One-Class Collaborative Filtering

Rong Pan[1] Yunhong Zhou[2] Bin Cao[3] Nathan N. Liu[3] Rajan Lukose[1]
Martin Scholz[1] Qiang Yang[3]
[1.]HP Labs, 1501 Page Mill Rd, Palo Alto, CA, 94304, US
{rong.pan,rajan.lukose,scholz}@hp.com
[2] Rocket Fuel Inc. Redwood Shores, CA 94065 yzhou@rocketfuelinc.com
[3.]Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, Hong Kong
{caobin, nliu, qyang}@cse.ust.hk

Abstract

Many applications of collaborative filtering (CF), such as news item recommendation and bookmark recommendation, are most naturally thought of as one-class collaborative filtering *(OCCF) problems. In these problems, the training data usually consist simply of binary data reflecting a user's action or inaction, such as page visitation in the case of news item recommendation or webpage bookmarking in the bookmarking scenario. Usually this kind of data are extremely sparse (a small fraction are positive examples), therefore ambiguity arises in the interpretation of the non-positive examples. Negative examples and unlabeled positive examples are mixed together and we are typically unable to distinguish them. For example, we cannot really attribute a user not bookmarking a page to a lack of interest or lack of awareness of the page. Previous research addressing this one-class problem only considered it as a classification task. In this paper, we consider the one-class problem under the CF setting. We propose two frameworks to tackle OCCF. One is based on weighted low rank approximation; the other is based on negative example sampling. The experimental results show that our approaches significantly outperform the baselines.*

1 Introduction

Personalized services are becoming increasingly indispensable on the Web, ranging from providing search results to product recommendation. Examples of such systems include recommending products at Amazon.com[1], DVDs at Netflix[2], News by Google [3] etc. The central technique used in these systems is collaborative filtering (CF) which aims at predicting the preference of items for a particular user based on the items previously rated by all users. The rating expressed in different scores (such as a 1-5 scale in Netflix) can be explicitly given by users in many of these systems. However, in many more situations, it also can be implicitly expressed by users' behaviors such as click or not-click and bookmark or not-bookmark. These forms of implicit ratings are more common and easier to obtain.

Although the advantages are clear, a drawback of implicit rating, especially in situations of data sparsity, is that it is hard to identify representative negative examples. All of the negative examples and missing positive examples are mixed together and cannot be distinguished. We refer to collaborative filtering with only positive examples given as One-Class Collaborative Filtering (OCCF). OCCF occurs in different scenarios with two examples as follows.

- *Social Bookmarks*: Social bookmarks are very popular in Web 2.0 services such as del.icio.us. In such a system, each user bookmarks a set of webpages which can be regarded as positive examples of the user's interests. But two possible explanations can be made for the behavior that a user did not bookmark a webpage. The first one is, the page is of the users' interest but she did not see the page before; the second one is the user had seen this page but it is not of her interest. We cannot assume all the pages not in his bookmarks are negative examples. Similar examples include social annotation, etc.

[1]http://www.amazon.com
[2]http://www.netflix.com
[3]http://news.google.com

- *Clickthrough History*: Clickthrough data are widely used for personalized search and search result improvement. Usually a triple $< u, q, p >$ indicates a user u submitted a query q and clicked a page p. It is common that pages that have not been clicked on are not collected. Similar to the bookmark example, we cannot judge whether the page is not clicked because of the irrelevance of its content or redundancy, for example.

There are several intuitive strategies to attack this problem. One approach is to label negative examples to convert the data into a classical CF problem. But this is very expensive or even intractable because the users generating the preference data will not bear the burden. In fact, users rarely supply the ratings needed by traditional learning algorithms, specifically not negative examples [23]. Moreover, based on some user studies [14], if a customer is asked to provide many positive and negative examples before the system performs well, she would get a bad impression of it, and may decline to use the system. Another common solution is to treat all the missing data as negative examples. Empirically, this solution works well (see Section 4.6). The drawback is that it biases the recommendation results because some of the missing data might be positive. On the other hand, if we treat missing as unknown, that is, ignore all the missing examples and utilize the positive examples only and then feed it into CF algorithms that only model non-missing data (as in [24]), a trivial solution arising from this approach is that all the predictions on missing values are positive examples. All missing as negative (AMAN) and all missing as unknown (AMAU) are therefore two extreme strategies in OCCF.

In this paper, we consider how to balance the extent of treating missing values as negative examples. We propose two possible approaches to OCCF. These methods allow us to tune the tradeoff in the interpretation of so-called negative examples and actually result in better performing CF algorithms overall. The first approach is based on weighted low rank approximation [24]. The second is based on negative example sampling. They both utilize the information contained in unknown data and correct the bias of treating them as negative examples. While the weighting-based approach solves the problem deterministically, the sampling-based method approximates the exact solution with much lower computational costs for large scale sparse datasets.

Our contributions are summarized as follows. First we propose two possible frameworks for the one-class collaborative filtering problem and provide and characterize their implementations; second, we empirically study various weighting and sampling approaches using several real world datasets. Our proposed solutions significantly outperform the two extremes (AMAN and AMAU) in OCCF problems, with at least 8% improvement over the best baseline approaches in our experiments. In addition, we show empirically that these two proposed solution frameworks (weighting and sampling based) for OCCF have almost identical performance.

The rest of the paper is organized as follows. In the next section, we review previous works related to the OCCF problems. In Section 3, we propose two approaches for OCCF problems. In Section 4, we empirically compare our methods to some baselines on two real world data sets. Finally, we conclude the paper and give some future works.

