Removing Redundancy and Inconsistency in Memory-Based Collaborative Filtering

Kai Yu
Siemens AG, Corporate Technology & University of Munich, Germany
kai.yu.external@mchp.siemens.de

Xiaowei Xu
Information Science Department
University of Arkansas at Little Rock
xwxu@ualr.edu

Anton Schwaighofer
Siemens AG, Corporate Technology & Technical University of Graz, Austria
anton.schwaighhofer.external@mchp.siemens.de

Volker Tresp
Siemens AG, Corporate Technology
Munich, Germany
volker.tresp@mchp.siemens.de

Hans-Peter Kriegel
Institute for Computer Science, University of Munich
Munich, Germany
kriegel@dbs.informatik.uni-muenchen.de

ABSTRACT

The application range of memory-based collaborative filtering (CF) is limited due to CF's high memory consumption and long runtime. The approach presented in this paper removes redundant and inconsistent instances (users) from the data. Our work shows that a satisfactory accuracy can be achieved by using only a small portion of the original data set, thereby alleviating the storage and runtime cost of the CF algorithm. In our approach, we consider instance selection as the problem of selecting informative data that increase the *a posteriori* probability of the optimal model. We evaluate the empirical performance of our approach on two real-world data sets and attain very promising results. Data size and prediction time are significantly reduced, while the prediction accuracy is on a par with results achieved by using the complete database.

Categories and Subject Descriptors

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Computer-supported cooperative work.*

General Terms

Algorithms, Theory

Keywords

Collaborative filtering, Data selection, Efficiency, Scalability

1. Introduction

The tremendous advance of data storage technology together with the sheer growth of electronic business and media has caused an explosive growth of information. In recent years, information filtering technology has emerged to help people handle the problem of information overload. A typical information filtering application concern recommender systems, which attempt to assist users in finding their favorite products (for example, CDs, books or movies) out of thousands or even millions of products offered by a vendor. Recommender systems are commonly based on collaborative filtering (CF). CF accumulates a database of user preferences and uses those to predict a novel user's preferences for unseen items, such as a new CD or movie.

Breese et al. [1998] identify two major classes of CF algorithms, memory-based approaches and model-based approaches. Memory-based CF approaches simply store all the data and defer the generalization (that is, the actual extraction of knowledge from the data) to the prediction phase. In CF literature, memory-based methods have been widely investigated. On the other hand, model-based algorithms first learn a descriptive model from the collected data and then use it to predict user preferences. Comparing model-based CF and memory-based CF, the following points are important to bear in mind: (1) Both approaches have comparable prediction accuracy; (2) Memory-based CF has a clear interpretation: user prefers those items that like-minded people prefer; (3) Memory-based CF can incrementally accommodate the information of new coming data, while typical model-based methods lack this ability. This is a crucial point, since the amount of available data is ever growing; (4) Memory-based CF suffers from slow response time, because each single prediction requires scanning the whole database. Thus, the computational cost and storage space scale linearly with the size of preference data stored for memory-based CF. (5) For model-based approaches, the learning phase (i.e. the time to train the model) may become prohibitively long for large data sets. Summing up, memory-based CF has advantages in terms of interpretability and adaptability. Both approaches have difficulties with large data sets either in terms of training time or in terms of response time and storage requirements.

The goal of the work presented in this paper is to reduce storage requirements and improve on response speed for memory based CF approaches by reducing the size of the instance base. An instance is here defined as a user with associated preference profile. The proposed method effectively removes redundant and inconsistent preference data by using a measure of instance relevance. By using only a small, suitably selected subset of the original data, one can speed up the prediction with only little loss of accuracy and furthermore reduce the storage cost.

This article is organized as follows. After a brief introduction to related work, we first investigate the general problem of instance selection for learning algorithms in Section 3. We propose a likelihood-based relevance measure to carry out instance selection. In Section 4, we introduce a probabilistic framework for CF. Based on the results of Section 3, we describe an algorithm, *profile filtering* (PF), to select informative instances for memory-based CF. In Section 5 we evaluate the PF algorithm on two datasets, EachMovie and MsWeb. The experimental results clearly demonstrate that PF significantly reduces the size of training data, while retaining an accuracy that is comparable to predictions made using the whole set of preference data. We end by giving conclusions and an outlook to future work

2. Related Work

Collaborative filtering (CF) has been a lively research area in recent years and proved successful in practice. A variety of algorithms have been proposed. The first CF algorithms were based on the observation that people usually trust the recommendations from like-minded friends. The Grouplens [Resnick et al, 1994] and Ringo [Shardanand and Maes, 1995] systems applied memory-based algorithms to help users to automatically find like-minded users and combine their opinions to make predictions. Different metrics to measure the preference similarity between users have been suggested, including Pearson coefficients [Resnick et al, 1994], constraint Pearson coefficients [Shardanand and Maes, 1995], vector similarity [Breese et al, 1998], and personality type [Pennock et al, 2000]. Model-based CF, in contrast, uses the user's preference database to learn a model, which is then used for predictions. The model is built off-line over a matter of hours or days. The resulting models are typically small, fast, and essentially as accurate as memory-based methods. Examples include Bayesian networks [Breese et al, 1998], clustering techniques [Breese et al., 1998; Ungar & Foster, 1998], neural networks [Billsus and Pazzani, 1998], induction rule learning [Basu et al, 1998], and linear classifiers [Zhang & Iyengar, 2001]. Breese et al. [1998] compare several memory-based methods and model-based methods. Their results indicate that the performances of memory-based methods based on the Pearson coefficient and Bayesian networks are comparable and better than the alternative approaches under study.

