missing value will be calculated by these few MF results.

RPU is going to compare with one-class SVM and one-class Collaborative Filtering in the experiment.

2.4.4 BPR

Bayesian Personalized Rankin (BPR) [17] creates user specific pair-wise preferences $i >_u j$ between a pair of items. i and j are the items and $>_u$ means the user like the item i more than item j. The concept is to find the preferences by the items whether had been seen or not. They identify the latent preferences of users by the historical buying behavior. This paper proposes the BPR-opt, which is the general optimization criterion for personalized ranking and proposes the learnBPR for learning models with respect to BPR-opt. BPR can be applied to Matrix Factorization and k-NN.

3. METHOD

RPU is an approach that focuses on dealing with the one-class recommendation problem. We have three parts in our RPU framework. Firstly, we calculate different kinds of features from the rating matrix by two sorts of Collaborative Filtering's concepts. Secondly, PU learning is used to classify and rank the unlabeled data by the probability. Thirdly, we choose some reliable found negative data as the negative training data in the next iteration by some strategies. With the gradually growing number of iterations, the performance will get better due to the reliable found negative data. In this section, we will describe the detail of each parts of RPU according to the order.

3.1 Feature Selection

*3.1.1 f*f feature*

The most well-known method of model-based collaborative filtering is Matrix Factorization (MF). Because MF can extract the latent factors from user-item rating matrix, we can regard the latent factors as features of user-item ratings. The two latent factor matrices are user-feature matrix and feature-item matrix. In classifier, we need a series of features of each user-item rating. Therefore, we can multiply two matrices to get the value of the features of each user-item rating. As Figure 3 illustrates how we calculate the features. We call this kind of feature f*f. To multiply User u's Feature k by Item i's Feature k, we can get the array of the user-item rating's features. The feature array of each user-item rating is $\text{feature}(U_u, I_i) = [U_u F_1 \times F_1 I_i, U_u F_2 \times F_2 I_i, ..., U_u F_k \times F_k I_i]$. U_u and I_i are the user and item we want to find the features.

3.1.2 Similarity feature

We usually use the user similarity or the item similarity to construct the memory-based CF in recommendation systems. According to the relationship between the users or the items, we can think up a similarity feature to define features from the memory-based CF. There is an example illustrates how we calculate the features value of the rating as Figure 4. We combine the users' similarity and the items' similarity. The formula is Cosine similarity Eq. (4) that we used in this calculation.

$$similarity = \cos(U_A, U_B) = \frac{U_A \cdot U_B}{\|U_A\| \|U_B\|} = \frac{\sum_{i=1} U_{Ai} \times U_{Bi}}{\sqrt{\sum_{i=1}^{n} (U_{Ai})^2} \sqrt{\sum_{i=1}^{n} (U_{Bi})^2}} \quad (4)$$

3.1.3 Hybrid feature (hybrid-MFsim)

We mix up two kinds of concepts of CF and generate the other sort of features, called Hybrid-MFsim. Hybrid-MFsim needs to use MF at first before we calculate the similarity between the users and the items of the features from latent factor matrices respectively. According to the principle of CF, there must have some relationships between the users or the items. In MF, the features of latent factor matrices can be described as the user's preference and the characteristics of items. Hence, *Hybrid-MFsim* implies that we calculate the similarity between user's preference and item's characteristics. Figure 5 illustrates how to calculate the value of features.