2 Related Works

2.1 Collaborative Filtering

In the past, many researchers have explored collaborative filtering (CF) from different aspects ranging from improving the performance of algorithms to incorporating more resources from heterogeneous data sources [1]. However, previous research on collaborative filtering still assumes that we have positive (high rating) as well as negative (low rating) examples. In the non-binary case, items are rated using scoring schemes. Most previous work focuses on this problem setting. In all the CF problems, there are a lot of examples whose rating is missing. In [2] and [19], the authors discuss the issue of modeling the distribution of missing values in collaborative filtering problems. Both of them cannot handle the case where negative examples are absent.

In the binary case, each example is either positive or negative. Das et al. [8] studied news recommendation, while a click on a news story is a positive example, and a non-click indicates a negative example. The authors compare some practical methods on this large scale binary CF problem. KDD Cup 2007 hosted a "Who rated What" recommendation task while the training data are the same as the Netflix prize dataset (with rating). The winner team [15] proposed a hybrid method combining both SVD and popularity using binary training data.

2.2 One-class Classification

Algorithms for learning from positive-only data have been proposed for binary classification problems. Some research addresses problems where only examples of the positive class are available [22] (refer to one-class

classification) where others also utilize unlabeled examples [16]. For one-class SVMs [22], the model is describing the single class and is learned only from positive examples. This approach is similar to density estimation [4]. When unlabeled data are available, a strategy to solve one-class classification problems is to use EM-like algorithms to iteratively predict the negative examples and learn the classifier [28, 17, 26]. In [9], Denis show that function classes learnable under the statistical query model are also learnable from positive and unlabeled examples if each positive example is left unlabeled with a constant probability.

The difference between our research and previous studies on learning from one-class data is that they aim at learning one single concept with positive examples. In this paper, we are exploring collaboratively learning many concepts in a social network.

2.3 Class Imbalance Problem

Our work is also related to the class imbalance problem which typically occurs in classification tasks with more instances of some classes than others. The one-class problem can be regarded as one extreme case of a class imbalance problem. Two strategies are used for solving the class imbalance problem. One is at the data level. The idea is to use sampling to re-balance the data [3] [18]. Another one is at the algorithmic level where cost-sensitive learning is used [10] [27]. A comparison of the two strategies can be found in [20].

3 Weighting & Sampling based Approaches

As discussed above, AMAN and AMAU (no missing as negative) are two general strategies for collaborative filtering, which can be considered to be two extremes. We will argue that there can be some methods in between that can outperform the two strategies in OCCF problems; examples include "all missing as weak negative" or "some missing as negative". In this section, we introduce two different approaches to address the issue of one-class collaborative filtering. They both balance between the strategies of missing as negative and missing as unknown. The first method uses weighted low rank approximations [24]. The idea is to give different weights to the error terms of positive examples and negative examples in the objective function; the second one is to sample some missing values as negative examples based on some sampling strategies. We first formulate the problem and introduce the main notation in this paper. In the next two subsections, we discuss the two solutions in details.

3.1 Problem Definition

Suppose we have m users and n items and the previous viewing information stored in a matrix $\boldsymbol{R}$. The element of $\boldsymbol{R}$ takes value 1, which represents a positive example, or '?', which indicates an unknown (missing) positive or negative example. Our task is to identify potential positive examples from the missing data based on $\boldsymbol{R}$. We refer to it as *One-Class Collaborative Filtering* (OCCF). Note that, in this paper, we assume that we have no additional information about users and items besides $\boldsymbol{R}$. In this paper, we use bold capital letters to denote a matrix. Given a matrix $\boldsymbol{A}$, A_{ij} represents its element, $\boldsymbol{A}_{i\cdot}$ indicates the i-th row of $\boldsymbol{A}$, $\boldsymbol{A}_{\cdot j}$ symbolizes the j-th column of $\boldsymbol{A}$, and $\boldsymbol{A}^T$ stands for the transpose of $\boldsymbol{A}$.

3.2 wALS for OCCF

Our first approach to tackle the one-class collaborative filtering problem is based on a weighted low-rank approximation [11, 24] technique. In [24], weighted low-rank approximations (wLRA) is applied to a CF problem with a naive weighting scheme assigning "1" to observed examples and "0" to missing (unobserved) values, which corresponds to the AMAU. Another naive method for OCCF is to treat all missing values as negative examples. However, because there are positive examples in missing values, this treatment can make mistakes. We address this issue by using low weights on the error terms. Next, we propose the weighted alternating least squares (wALS) for OCCF. We further discuss various weighting schemes different from naive schemes AMAU ([24]) and AMAN.

Given a matrix $\boldsymbol{R} = (R_{ij})_{m\times n} \in \{0,1\}^{m\times n}$ with m users and n items and a corresponding non-negative weight matrix $\boldsymbol{W} = (W_{ij})_{m\times n} \in \Re_+^{m\times n}$, weighted low-rank approximation aims at approximating $\boldsymbol{R}$ with a low rank matrix $\boldsymbol{X} = (X_{ij})_{m\times n}$ minimizing the objective of a weighted Frobenius loss function as follows.

$$\mathcal{L}(\boldsymbol{X}) = \sum_{ij} W_{ij} (R_{ij} - X_{ij})^2 . \quad (1)$$

In the above objective function $\mathcal{L}(\boldsymbol{X})$ (Eq. (1)), $(R_{ij} - X_{ij})^2$ is the common square error term often seen in low-rank approximations, and W_{ij} reflects the contribution of minimizing the term to the overall objective $\mathcal{L}(\boldsymbol{X})$. In OCCF, we set $R_{ij} = 1$ for positive examples; for missing values, we posit that most of them are negative examples. We replace all the missing values with zeros. As we have high confidence on the observed positive examples where $R_{ij} = 1$, we set

the corresponding weights W_{ij} to 1. In contrast, different from the simple treatment of missing as negative, we lower the weights on "negative" examples. Generally we set $W_{ij} \in [0, 1]$ where $R_{ij} = 0$. Before discussing the weighting schemes for "negative" examples, we show how to solve the optimization problem $\text{argmin}_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X})$ effectively and efficiently.