Up to now, work on CF mainly focuses on the issue of prediction accuracy. Other important aspects, like scalability, incremental processing, and interactive CF have received only little attention.

Instance selection has been a subject of extensive studies in the area of memory-based (instance-based) learning [Liu and Motoda, 2001]. Similarly to memory-based CF, memory-based learning methods simply store all the training instances. They reply to classification queries by evaluating the similarity to their stored instances. Instance selection algorithms usually seek to select representative instances out of the whole training set. Depending on the method, "representative" points may be border points, e.g. IB2 [Aha et al, 1991] or central points [Zhang, 1992]. The intuition behind retaining border points is that "internal" points do not affect the decision boundaries as much as border points, and thus can be removed. However, noisy points are prone to be judged as border points and added to the training set. Selecting central points as representative is less sensitive to noise, yet it may fail to characterize the decision boundary accurately. Another class of algorithms attempts to remove noisy points by considering the labels of their neighbors, e.g. DROP3 in [Wilson and Martinez, 2000]. The studies of instance selection for instance-based learning are closely related to the work we present in this article.

Preliminary work on instance selection techniques for memory-based CF has been presented in [Yu et al, 2002], where two separate methods are proposed to remove redundant and inconsistent instances, respectively. They use different instance bases when predicting ratings on different items and thus do not really reduce the amount of data. This paper addresses the two kinds of removals by using a unified likelihood-based relevance measure and proposes a novel method to actually reduce the data. In addition, this paper provides a meaningful probabilistic point of view to look at the data selection problem in memory-based CF.

3. Instance Relevance

In this section we discuss the role of instances and draw some general conclusions about instance relevance, which are applicable to a range of probabilistic model-based learning algorithms. We will make the connection to memory-based CF in Section 4, where we show how memory-based CF can also be interpreted as a probabilistic model-based learning approach. The conclusions drawn in this section provide a theoretic preparation for our data selection algorithm, which will be introduced in Section 4.

3.1 MAP hypothesis

Bayesian methods provide a general perspective for understanding many learning algorithms, including those that do not explicitly manipulate probabilities. We begin by introducing the learning problem as well as necessary notations. The goal is to learn a target function $f : X \rightarrow Y$, which is a mapping from attribute space X to class label space Y. A training data set D is given, which consists of m instances $\{d_1 \dots d_m\}$, where each instance is an attribute/label pair $d_i = (x_i, y_i)$. In many learning scenarios, the learning algorithm L considers a candidate hypothesis space H and aims at finding the most probable hypothesis $h \in \mathrm{H}$ given the observed data D which is called the *maximum a posteriori* (MAP) hypothesis:

$$h_{MAP} \equiv \arg\max_{h \in \mathrm{H}} P(h \mid D) \tag{1}$$

3.2 Likelihood-Based Instance Relevance

We consider an iterative learning scheme, in which only relevant new data are added to the already existing training data set. Let D^t be the data set selected until iteration t, let D^{-t} denote the remaining data, and let $h^t_{MAP} \in \mathrm{H}$ be the *MAP hypothesis* derived from D^t. Then for any hypothesis $h \in \mathrm{H}$, using Bayes' theorem we have

$$P\left(h \mid D^{t+1}\right) = P\left(h \mid D^t, d_{t+1}\right) = \frac{P(d_{t+1} \mid h)P(h \mid D^t)}{\int_{h \in \mathrm{H}} P(d_{t+1} \mid h)P(h \mid D^t)dh} \tag{2}$$

Using the Laplace approximation [Heckerman, 1995], we derive the following approximate expression:

$$\frac{P(d_{t+1}\mid h)P(h\mid D^t)}{\int_{h\in H} P(d_{t+1}\mid h)P(h\mid D^t)dh} \approx C\cdot\frac{P(d_{t+1}\mid h)P(h\mid D^t)}{P(d_{t+1}\mid h^t_{MAP})} = C\cdot P(h\mid D^t)\frac{P(d_{t+1}\mid h)}{P(d_{t+1}\mid h^t_{MAP})} \quad (3)$$

where C is a term independent of h and d_{t+1}. Since the goal of learner L is to output h_{MAP} based on the complete data set D (see Eq.(1)), we aim at finding $d_{t+1}\in D^{-t}$ such that h_{MAP} becomes more likely when learner L observes the subset $D^{t+1}=\{D^t, d_{t+1}\}$. Because Eq.(2) and (3) are correct for any h in H, we replace h in Eq.(2) and (3) by h_{MAP} and obtain an interesting result:

$$\frac{P(h_{MAP}\mid D^{t+1})}{P(h_{MAP}\mid D^t)} \approx C\cdot\frac{P(d_{t+1}\mid h_{MAP})}{P(d_{t+1}\mid h^t_{MAP})} \quad (4)$$