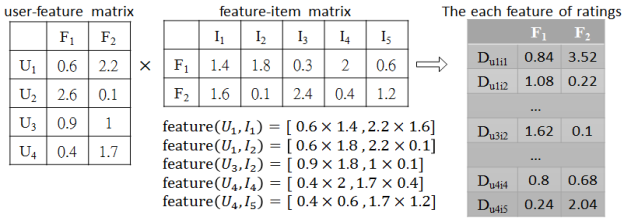

Figure 3. The example of *MF* feature

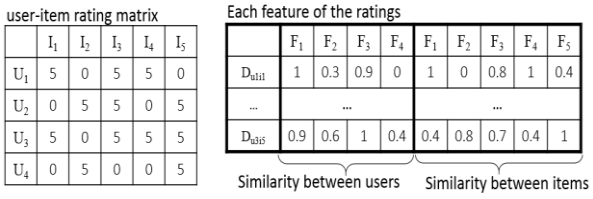

Figure 4. The example of similarity feature

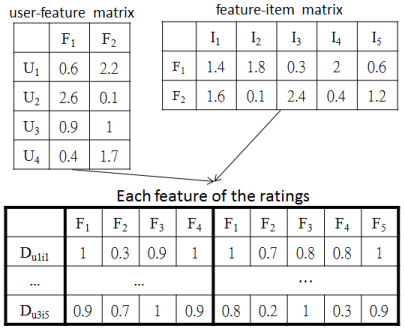

Figure 5. The example of Hybrid-MFsim

3.2 Kernel Framework of RPU

Figure 6 illustrates how RPU works. We need to define features which we described the details of it in the previous section before we get into the PU learning part. We will rank the probability of unlabeled data and choose the smallest to be the found-negative data. If there has the users or the items which have no found-negative data, it might impact the performance of following iterations. Therefore, each user and item needs to have one found-negative data at least. In the next iteration, if the number of found-negative data is smaller than the number of the training positive data, we will random some data as negative until the number of the negative data is same as the number of the training positive data. If the number of the found-negative data is larger than the number of the training positive data, then we use the found-negative data and the training positive data to the classifier.

In the feature selection part, we use the best two features in RPU respectively, similarity and hybrid-MFsim features. For similarity feature, we calculate the value at the beginning of the process. For hybrid-MFsim feature, we update the features' value at every five iterations because we believe the found negative data are the true negative and are going to promote the performance by updating two latent factor matrices. After updating MF latent factor matrices, RPU can get the better features.

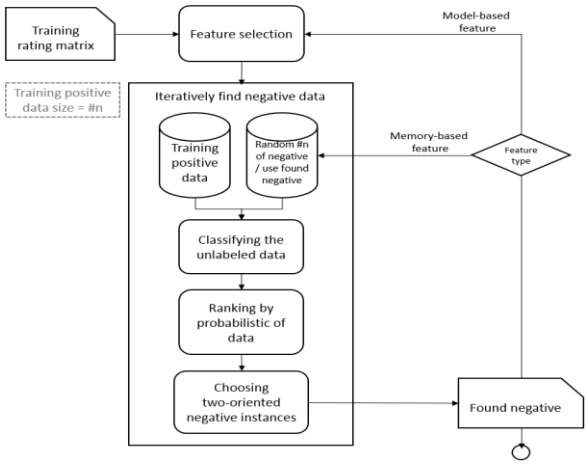

Figure 6. The framework of RPU.

3.2.1 Concentration of Found Negative Data

The smallest probability of data is going to be the negative data in the next iteration. In hybrid-MFsim feature selection of RPU, this finding negative strategy has a problem. The found negative would assemble in the same columns or the same rows because of the characteristics of MF and similarity. We call it "concentration problem". An intuitive method to solve this problem is that we do not choose the smallest probability of data as the found negative data. We set the small threshold of data and random found negative data from the data which the probability is under the threshold. In addition, ensuring to find at least one negative data for each user and item is also important. We use this concept to avoid the concentration problem as well. For instance, we random some found-negative data from the first third unlabeled set by each user and item. Because we don't insist on the smallest probability of data, we can find the data that the performance is not bad but avoiding the data concentration. The Figure 7 shows this phenomenon. The red grids are the found negative of the iteration. In Figure 8, it is the strategy that we set the threshold as 33.3% and random some negative before the first third of the unlabeled set. We can clearly find out the difference between these two strategies. Setting threshold can alleviate this defect.

After all, when the input data coming, which has the only positive data in it, the input data will be calculated the values of features first. We learn the whole ratings' probability in matrix from the input data and the found negative data, and afterwards we find some negative data from those unknown ratings. The new set of negative and the input positive data are used to be the new training data for next iteration. Eventually, we can find a batch of negative with the increasing number of iterations. It is our procedure, which is the PU learning framework that especially aims at rating matrix, to recommend user some item by only positive data.