Consider the decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ where $\boldsymbol{U} \in \Re^{m\times d}$ and $\boldsymbol{V} \in \Re^{n\times d}$. Note that usually the number of features $d \ll r$ where $r \approx \min(m, n)$ is the rank of the matrix $\boldsymbol{R}$. Then we can re-write the objective function (Eq. (1)) 3 as Eq. (2)

$$\mathcal{L}(\boldsymbol{U}, \boldsymbol{V}) = \sum_{ij} W_{ij} \left(R_{ij} - \boldsymbol{U}_{i.}\boldsymbol{V}_{j.}^T\right)^2. \quad (2)$$

To prevent overfitting, one can append a regularization term to the objective function $\mathcal{L}$ (Eq. (2)):

$$\begin{aligned}\mathcal{L}(\boldsymbol{U}, \boldsymbol{V}) &= \sum_{ij} W_{ij} \left(R_{ij} - \boldsymbol{U}_{i.}\boldsymbol{V}_{j.}^T\right)^2 \\ &\quad +\lambda\left(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2\right), \quad (3)\end{aligned}$$

or

$$\begin{aligned}\mathcal{L}(\boldsymbol{U}, \boldsymbol{V}) &= \sum_{ij} W_{ij} \left(\left(R_{ij} - \boldsymbol{U}_{i.}\boldsymbol{V}_{j.}^T\right)^2 \right. \\ &\quad \left. + \lambda\left(\|\boldsymbol{U}_{i.}\|_F^2 + \|\boldsymbol{V}_{j.}\|_F^2\right)\right). \quad (4)\end{aligned}$$

In Eq. (3) and Eq. (4), $\|.\|_F$ denotes the Frobenius norm and λ is the regularization parameter which, in practical problems, is determined with cross-validation. Note that Eq. (4) subsumes the special case of regularized low-rank approximation in [21, 30]. Zhou et al. [30] show that the alternating least squares (ALS) [11] approach is efficient for solving these low rank approximation problems. In this paper, we extend this approach to weighted ALS (wALS). Now we focus on minimizing the objective function $\mathcal{L}$ (Eq. (4)) to illustrate how wALS works.

Taking partial derivatives of $\mathcal{L}$ with respect to each entry of $\boldsymbol{U}$ and $\boldsymbol{V}$, we obtain

$$\begin{aligned}\frac{1}{2}\frac{\partial \mathcal{L}(\boldsymbol{U}, \boldsymbol{V})}{\partial U_{ik}} &= \textstyle\sum_j W_{ij}\left(\boldsymbol{U}_{i.}\boldsymbol{V}_{j.}^T - R_{ij}\right)V_{jk} \\ &\quad +\lambda\left(\textstyle\sum_j W_{ij}\right)U_{ik}, \forall 1 \le i \le m, 1 \le k \le d. \quad (5)\end{aligned}$$

Then we have

$$\begin{aligned}&\frac{1}{2}\frac{\partial \mathcal{L}(\boldsymbol{U}, \boldsymbol{V})}{\partial \boldsymbol{U}_{i.}} \\ =\ & \frac{1}{2}\left(\frac{\partial \mathcal{L}(\boldsymbol{U}, \boldsymbol{V})}{\partial U_{i1}}, \ldots, \frac{\partial \mathcal{L}(\boldsymbol{U}, \boldsymbol{V})}{\partial U_{id}}\right) \\ =\ & \boldsymbol{U}_{i.}\left(\boldsymbol{V}^T\widetilde{\boldsymbol{W}_{i.}}\boldsymbol{V} + \lambda\left(\textstyle\sum_j W_{ij}\right)\boldsymbol{I}\right) - \boldsymbol{R}_{i.}\widetilde{\boldsymbol{W}_{i.}}\boldsymbol{V},\end{aligned}$$

Algorithm 1 Weighted Alternating Least Squares (wALS)

Require: data matrix $\boldsymbol{R}$, weight matrix $\boldsymbol{W}$, rank d
Ensure: Matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ with ranks of d
  Initialize $\boldsymbol{V}$
  repeat
    Update $\boldsymbol{U}_{i.}, \forall i$ with Eq. (6)
    Update $\boldsymbol{V}_{j.}, \forall j$ with Eq. (7)
  until convergence.
  return $\boldsymbol{U}$ and $\boldsymbol{V}$

where $\widetilde{\boldsymbol{W}_{i.}} \in \Re^{n\times n}$ is a diagonal matrix with the elements of $\boldsymbol{W}_{i.}$ on the diagonal, and $\boldsymbol{I}$ is a $d \times d$ identity matrix.