Eq.(4) shows that the increase of the probability of h_{MAP} after observing d_{t+1} is proportional to the likelihood ratio of d_{t+1} with respect to h_{MAP} and h^t_{MAP}. Therefore the *optimal* data point is the one that can maximize the probability of h_{MAP}.

$$\begin{aligned} d_{t+1} &= \arg\max_{d\in D^{-t}} \frac{P(h_{MAP}\mid D^{t+1})}{P(h_{MAP}\mid D^t)} \\ &\approx \arg\max_{d\in D^{-t}} \frac{P(d\mid h_{MAP})}{P(d\mid h^t_{MAP})} \\ &= \arg\max_{d\in D^{-t}} R^t(d) \end{aligned} \quad (5)$$

where

$$\begin{aligned} R^t(d) &= \log P(d\mid h_{MAP}) - \log P(d\mid h^t_{MAP}) \\ &= \log P(x,y\mid h_{MAP}) - \log P(x,y\mid h^t_{MAP}) \\ &= \log P(y\mid h_{MAP},x) - \log P(y\mid h^t_{MAP},x) \\ &\quad + \log P(x\mid h_{MAP}) - \log P(x\mid h^t_{MAP}) \\ &= \log P(y\mid h_{MAP},x) - \log P(y\mid h^t_{MAP},x) \end{aligned} \quad (6)$$

In Eq.(6) we adopt a common assumption that input data x of instance d is independent of any hypothesis in H. It is clear from Eq.(5) that the probability of h_{MAP} increases after observing an instance $d\in D^{-t}$ with positive $R^t(d)$, whereas an instance d with zero or negative $R^t(d)$ will not influence or decrease the probability of h_{MAP}. Based on these observations, we look upon $R^t(d)$ defined in Eq.(6) as a measure of instance relevance with respect to the target hypothesis h_{MAP} on the base of the previously selected data D^t. Since $R^t(d)$ is instance d's *logarithmic likelihood ratio* with respect to the two hypotheses h_{MAP} and h^t_{MAP}, we call $R^t(d)$ the likelihood-based instance relevance (LIR) measure. A large LIR measure indicates that the corresponding instance has a high contribution to the likelihood of the target hypothesis h_{MAP}.

Some intuitive interpretations may be helpful to better understand the proposed LIR measure.

- *Redundant instance*: $\log P(d|h^t_{MAP})$ encodes the log-likelihood of instance d with respect to the current hypothesis h^t_{MAP}. From Eq.(6), we can see that instances with lower $P(d|h^t_{MAP})$ are more relevant to increase the probability of h_{MAP}. For example, a student in real life would prefer to learn facts that are novel or unknown to his/her current knowledge. Similarly, a learning algorithm can gain more information from instances that are unlikely given the current hypothesis. Patterns with currently high probability are considered *redundant* since the learner already knows about these patterns.
- *Inconsistent instances:* $\log P(d|h_{MAP})$ encodes the log-likelihood of instance d with respect to the target hypothesis h_{MAP}. From Eq.(6), we see that instances with higher $P(d|h_{MAP})$, are more likely to increase the probability of h_{MAP}. $-\log P(d|h_{MAP})$ actually measures the degree of mismatch between the instance d and the target hypothesis h_{MAP}. Instances with low log-likelihood with respect to the target hypothesis h_{MAP} are considered to be noise because they are *inconsistent* with the target model.
- $R^t(d) = \log P(d|h_{MAP}) - \log P(d|h^t_{MAP})$ combines the two above perspectives and encodes the overall relevance of instance d with respect to h_{MAP} and h^t_{MAP}. It indicates an instance's contribution to increasing the probability of h_{MAP}. An intuitive interpretation to this integral relevance measure is that the instances with high $R^t(d)$ are those that are considered as novel and consistent by the learner. Mind that the two terms making up $R^t(d)$ can not be treated separately, since noisy or inconsistent instances are always judged as novel.

Based on the above discussions, we give the definition of instance relevance.

Definition 1. (*Relevant instance and irrelevant instance*) Let $h_{MAP}\in H$ be the MAP hypothesis in H given the whole data D, and D^t the selected data until iteration t. Then instance d is relevant with respect to h_{MAP} given selected data D^t, if its instance relevance measured by Eq.(6) is positive, otherwise it is irrelevant.

In the next section, we will introduce a data selection algorithm for memory-based CF.