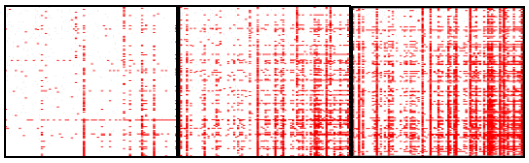

Figure 7(a) Figure 7(b) Figure 7(c)

Figure 7. The choosing smallest strategy of found negative distribution of (a) the first, (b) the fifth and (c) the tenth iteration.

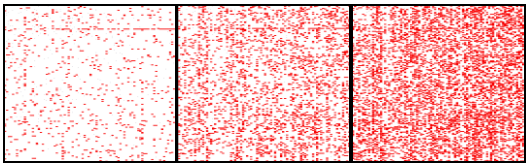

Figure 8(a) Figure 8(b) Figure 8(c)

Figure 8. The random before one third strategy of found negative distribution of (a) the first, (b) the fifth and (c) the tenth iteration.

4. EXPERIMENTS

4.1 Dataset

In order to know whether RPU can work on general recommendation problem, we decided to use the 5 ranks rating data in our experiments. In this paper, we use two dataset, MovieLens and TencentWeibo.

4.1.1 MovieLens

We use MovieLens dataset for our experiments which is available on GroupLens website. According to [20], in these kind of one-class problem, we can use the general 5 ranks dataset but we need to reconstruct the rating matrix. Because the values range from one to five, we should normalize the ratings to one and five which means negative and positive data. After we reconstruct the rating, we have positive data and negative data. We extract 1500 positive data as training data. The rest of the data are testing dataset. Because we want to keep the testing data having half positive and half negative, we divide the testing dataset into three balance subsets. The dataset statistic is as bellow, Table 1. We totally have three sets of training data and three sets of testing data of each training dataset. Training proportion means the proportion of training data in the whole rating matrix. The rest of values are the quantity of data.

4.1.2 *TencentWeibo*

TencentWeibo dataset can be downloaded from KDDCUP 2012 track 1[1]. The dataset has users and items. The item in this dataset can be a person, an organization, or a group. The rating is positive and negative, so we don't have to reconstruct this dataset. Positive data means the user follows the item and negative is not. The dataset indicate the users are going to receive the recommendation of people, organizations, and groups or not. We have three dataset and each dataset has three different testing set because we expect the testing data's distribution of positive and negative is balance. The dataset statistic is as bellow. It is noted that TencentWeibo datasets is sparser than MovieLens datasets.

Table 1. The statistic of MovieLens dataset

Dataset	User size	Item size	Training positive	Training proportion	Testing positive	Testing negative
1	266	180	1500	0.030	619	619
2	275	189	1500	0.029	471	471
3	261	194	1500	0.030	486	486

Table 2. The statistic of TencentWeibo dataset

Dataset	User size	Item size	Training positive	Training proportion	Testing positive	Testing negative
1	145	175	450	0.018	374	374
2	140	165	450	0.019	335	335
3	142	164	450	0.019	330	330

4.2 Baseline

To compare the feature selections in the one-class SVM [3], we use *r-item* feature to be the baseline. The features of the classifier are defined by [16]. The feature also based on memory-based CF. It considers the user's rating on the remaining set of items. We call this feature *r-item* feature in the rest of paper. We compare the three features that we proposed in this paper with r-item feature by using one-class SVM.

We also compare with other methods of one-class collaborative filtering. sALS is a one-class collaborative filtering method that proposed by Rong Pan et al. We use the best one of the three strategies in their paper, which is user-oriented. After sampling step, the sampling data is going to be the training data in matrix factorization. We sample five sets of negative and execute MF three times before we average the results. AMAN and AMAU are two extreme kinds of matrix factorization application. If we take the values which are smaller than three as negative, it is not a fair way for AMAN and AMAU. These two kinds of methods get the continuous value. Therefore, we rank the items by each user's ratings which are calculating by AMAN or AMAU. We regard the first half values as positive and the second half values as negative. Although BPR [17] can be another way of one-class collaborative filtering, BPR is focus on optimizing MF and k-NN, not focus on sampling. However, what we propose in this paper is more like sampling the negative instances, so we don't compare the performance here.