Fixing $\boldsymbol{V}$ and solving $\frac{\partial \mathcal{L}(\boldsymbol{U}, \boldsymbol{V})}{\partial \boldsymbol{U}_{i.}} = 0$, we have

$$\boldsymbol{U}_{i.} = \boldsymbol{R}_{i.}\widetilde{\boldsymbol{W}_{i.}}\boldsymbol{V}\left(\boldsymbol{V}^T\widetilde{\boldsymbol{W}_{i.}}\boldsymbol{V} + \lambda\left(\textstyle\sum_j W_{ij}\right)\boldsymbol{I}\right)^{-1}, \quad \forall 1 \le i \le m. \quad (6)$$

Notice that the matrix $\boldsymbol{V}^T\widetilde{\boldsymbol{W}_{i.}}\boldsymbol{V} + \lambda\left(\sum_j W_{ij}\right)\boldsymbol{I}$ is strictly positive definite, thus invertible. It is not difficult to prove that without regularization, $\boldsymbol{V}^T\widetilde{\boldsymbol{W}_{i.}}\boldsymbol{V}$ can be a degenerate matrix which is not invertible.

Similarly, given a fixed $\boldsymbol{U}$, we can solve $\boldsymbol{V}$ as follows.

$$\boldsymbol{V}_{j.} = \boldsymbol{R}_{.j}^T\widetilde{\boldsymbol{W}_{.j}}\boldsymbol{U}\left(\boldsymbol{U}^T\widetilde{\boldsymbol{W}_{.j}}\boldsymbol{U} + \lambda\left(\textstyle\sum_i W_{ij}\right)\boldsymbol{I}\right)^{-1}, \quad \forall 1 \le j \le n, \quad (7)$$

where $\widetilde{\boldsymbol{W}_{.j}} \in \Re^{m\times m}$ is a diagonal matrix with the elements of $\boldsymbol{W}_{.j}$ on the diagonal.

Based on Eq. (6) and Eq. (7), we propose the following iterative algorithm for wLRA with regularization (based on Eq. (4)). We first initialize the matrix $\boldsymbol{V}$ with Gaussian random numbers with zero mean and small standard deviation (we use 0.01 in our experiments). Next, we update the matrix $\boldsymbol{U}$ as per Eq. (6) and then update the matrix $\boldsymbol{V}$ based on Eq. (7). We repeat these iterative update procedures until convergence. We summarize the above process in **Algorithm 1** which we refer to as Weighted Alternating Least Squares (wALS). Note that for the objective function Eq. (3) with a uniform regularization term, we only need to change both $\left(\sum_j W_{ij}\right)$ in Eq. (6) and $(\sum_i W_{ij})$ in Eq. (7) to 1.

3.2.1 Weighting Schemes: Uniform, User Oriented, and Item Oriented

As we discussed above, the matrix $\boldsymbol{W}$ is crucial to the performance of OCCF. $\boldsymbol{W} = \boldsymbol{1}$ is equivalent to the case of AMAN with the bias discussed above. The basic idea

Table 1. Weighting Schemes

	Pos Examples	"Neg" Examples
Uniform	$W_{ij} = 1$	$W_{ij} = \delta$
User-Oriented	$W_{ij} = 1$	$W_{ij} \propto \sum_j R_{ij}$
Item-Oriented	$W_{ij} = 1$	$W_{ij} \propto m - \sum_i R_{ij}$

of correcting the bias is to let W_{ij} involve the credibility of the training data ($\boldsymbol{R}$) that we use to build a collaborative filtering model.d For positive examples, they have relative high likeliness to be true. We let $W_{ij} = 1$ for each pair of (i, j) that $R_{ij} = 1$. For missing data, it is very likely that most of them are negative examples. For instance, in social bookmarking, a user has very few web pages and tags; for news recommendation, a user does not read most of the news. That is why previous studies make the AMAN assumption although it biases the recommendations. However, we notice that the confidence of missing values being negative is not as high as of non-missing values being positive. Therefore, essentially, we should give lower weights to the "negative" examples. The first weighting scheme assumes that a missing data being a negative example has an equal chance over all users or all items, that is, it uniformly assign a weight $\delta \in [0, 1]$ for "negative" examples. The second weighting scheme posits that if a user has more positive examples, it is more likely that she does not like the other items, that is, the missing data for this user is negative with higher probability. The third weighting scheme assumes that if an item has fewer positive examples, the missing data for this item is negative with higher probability. We summarize these three schemes in Table 1. A parameter for the three schemes is the ratio of the sum of the positive example weights to the sum of the negative example weights. We will discuss the impact of the parameter in Section 4.6. In the future, we plan to explore more weighting schemes and learn the weight matrix $\boldsymbol{W}$ iteratively.

3.3 Sampling-based ALS for OCCF

As we state above, for one-class CF, a naive strategy is to assume all missing values to be negative. This implicit assumption of most of the missing values being negative is roughly held in most cases. However, the main drawback here is that the computational costs are very high when the size of the rating matrix $\boldsymbol{R}$ is large. wALS has the same issue. We will analyze its computational complexity in the next subsection. Another critical issue with the naive strategy is the imbalanced-class problem discussed in Section 2.3.