4. Instance Selection for Memory-Based CF

We now turn our attention to collaborative filtering (CF) for recommender systems. A general probabilistic model for the widely applied memory-based CF algorithms will be introduced. In Section 4.2, we describe an algorithm named *profile filtering* (PF) to identify informative instances in large user preference database and form a reduced instance base for CF.

4.1 A Probabilistic Model

Memory-based CF maintains a database of user ratings on items, denoted by $\boldsymbol{V}$, and predicts the active (query) user's ratings on not yet rated items based on the ratings of other like-minded users in $\boldsymbol{V}$. Let $\boldsymbol{V}$ be an $n\times m$ user rating matrix with entries $V_{i,j}$ being the rating of user i on item j, where n is the number of users, $\boldsymbol{T}$ the set of all items, and $m = |\boldsymbol{T}|$ the total number of items. If we denote the set of possible rating values as $\boldsymbol{R}$, then each entry $V_{i,j}\in \boldsymbol{R}\cup\{\emptyset\}$ is either an actual score or Ø, indicating a 'missing value'. Since each user typically only rated a small number of items, matrix $\boldsymbol{V}$ normally has a large number of missing-valued entries. More specifically, the item set rated by user i is denoted as $\boldsymbol{T_i}\subseteq \boldsymbol{T}$ and the set of items not rated by user i is denoted as $\boldsymbol{N_i}\subseteq \boldsymbol{T}$. In the following text, $\boldsymbol{V_i}$ is used to represent the row vector $\{V_{i,1}, V_{i,2},\ldots, V_{i,m}\}$ containing all ratings of user i.

Then given the active user a's ratings $\boldsymbol{V_a}$, a probabilistic CF method computes the *a posteriori* probability distribution of the active user's rating on the items $j\in \boldsymbol{N_a}$:

$$P(\ V_{a,j} = x \mid \boldsymbol{V_a},\ \boldsymbol{V}\ ) \qquad \text{where } x\in \boldsymbol{R}\ ,\ j\in \boldsymbol{N_a} \tag{7}$$

The prediction of the active user's rating on item *j* is computed as the expected value of *x* with respect to the posterior distribution:

$$\sum_{x\in\boldsymbol{R}} x \cdot P(V_{a,j} = x \mid \boldsymbol{V_a}, \boldsymbol{V}) \tag{8}$$

CF algorithms differ in the way of estimating the probability given by Eq.(7). A typical model-based CF that explicitly calculates Eq.(7) is described by [Breese et al, 1998] and is based on Bayesian Belief Networks [Breese et al, 1998]. In this paper we will focus on the class of memory-based CF methods.

Pennock et al [2000] introduce a vector of "true" ratings of all seen items to describe the user *i*'s personality type. We follow the line of Pennock et al. and generalize it to a wide probabilistic framework of memory-based CF methods.

Treating each user's ratings $\boldsymbol{V_i}\in \boldsymbol{V}$ as a prototype preference pattern *i*, we can rewrite the probability distribution Eq.(7) as the following:

$$P(V_{a,j} = x \mid \boldsymbol{V_a}, \boldsymbol{V}) = \sum_{i=1}^{n} P(V_{a,j} = x \mid i) P(i \mid \boldsymbol{V_a}, \boldsymbol{V}) \tag{9}$$

where $P(i|\boldsymbol{V_a},\ \boldsymbol{V})$ is the probability that the active user *a* has preference pattern *i*. Pennock et al. apply a naïve Bayesian approach to explicitly estimate the likelihood $P(i|\boldsymbol{V_a},\ \boldsymbol{V})$. Most other memory-based approaches use some similarity measure between users to implicitly derive the likelihood in the following way:

$$P(i \mid \boldsymbol{V_a}, \boldsymbol{V}) = \frac{\text{Similarity}(\boldsymbol{V_a}, \boldsymbol{V_i})}{\sum_{k=1}^{n} \text{Similarity}(\boldsymbol{V_a}, \boldsymbol{V_k})} \tag{10}$$

$$P(V_{a,j} = x \mid i) = \delta(V_{i,j}, x) = \begin{cases} 1 & \text{if } V_{i,j} = x \\ 0 & \text{otherwise} \end{cases}$$

For the similarity function we only have to assume that it is non-negative, symmetrical about $\boldsymbol{V}_i$ and that the integral over $\boldsymbol{V}_a$ is equal for all $\boldsymbol{V}_i$. Then Eq.(9) can be written as:

$$P(V_{a,j} = x \mid \boldsymbol{V_a}, \boldsymbol{V}) = \frac{\sum_{i=1}^{n} \delta(V_{i,j}, x)\text{Similarity}(\boldsymbol{V_a}, \boldsymbol{V_i})}{\sum_{i=1}^{n} \text{Similarity}(\boldsymbol{V_a}, \boldsymbol{V_i})} \tag{11}$$

The Pearson correlation coefficient (see [Resnick et,al. 1994]) is considered to be the most successful similarity measure for CF. In our work all the negative Pearson coefficients are changed to be zero to guarantee that the similarity measure is non-negative. Once the posterior probabilities Eq.(11) have been estimated, the final prediction is given by Eq.(8). Obviously, the computational complexity of evaluating Eq.(11) is O(*nm*). In the probabilistic framework of memory-based CF, the $n \times m$ matrix $\boldsymbol{V}$ can be viewed as a model or hypothesis containing *n* prototype of preference patterns.