4.3 Experimental results

4.3.1 *Feature Selection in OCSVM*

In Section 3.1, we proposed three kinds of feature selections, that is the model-based CF method (f*f), the memory-based CF method (similarity) and hybrid method (hybrid-MFsim). To make sure that our features can be used to one-class SVM, we compare the accuracy of these three different kinds of feature selections in one-class SVM. According to the previous paper [5], the accuracy is a better metric for top-N recommendation. One-class recommendations are also going to find some items recommending to the users. One-class recommendations are a top-N-like recommendation without ordering the items. Therefore, in this paper, we use the accuracy to evaluate the performance of our methods and the baselines.

Figure 9 shows the performance comparison of our proposed feature selection methods and the baseline feature selection method, r-item, in both MovieLens and TencentWeibo by using OCSVM. We can find the similarity feature is the best in MovieLens dataset and the hybrid-MFsim is the best in TencentWeibo dataset. The r-item and f*f feature are the worst. Both similarity feature and hybrid-MFsim feature has the concept that users and items should have some relationships between them. That is to say, the relationships between users and items are significant concepts. The performance can be improved a lot by using these two kinds of features. Both two categories of CF methods are meaningful idea for recommendation by using OCSVM.

In MF, two latent factor matrices would be produce. Both of f*f feature and hybrid-MFsim feature calculate the features from the latent factor matrices. f*f feature use the latent features of latent factor matrices directly. The features might be wrong because the proportion of training data in rating matrix is small in the beginning. However, hybrid-MFsim would calculate the similarity between the latent features. That is, although hybrid-MFsim is also use the latent feature of latent factor matrices, the wrong feature in the matrix will be dominated by other user's feature and item's feature.

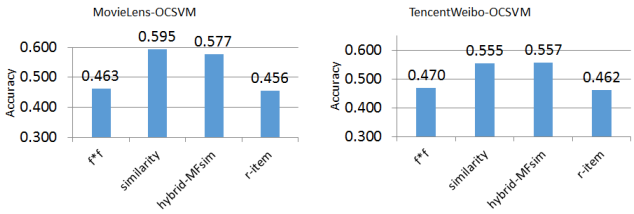

Figure 9. The accuracy of each feature in OCSVM (a) MovieLens (b) TencentWeibo

4.3.2 *Comparison with Baseline*

As shown in Figure 10, RPU has a higher accuracy in MovieLens datasets. AMAN and AMAU's results are not good but better than OCSVM with r-item feature. These are obviously not a very good solution in this problem. sALS is better than OCSVM with r-item feature as well but worse than RPU in MovieLens dataset. The accuracy of OCSVM with similarity feature and hybrid-MF feature are 10 percent and 8 percent more than OCSVM with r-item feature. Both similarity and hybrid-MFsim feature in RPU manifest great performance. RPU with similarity feature and hybrid-MFsim feature gain around 8 percent more accuracy than OCSVM with similarity feature and 14 percent more accuracy than OCSVM with the r-item feature. That is, in OCSVM, the features that we propose can improve the accuracy. Even more, RPU can perform great than other methods.

In TencentWeibo dataset which shows the results in Figure 11, RPU with both features are better than other baseline. However, RPU with similarity feature is not as good as the other dataset. It is because TencentWeibo datasets are sparser than MovieLens.

[1] http://www.kddcup2012.org/c/kddcup2012-track1

It is hard to tell the difference between the users' and items' similarity because it only has seldom training ratings of each column and row in the rating matrix. It indicates hybrid-MFsim is a more stable way in one-class recommendation problem.

5. CONCLUSIONS

In this paper, we proposed RPU, which is a PU learning framework to deal with the one-class problem in a recommendation system, especially containing positive opinion only. In a traditional recommendation system, we recommend user the item by analyzing their buying behavior. In the case, we usually can get only user-item-rating matrix. However, we might meet one class problem at times such as using click behavior or some kind of particular rating to recommend. According to these restrictions, it is hard to recommend users some item by traditional methods. We proposed two kinds of useful features in our paper. The two features are helpful in OCSVM. Moreover, we contribute a PU learning framework, RPU, which aims at the recommendation system. Through the innovative features and the step of negative selection, RPU can function on traditional rating matrix of a recommendation system and the performance has great improvement by applying the proposed framework.