In this subsection, we present a stochastic method

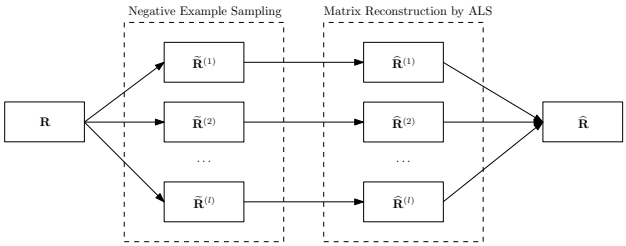


Figure 1. A diagram that illustrates an ensemble based on negative exampling sampling for OCCF

based on negative example sampling for OCCF as shown in Figure 1. In phase I, we sample negative examples from missing values. Based on an assumed probability distribution, we generate a new matrix $\widetilde{\boldsymbol{R}}^{(i)}$ including all positive examples in $\boldsymbol{R}$. In phase II, for each $\widetilde{\boldsymbol{R}}^{(i)}$, we re-construct the rating matrix $\widehat{\boldsymbol{R}}^{(i)}$ by a special version of wALS which we discuss in Section 3.2. Finally, we combine all the $\widehat{\boldsymbol{R}}^{(i)}$ with equal weights generating a matrix $\widehat{\boldsymbol{R}}$ which approximates $\boldsymbol{R}$. We refer to this method as sampling ALS Ensemble (sALS-ENS).

3.3.1 Sampling Scheme

Since there are too many negative examples (compared to positive ones), it is costly and not necessary to learn the model on all entries of $\boldsymbol{R}$. The idea of sampling can help us to solve the OCCF problem. We use a fast ($O(q)$) random sampling algorithm [25] to generate new training data $\widehat{\boldsymbol{R}}$ from the original training data $\boldsymbol{R}$ by negative example sampling given a sampling probability matrix $\widehat{\boldsymbol{P}}$ and negative sample size q. As OCCF is a class-imbalanced problem, where positive examples are very sparse, we transfer all positive examples to the new training set. We then sample negative examples from missing data based on $\widehat{\boldsymbol{P}}$ and the negative sample size q.

In this algorithm, $\widehat{\boldsymbol{P}}$ is an important input. In this paper, we provide three solutions which correspond to the following sampling schemes:

1. Uniform Random Sampling: $\widehat{P}_{ij} \propto 1$. All the missing data share the same probability of being sampled as negative examples.
2. User-Oriented Sampling: $\widehat{P}_{ij} \propto \sum_i I[R_{ij} = 1]$, that is, if a user has viewed more items, those items that she has not viewed could be negative with higher probability.
3. Item-Oriented Sampling: $\widehat{P}(i, j) \propto 1/\sum_j I[R_{ij} =$

1], which means that if an item is viewed by less users, those users that have not viewed the item will not view it either.

3.3.2 Bagging

After generating a new training matrix by the above algorithm, we can use a low-rank matrix $\widehat{\boldsymbol{R}}$ to approximate $\widetilde{\boldsymbol{R}}$ using wALS. Because $\widetilde{\boldsymbol{R}}$ is stochastic, $\widehat{\boldsymbol{R}}$ can also be biased and unstable. A practical solution to the problem is to construct an ensemble. In particular, we use the bagging technique [6] (**Algorithm 2**).

Algorithm 2 Bagging Algorithm for OCCF

Require: matrix $\boldsymbol{R} \in \Re^{m \times n}$, matrix $\widehat{\boldsymbol{P}} \in \mathbb{R}^{m \times n}$, sample size q, number of single predictor ℓ
Ensure: Reconstructed matrix $\widehat{R}$
  for $i = 1 : \ell$ **do**
    Generate a new training matrix $\widetilde{\boldsymbol{R}_i}$ by negative example sampling (Section 3.3.1)
    Reconstruct $\widehat{\boldsymbol{R}_i}$ from $\widetilde{\boldsymbol{R}_i}$ by ALS [30]
  end for
  $\widehat{\boldsymbol{R}} = \frac{1}{\ell} \sum_{i=1}^{\ell} \widehat{\boldsymbol{R}_i}$
  return $\widehat{\boldsymbol{R}}$

3.4 Computational Complexity Analysis

Next, we analyze the running time of wALS and sALS-ENS. Recall that U is a $m \times d$ matrix, V is a $n \times d$ matrix, and $d \leq \min\{m, n\}$. For wALS, each step of updating U (or M) takes time $O\left(d^2 nm\right)$ (based on Eqs. 6 and 7). The total running time of wALS is

$$O\left(d^2 n_t mn\right)$$

assuming that it takes n_t rounds to stop.

For sALS-ENS similarly, assuming that ALS takes on-average n_t rounds to stop, and the number of NES predictors is ℓ, then its total running time is

$$O\left(d^2 \ell n_t \left(n_r \left(1+\alpha\right) + \left(m+n\right) d\right)\right).$$

In practice, n_t, ℓ, α are small constants (n_t ranges from 20 to 30, $\ell \leq 20$, and $\alpha \leq 5$), and $\left(n+m\right) d \leq n_r \left(1+\alpha\right)$. Therefore the running time of wALS is $O(mn)$, while the running time of sALS-ENS is $O(n_r)$. Thus sALS-ENS is more scalable to large-scale sparse data compared to wALS. To be precise, the running time ratio of wALS to sALS-ENS is $\left(\frac{mn}{\ell(n_r(1+\alpha)+(n+m)d)}\right)$. When the data is very sparse $\left(\frac{n_r}{mn} \ll 1\right)$, then ALS-ENS takes less time to finish than wALS; otherwise, wALS is faster.

4 Experiments

4.1 Validation Datasets

We use two test datasets to compare our proposed algorithms with possible baselines. The first dataset, the Yahoo news dataset, is a news click through record stream[4]. Each record is a user-news pair which consists of the user id and the URL of the Yahoo news article. After preprocessing to make sure that the same news always gets the same article id, we have 3158 unique users and 1536 identical news stories. The second dataset is from a social bookmarking site. It is crawled from http://del.icio.us. The data contains 246,436 posts with 3000 users and 2000 tags.