4.2 Profile Filtering: Removing Redundant and Inconsistent Preference Patterns

We now make the connection back to the discussion of Section 3.2, where we have discussed the relevance of a given training example with respect to initial and current model. CF can be viewed as a typical learning problem, which is to find the target function *f*: $\boldsymbol{V}^{\text{seen}} \rightarrow \boldsymbol{V}^{\boldsymbol{unseen}}$, a mapping from the observed ratings to the unobserved ratings of users. Thus the training instance or user *i*'s ratings $\boldsymbol{V_i}$ can be formalized into the standard input-output instance form $(\boldsymbol{V}^*_{i,j}\ ,\ V_{i,j})$, $j\in \boldsymbol{T_i}$, where $\boldsymbol{V}^*_{i,j}$ denotes the attribute vector derived by hiding the entry $V_{i,j}$ of $\boldsymbol{V_i}$, and $V_{i,j}$ is the corresponding instance label. Let $\boldsymbol{V}^t$ denote the selected data at the *t*-th iteration and also the MAP hypothesis derived from it. Thus according to Eq.(6) the likelihood-based instance relevance(LIR) measure of instance $\boldsymbol{V_i}\,(\boldsymbol{V}^*_{i,j}\ ,\ V_{i,j})$ with respect to the hypotheses $\boldsymbol{V}$ and $\boldsymbol{V}^t$ is:

$$R^t(V_{i,j}, \boldsymbol{V}^*_{i,j}) = \log P(V_{i,j} \mid \boldsymbol{V}^*_{i,j}, \boldsymbol{V}) - \log P(V_{i,j} \mid \boldsymbol{V}^*_{i,j}, \boldsymbol{V}^t) \tag{12}$$

To avoid negative infinite log-likelihood in computing Eq.(12), we set a small constant to replace the zero likelihood. Consider all the seen ratings in $\boldsymbol{V_i}$ can serve as instance labels, we further calculate the averaged LIR measure of instance $\boldsymbol{V_i}$ over all seen ratings of $\boldsymbol{V_i}$:

$$\begin{aligned} R^t_{avrg}(\boldsymbol{V}_i) &= \frac{1}{|\boldsymbol{T}_i|} \sum_{j\in T_i} \left(\log P(V_{i,j} \mid \boldsymbol{V}^*_{i,j}, \boldsymbol{V}) - \log P(V_{i,j} \mid \boldsymbol{V}^*_{i,j}, \boldsymbol{V}^t) \right) \\ &= \frac{1}{|\boldsymbol{T}_i|} \sum_{j\in T_i} \frac{\log P(V_{i,j} \mid \boldsymbol{V}^*_{i,j}, \boldsymbol{V})}{\log P(V_{i,j} \mid \boldsymbol{V}^*_{i,j}, \boldsymbol{V}^t)} \end{aligned} \tag{13}$$

Combining Eq.(11) and Eq.(13), the LIR measure of instance $\boldsymbol{V_i}$ in the context of $\boldsymbol{V}$ and $\boldsymbol{V}^t$ can be explicitly calculated. The *profile filtering* (PF) algorithm, as shown in Table1, starts out with the first instance $\boldsymbol{V}_1$ of user preference database $\boldsymbol{V}$ as the initial hypothesis. For each following instance, PF computes the instance relevance, as given by Definition 1 and Eq.(13) and adds them to the instance base if the instance is considered relevant, that is, if its R^t_{avrg} is positive. The behavior of PF can be intuitively interpreted as seeking instances with novel profile, thereby eliminating redundant instances whose profile has already been learned. On the other hand, it also avoids instances that are inconsistent with other instances in $\boldsymbol{V}$**.** Finally the resulting instance base $\boldsymbol{V}^{out}$ will be much smaller than the original one, while the accuracy of CF prediction based on $\boldsymbol{V}^{out}$ should be not significantly degraded.