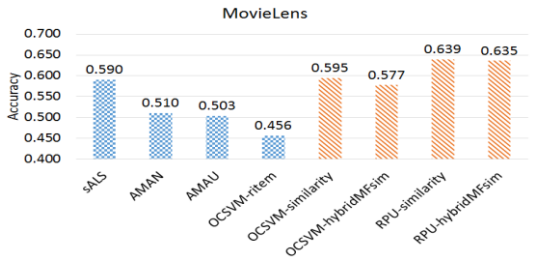

Figure 10. The accuracy of comparison between baseline and RPU.

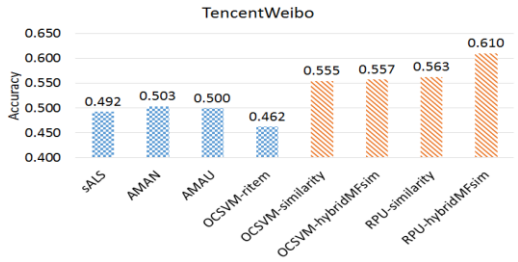

Figure 11. The accuracy of comparison between baseline and RPU.

REFERENCES

[1] Breese, J. S., Heckerman, D. and Kadie, C. *Empirical analysis of predictive algorithms for collaborative filtering*. Morgan Kaufmann Publishers Inc., City, 1998.

[2] Burke, R. *Hybrid web recommender systems*. Springer, City, 2007.

[3] Chang, C. C. and Lin, C. J. {LIBSVM}: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 3 2011), 27:21--27:27.

[4] Chen, Y., Zhou, X. S. and Huang, T. S. *One-class SVM for learning in image retrieval*. IEEE, City, 2001.

[5] Cremonesi, P., Koren, Y. and Turrin, R. *Performance of recommender algorithms on top-n recommendation tasks*. ACM, City, 2010.

[6] Elkan, C. and Noto, K. *Learning classifiers from only positive and unlabeled data*. ACM, City, 2008.

[7] Funk, S. *Netflix update: Try this at home*. City, 2006.

[8] Hill, W., Stead, L., Rosenstein, M. and Furnas, G. *Recommending and evaluating choices in a virtual community of use*. ACM Press/Addison-Wesley Publishing Co., City, 1995.

[9] Jannach, D., Zanker, M., Felfernig, A. and Friedrich, G. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[10] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R. and Riedl, J. GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, 40, 3 1997), 77-87.

[11] Koren, Y., Bell, R. and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 8 2009), 30-37.

[12] Liu, B. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

[13] Liu, B., Lee, W. S., Yu, P. S. and Li, X. *Partially supervised classification of text documents*. Citeseer, City, 2002.

[14] Mahmood, T. and Ricci, F. *Improving recommender systems with adaptive conversational strategies*. ACM, City, 2009.

[15] Manevitz, L. M. and Yousef, M. One-class SVMs for document classification. *the Journal of machine Learning research*, 22002), 139-154.

[16] Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M. and Yang, Q. *One-class collaborative filtering*. IEEE, City, 2008.

[17] Rendle, S., Freudenthaler, C., Gantner, Z. and Schmidt-Thieme, L. *BPR: Bayesian personalized ranking from implicit feedback*. AUAI Press, City, 2009.

[18] Resnick, P. and Varian, H. R. Recommender systems. *Communications of the ACM*, 40, 3 1997), 56-58.

[19] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. *Item-based collaborative filtering recommendation algorithms*. ACM, City, 2001.

[20] Sindhwani, V., Bucak, S., Hu, J. and Mojsilovic, A. *A family of non-negative matrix factorizations for one-class collaborative filtering problems*. City, 2009.

[21] Vlasveld, R. *Introduction to One-class Support Vector Machines*. City, 2013.