4.2 Experiment setting and Evaluation Measurement

As a most frequently used methodology in machine learning and data mining, we use cross-validation to estimate the performance of different algorithms. The validation datasets are randomly divided into training and test sets with a 80/20 splitting ratio. The training set contains 80% known positive examples and the other elements of the matrix are treated as unknown. The test set includes the other 20% known positive and all unknown examples. Note that the known positives in the training set are excluded in the test process. The intuition of good performance of a method is that the method has high probabilities to rank known positives over unknown examples most of which are usually negative examples. We evaluate the performance on the test set using MAP and half-life utility which will be discussed below. We repeat the above procedure 20 times and report both mean and standard deviation of the experimental results. The parameters of our approaches and the baselines are determined by cross-validation.

MAP (Mean Average Precision) is widely used in information retrieval for evaluating the ranked documents over a set of queries. We use it in this paper to assess the overall performance based on precisions at different recall levels on a test set. It computes the mean of average precision (AP) over all users in the test set, where AP is the average of precisions computed at all positions with a preferred item:

$$AP_u = \frac{\sum_{i=1}^{N} prec\left(i\right) \times pref\left(i\right)}{\#\text{ of preferred items}}, \tag{8}$$

where i is the position in the rank list, N is the number of retrieved items, $prec\left(i\right)$ is the precision (fractions of

[4]We thank "NielsenOnline" for providing the clickstream data.

retrieved items that are preferred by the user) of a cut-off rank list from 1 to i, and $pref\,(i)$is a binary indicator returning 1 if the i-th item is preferred or 0 otherwise.

Half-life Utility (HLU): Breese et al. [5] introduced a half-life utility [13] ("cfaccuracy" [12]) to estimate of how likely a user will view/choose an item from a ranked list, which assumes that the user will view each consecutive item in the list with an exponential decay of possibility. A half-life utility over all users in a test set is defined as in Eq. (9).

$$R = 100\frac{\sum_u R_u}{\sum_u R_u^{max}}, \tag{9}$$

where R_u is the expected utility of the ranked list for user u and R_u^{max} is the maximally achievable utility if all true positive items are at the top of the ranked list. According to [5], R_u is defined as follows.

$$R_u = \sum_j \frac{\delta\,(j)}{2^{(j-1)(\beta-1)}}, \tag{10}$$

where $\delta\,(j)$ equals 1 if the item at position j is preferred by the user and 0 otherwise, and β is the half-life parameter which is set to 5 in this paper, which is the same as in [5].

4.3 Baselines

We evaluate our approaches weighting/sampling negative examples by comparing with two categories of baselines treating all missing as negative (AMAN) or treating all missing as unknown (AMAU).

4.3.1 AMAN

In AMAN settings, most traditional collaborative filtering algorithms can be directly applied. In this paper, we use several well-known collaborative filtering algorithms combined with the AMAN strategy as our baselines, which include the alternating least squares with the missing as negative assumption (ALS-AMAN), singular value decomposition (SVD) [29], and a neighborhood-based approach including user-user similarity[1] and item-item similarity algorithms[1].

4.3.2 AMAU

Following the AMAU strategy, it is difficult to adapt traditional collaborative filtering algorithms to obtain non-trivial solutions, as we discussed in Section 1. In this case, ranking items by their overall popularity is a simple but widely used recommendation method. Another possible approach is to convert the one-class collaborative filtering problem into a one-class classification problem. In this paper, we also include one such algorithm, namely the one-class SVM [22] into our pool of baseline methods. The idea is to create a one-class SVM classifier for every item, which takes a user's ratings on the remaining set of items as input features and predicts if the user's rating on the target item is positive or negative. The set of training instances for a SVM classifier consists of the rating profiles of those users who have rated the target item, which should consists of positive examples only in the one-class collaborative filtering setting, which could be used to train a one-class SVM classifier for each target item.

4.4 Impact of Number of Features

Figure 2 shows the impact of the number of features (parameter d) on SVD and wALS. We can see for SVD, the performance will first increase and then drop as we increase the number of features. But for wALS, the performance is much more stable and keeps increasing. The performance of wALS usually converge at around 50 features. In our following experiments, we will use the optimal feature number for SVD (10 for Yahoo news data and 16 for user-tag data) and wALS (50).

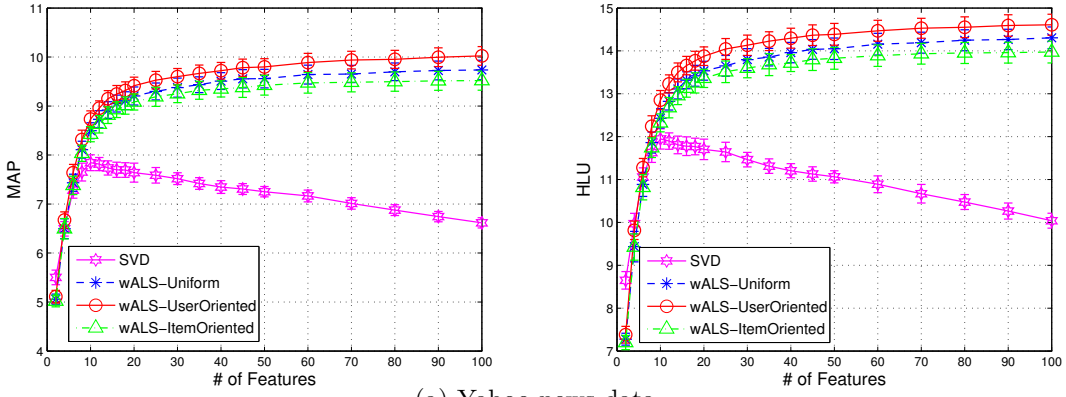
4.5 Sampling and Weighting Approaches

In Section 3, we introduced two approaches to OCCF based on sampling and weighting. For each approach, we proposed three types of schemes, that is uniform, user-oriented and item-oriented. In this section, we will compare the three schemes for both sampling and weighting.