Table 1: The algorithm of *Profile Filtering* for memory-based CF

```
Input:    $\boldsymbol{V}$ is the $n \times m$ matrix of $n$ user ratings on $m$ items
Output:   $\boldsymbol{V}^{out}$ is the resulted instance base
Function: Profile Filtering ($\boldsymbol{V}$)
Initialize the selected instance base $\boldsymbol{V}^t \leftarrow \boldsymbol{V}_1$, $t \leftarrow 1$
$i \leftarrow 2$
For each instance $\boldsymbol{V_i}$ in $\boldsymbol{V}$
    Compute: $R^t_{avrg}(\boldsymbol{V_i}) = \frac{1}{|T_i|}\sum_{j\in T_i} \log \frac{P(\boldsymbol{V}_{i,j} \mid \boldsymbol{V}^*_{i,j}, \boldsymbol{V})}{P(\boldsymbol{V}_{i,j} \mid \boldsymbol{V}^*_{i,j}, \boldsymbol{V}^t)}$
    If $R_i > 0$, then $\boldsymbol{V_i}$ is added into $\boldsymbol{V}^t$, and $t \leftarrow t + 1$
    $i \leftarrow i + 1$
End for
$\boldsymbol{V}^{out} \leftarrow \boldsymbol{V}^t$
```

5. Empirical Study

In this section we evaluate the *profile filtering* (PF) algorithm on two benchmark datasets: the EachMovie database, available from the Digital Equipment Research Center[1], and the MsWeb dataset, available from the UCI KDD data depositary [Hettich and Bay, 1999].

Table 2: Summary of the data sets

	Eachmovie	Msweb
Training instances	30000	32711
Test instances	5000	5000
Items	1628	297
Mean votes per training instance	35.5	3.02
Mean votes per test instance	38.2	3.01

5.1 Data Sets and Experiment Setup

Eachmovie contains ratings from 72,916 users on 1,628 movies. User ratings were recorded on a numeric six-point scale from zero to five. We select the first 35000 users in EachMovie data set and divide them into training set and test set. The second benchmark data set, MsWeb, was introduced by [Breese et al., 1998] and added to the UCI repository under the name *anonymous-msweb*. It contains for each user the web page groups (called vroots) that were visited in a fixed time period. We use 0 to represent 'not visited' and 1 to represent 'visited'. Eachmovie and MsWeb represent two kinds of typical datasets in recommender systems. EachMovie contains preference data explicitly indicated by numeric ratings, whereas MsWeb contains implicit user preferences indicated by usage (e.g. 'visited', 'purchased' et al.). A summary of the two data sets is given in Table 2.

In our experiments, we use the PF algorithm described in Section 4 to reduce the size of the instance base. We then evaluate the prediction quality of a memory-based CF algorithm using the reduced instance base. We compare the results of PF with three other methods of instance selection:

- No Sampling (NoSamp): Predicting user preferences using the whole data
- Random Sampling (RandSamp): Predicting user preferences using a reduced instance base that is a random subset of the original (full) data. For RandSamp, we present results that are averaged by 5 random samplings.
- Rating-based Sampling (RateSamp): Predicting user preferences using a reduced instance base obtained by selecting those users that have rated most items (largest set Ti, see Section 4.1). This sampling strategy follows the intuition that instances with the most ratings are the most informative.

In all the experiments we use the Pearson correlation coefficient [Resnick et,al. 1994] as the similarity measure in Eq.(10). Negative coefficients are set to small positve values (see section 4.1). We will consider different application scenarios for the two types of data sets. For the EachMovie data set, the task is to explicitly predict the user's ratings on particular movies. For the MsWeb data, the goal is to rank user preferences on all vroots and then recommend the top ones to the user.

5.2 EachMovie: Predicting User Ratings

In this set of experiments, we evaluate the accuracy of predicting user ratings on particular movies, when only a reduced set of training data is available. We compare the results obtained by reducing the training data through random sampling, rating-based sampling, our proposed PF algorithm and a setting where the full training data is used (see previous section). In order to evaluate the quality of prediction, we follow the experimental setup in [Shardanand and Maes, 1995], in which a certain percentage of each user's ratings in the test set were hidden. We randomly select 70% of a test user's ratings "seen" ratings and use them to predict the rest of 30% hidden ratings. We use *mean absolute error* (MAE) and *e-accuracy* to evaluate the accuracy of prediction. MAE is the average difference between the actual ratings and the predicted ratings. This metric has been widely used in previous work [Breese et al., 1998; Herlocker et al., 1999; Pennock et al. 2000 Resnick et al., 1994; Shardanand and Maes, 1995]. *e-accuracy* is the percentage of tests whose absolute error is less than *e*. We believe it provides more information about the distribution of errors. In particular, when *e* is set to be 0.5, the rounded value of a successful prediction exactly equals the actual rating. It has been argued that CF accuracy is most crucial when predicting extreme ratings (very high or very low) for products [Pennock et al. 2000; Shardanand and Maes, 1995]. Intuitively, since the goal is to provide recommendations, high accuracy on the high rated and low rated products is most preferred. Therefore we also investigate the accuracy in predicting extreme ratings (*Extremes*), where the actual rating is 0,1,2, or 5. (This choice results from the observation that more than 50% of ratings in EachMovie are 3 or 4.)