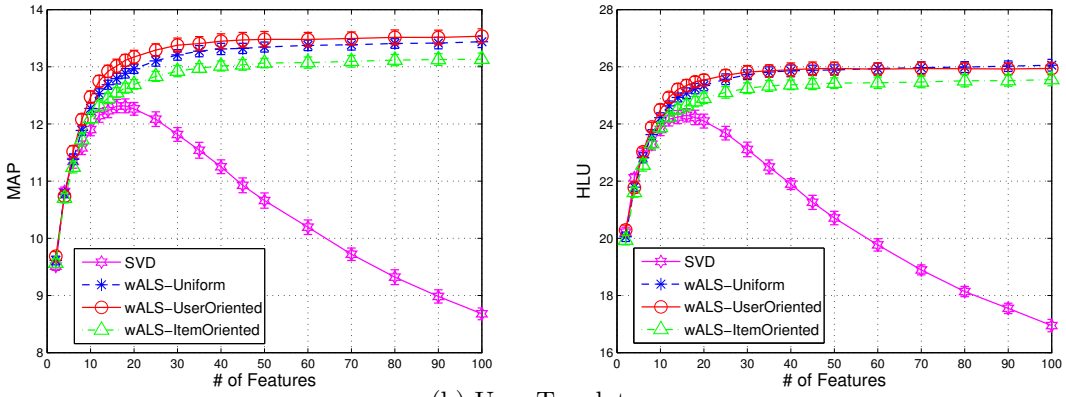
Table 2 compares these schemes. Among the weighting schemes, the user-oriented weighting scheme is the best and the item-oriented weighting scheme is the worst. The uniform weighting lies in between. This may be due to the imbalance between the number of users and the number of items. For the current datasets, the number of users is much larger than the number of items.

4.6 Comparison with Baselines

Figure 3 shows the performance comparisons of different methods based on the missing as unknown strategy (Popularity and SVM), methods based on the missing as negative strategy (SVD and ALS-AMAN) and our proposed methods(wALS, sALS). The x-axes α is defined as $\alpha = (\sum_{ij\in R_{ij}=0} W_{ij})/(\sum_{ij\in R_{ij}=1} W_{ij})$. For comparison consideration, given α, the negative sampling parameter q in **Algorithm** 2 (sALS-ENS) is set



(a) Yahoo news data



(b) User-Tag data

Figure 2. Impact of number of features

	Yahoo News				User-Tag			
	sALS-ENS		wALS		sALS-ENS		wALS	
	MAP	HLU	MAP	HLU	MAP	HLU	MAP	HLU
Uniform	9.55	13.98	9.56	14.05	13.33	25.81	13.35	25.89
UserOriented	**9.80**	**14.42**	**9.82**	**14.39**	**13.44**	**25.90**	**13.48**	**25.94**
ItemOriented	9.42	13.76	9.41	13.83	13.03	25.32	13.07	25.43

Table 2. Comparisons of different weighting and sampling schemes

to $\alpha \times n_r$, where n_r indicates the number of total positive examples. The baseline popularity will not change with the parameter α, therefore it is shown in a horizontal line in the figures. The same holds for the baselines SVD and ALS-AMAN. It can be seen from the figures that our methods outperform all the methods based on missing as negative and missing as unknown strategies.

Parameter α controls the proportion of negative examples. As $\alpha \to 0$, the methods are approaching the AMAU strategies and as $\alpha \to 1$, the methods are approaching the AMAU strategies. We can clearly see that the best results lie in between. That is to say both weighting and sampling methods outperform the baselines. The weighting approach slightly outperform the sampling approach. But as indicated in Table (3), the sampling approach is much more efficient when α is relative small.

	wALS	sALS	sALS-ENS
$\alpha = 1$	904.00	32.58	651.67
$\alpha = 2$	904.00	37.45	749.13

Table 3. Runtime (seconds) of wALS, sALS ans sALS-ENS with 20 sALS combinations on the Yahoo News data.

We can also see that compared to the AMAU strategies the missing as negative strategy is more effective. This is because although the label information of unlabeled examples are unknown, we still have the prior knowledge that most of them are negative examples. Disregarding such information does not lead to competitive recommendations. This is somewhat different with the conclusion in class imbalance classification problems where discarding examples of the dominating class often produces better results [7].

5 Conclusion and Future Work

Motivated by the fact that negative examples are often absent in many recommender systems, we formulate the problem of one-class collaborative filtering (OCCF). We show that some simple solutions such as the "all missing as negative" and "all missing as unknown" strategies both bias the predictions. We correct the bias in two different ways, negative example weighting and negative example sampling. Experimental results show that our methods outperform state of the art algorithms on real life data sets including social bookmarking data from del.icio.us and a Yahoo news dataset.

For future work, we plan to address the problem how to determine the parameter α. We also plan to test other weighting and sampling schemes and to identify the optimal scheme. We also plan to study the relationship between wALS and sALS-ENS theoretically.

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. *TKDE*, 17(6):734–749, 2005.

[2] Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *STOC*, pages 619–626. ACM, 2001.