Table 3. Prediction accuracy (EachMovie)

Method	All			Extremes		
	MAE	0.5-Accu.	1.0-Accu.	MAE	0.5-Accu.	1.0-Accu.
NoSamp	**0.915**	**0.350**	**0.636**	**1.27**	**0.148**	**0.404**
RandSamp	0.962	0.326	0.613	1.33	0.127	0.369
RateSamp	0.945	0.331	0.621	1.31	0.132	0.382
PF	**0.916**	**0.351**	**0.634**	**1.27**	**0.149**	**0.403**

[1] http://www.research.digital.com/SRC/EachMovie/.

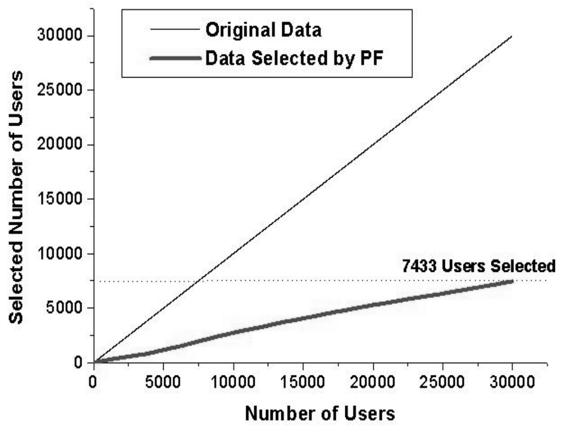

Figure 1. Selected users vs. original users (EachMovie)

We use the PF algorithm to filter the preference data and then evaluate the accuracy of memory-based CF using the reduced data. The PF algorithm selects a total of 7433 users from the preference data of 30000 users, giving a selection rate of 24.8%. For comparison, we also select 7433 users by random sampling (RandSamp) and rating-based sampling (RateSamp) to construct instance bases for comparison.

Table 3 shows the prediction accuracy of NoSamp, RandSamp, RateSamp and PF. It can be clearly seen that the prediction quality of memory-based CF using an instance base that is selected by our proposed PF algorithm is almost the same as the accuracy achieved by using the whole database. This holds for all measures of accuracy we have evaluated (MSE, e-accuracy, both on the full set of values and on the extreme values). Among the different selection schemes, random sampling (RandSamp) shows the worst prediction quality. The results of rating-based sampling (RateSamp) are slightly better than RandSamp, but still much worse than the benchmark results without sampling (NoSamp). The observation indicates that instances with more ratings are not necessarily more informative.

After having examined all the training data, the PF algorithm had selected a subset of 7433 (24.8%) relevant instances. Since the time complexity of prediction is O(*nm*), a selection rate of 24.8% leads to a speed-up of 4 in the prediction phase. Figure 1 shows for the PF algorithm how the number of selected users grows with the training data.

Summing up, our profile-filtering algorithm has proven to be the best sampling strategy among all compared methods. The PF algorithm achieves an accuracy that is the same as the accuracy achieved without sampling, yet provides a reduction by a factor of 4 in terms of prediction speed and memory consumption on the EachMovie data.

5.3 MsWeb: Ranking User Preferences

For the MsWeb data, we use essentially the same experimental setup as for the EachMovie data (see Section 5.2). For evaluating the performance, we randomly hide 30% of the visited vroots for each test user and predict the ratings for hidden vroots as well as unvisited vroots. We rank the predicted vroots according to the predicted ratings and recommend the top 5 to the active user. A successful recommendation is required to hit those 30% hidden vroots that are actually visited by the active user. Three metrics, *Recall*, *Precision* and *Success Rate* are applied for evaluations. *Recall* is the percentage of items liked by a user that are recommended to him/her; *Precision* is the percentage of items recommended to a user that the user likes; *Success Rate* is the percentage of cases that at least one liked items is recommended to the user. In the MsWeb data set, liked items are visited web pages.

Table 4 shows the prediction accuracy of CF using the whole preference data (NoSamp) and that of the sampling strategies RandSamp, RateSamp and PF. The PF algorithm selects a total of 3648 users, we thus also selected 3648 users by random sampling and rating-based sampling for comparison. Again, the prediction accuracy of PF is almost the same as that of the CF algorithm using the original data, while the results of RandSamp and RateSamp are much worse. Since the prediction algorithm has time and space complexity *O*(*nm*), the PF algorithm with selection rate of 12.2% will speed up recommendations by a factor of 8.2.

Figure 2 shows the growth of instances selected by PF as more and more training data is examined.