[3] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6(1):20–29, 2004.

[4] S. Ben-David and M. Lindenbaum. Learning distributions by their density levels: a paradigm for learning without a teacher. *J. Comput. Syst. Sci.*, 55:171–182, 1997.

[5] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52. Morgan Kaufmann, 1998.

[6] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations*, 6:1–6, 2004.

[8] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280. ACM, 2007.

[9] F. Denis. PAC learning from positive statistical queries. In *ALT, LNCS 1501*, pages 112–126, 1998.

[10] C. Drummond and R. Holte. C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Proc. ICML'2003 Workshop on Learning from Imbalanced Data Sets II*, 2003.

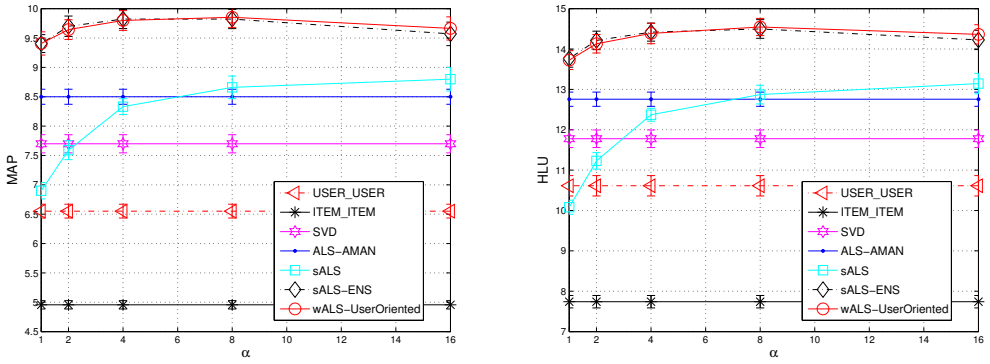
[11] K. R. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21(4):489–498, 1979.

[12] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. M. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR*, 1:49–75, 2000.

[13] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *TOIS*, 22(1):5–53, 2004.

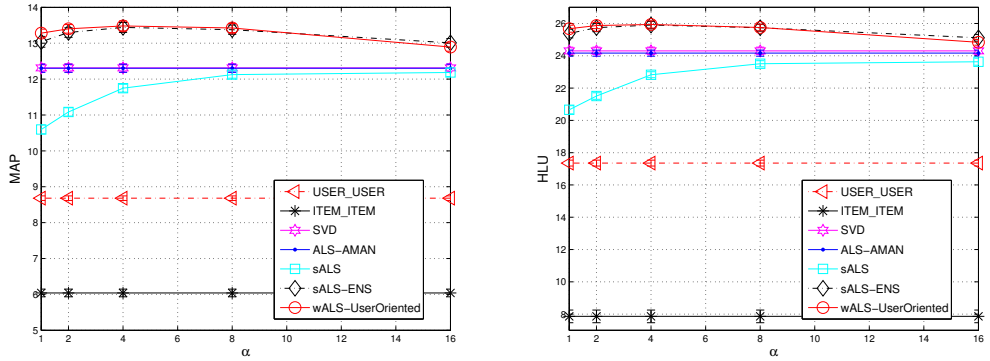
[14] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37:18–28, 2003.

[15] M. Kurucz, A. A. Benczur, T. Kiss, I. Nagy, A. Szabo, and B. Torma. Who rated what: a combination of SVD, correlation and frequent sequence mining. In *Proc. KDD Cup and Workshop*, 2007.



	POP	OCSVM		POP	OCSVM
MAP	3.4528	2.2545	HLU	5.3461	2.5958

(a) Yahoo news data



	POP	OCSVM		POP	OCSVM
MAP	7.315	3.2500	HLU	15.7703	3.6260

(b) User-Tag data

Figure 3. Impact of the ratio of negative examples to positive examples (α).

[16] W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, pages 448–455. AAAI Press, 2003.

[17] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. In *ICDM*, pages 179–188. IEEE Computer Society, 2003.

[18] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory undersampling for class-imbalance learning. In *ICDM*, pages 965–969. IEEE Computer Society, 2006.

[19] B. Marlin, R. Zemel, S. Roweis, and M. Slaney. Collaborative filtering and the missing at random assumption. In *UAI*, 2007.

[20] B. Raskutti and A. Kowalczyk. Extreme re-balancing for svms: a case study. *SIGKDD Explorations*, 6:60–69, 2004.

[21] R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798. ACM, 2007.

[22] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[23] I. Schwab, A. Kobsa, and I. Koychev. Learning user interests through positive examples using content analysis and collaborative filtering, 2001. Internal Memo.

[24] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *ICML*, pages 720–727. AAAI Press, 2003.

[25] J. S. Vitter. Faster methods for random sampling. *Commun. ACM*, 27(7):703–718, 1984.

[26] G. Ward, T. Hastie, S. Barry, J. Elith, and J. Leathwick. Presence-only data and the EM algorithm. *Biometrics*, 2008.

[27] G. M. Weiss. Mining with rarity: a unifying framework. *SIGKDD Explorations*, 6:7–19, 2004.

[28] H. Yu, J. Han, and K. C.-C. Chang. PEBL: positive example based learning for web page classification using SVM. In *KDD*, pages 239–248. ACM, 2002.

[29] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman. Using singular value decomposition approximation for collaborative filtering. In *CEC*, pages 257–264. IEEE Computer Society, 2005.

[30] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *AAIM, LNCS 5034*, pages 337–348, 2008.