Table 4: Ranking accuracy (MsWeb)

Method	Recall	Precision	Success
NoSamp	**0.564**	**0.202**	**0.742**
RandSamp	0.501	0.180	0.669
RateSamp	0.530	0.191	0.697
PF	**0.561**	**0.201**	**0.736**

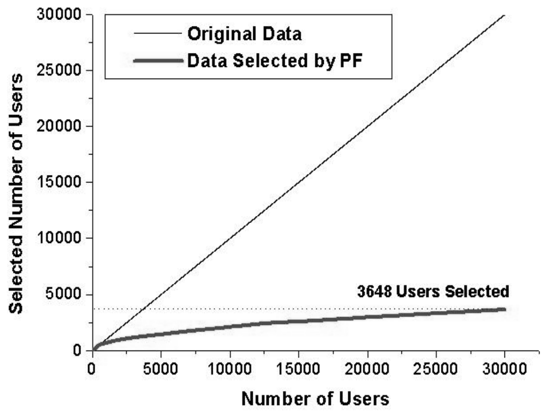

Figure 2. Selected users vs. original users (MsWeb)

5.4 Complexity and Scalability

The reduction rate of *PF* for MsWeb is more significant than the EachMovie dataset. This may be explained by the fact that the complexity of MsWeb is much lower than that of Eachmovie. EachMovie has 1628 dimensions (items) and 6 possible values along each dimension, while MsWeb has only 297 dimensions and 2 possible values.

In general it is required that time and space complexity of scalable algorithms increase at most linearly with the size of processed data. If we carefully compare the graphs in Figures 1 and 2, we observe that in Figure 1 the number of selected instances of EachMovie data scales almost linearly with the size of the original data, while the number of selected instances for MsWeb grows "sub-linearly". Since both time and space complexity of the prediction scale linearly with the size of the selected instance base, prediction scales linearly with the size of EachMovie data and sub-linearly with the size of MsWeb data.

This may be explained as follows. As pointed out in Section 4.2, the PF algorithm removes redundant instances with profiles that have already been learned. Data sets with rather low complexity, such as MsWeb, may be described sufficiently by the data at hand. For "learnable" problems, the probability of observing newly arriving redundant instances is getting higher, while learned instances are getting more and more sufficient to describe the distribution of the full data.[2] This results in the sub-linear growth observed in Figure 2. On the other hand, for the highly complex EachMovie data, the probability of encountering redundant instances remains almost constant during the instance selection process. The given data set of 30.000 EachMovie examples is still far from being sufficient for describing the high complexity of the data. We can expect a similar behavior of encountering more redundant instances only if more EachMovie data are considered.

Summing up, we attribute the attractive scaling behavior of the proposed PF algorithm to its ability of removing redundant instances. By removing redundant instances, the description of the overall data is based on only a subset of data, the size of which is in turn related to the inherent complexity of the problem considered. PF helps to extract this sufficient subset of the data and may lead to a sub-linear growth of prediction complexity.

5.5 Sensitivity to the Order of Instances

Now we discuss a weakness of our proposed method. PF is a rather fast algorithm since each instance is only evaluated one, i.e. the data set is simply filtered. The disadvantage is that since PF is a sequential algorithm, it naturally depends on the order in which instances are presented. We perform PF in the reverse order of EachMovie and MsWeb data sets. The results together with the results obtained in origin order are compared in Table 5 and Table 6. It turns out that instance ordering is not a severe problem for the PF algorithm in terms of accuracy. However, the size of selected data is sensitive to the order of presenting instances. If PF is run in the reverse order, 33.2% of the EachMovie data and 18.6% of the MsWeb data are retained, whereas 24.8% of the EachMovie data and 12.2% of the MsWeb data are retained in the original order. This may be partially explained by the following examples. If many highly informative instances happen to be observed by PF at the start phase, then fewer consistent instances are needed by PF in the following phase. In contrast, if many slightly relevant instances are encountered in the start phase, then more instances are needed later. Thus PF's result of selected data size may heavily dependent on the order of presenting instances. We need to alleviate this dependence in our future work. Fortunately, the experiments demonstrate that the prediction accuracy using PF-selected instance base is not sensitive to the order. The accuracy of reverse order is also almost the same with the results of memory-based CF using the full data. This is because that each selected instance consistently increases the *a posteriori* probability of the optimal user preference model.

[2] More rigorous details about sampling complexity can be found in [Mitchell, 1997].

Table 5. PF in different orders (EachMovie)

Method	All			Extremes			Selection Rate
	MAE	0.5-Accu.	1.0-Accu.	MAE	0.5-Accu.	1.0-Accu.	
PF	0.916	0.351	0.634	1.27	0.149	0.403	24.8%
PF *	0.918	0.347	0.630	1.28	0.149	0.405	33.2%

*PF performed in the reverse order of instances

Table 6. PF in different orders (MsWeb)

Method	Recall	Precision	Success	Selection Rate
PF	0.561	0.201	0.736	12.2%
PF *	0.562	0.202	0.740	18.6%

*PF performed in the reverse order of instances

6. Conclusions and Future Work

Memory-based collaborative filtering (CF) methods have proven to be effective in predicting user preferences. Memory based CF typically suffers from high response time and high storage cost due to its large instance base. In this paper we have investigated the problem of instance selection for memory-based CF. We have proposed the *profile filtering* (PF) algorithm to reduce the size of the instance base that is used in a CF method. A likelihood-based formalism is proposed to measure the instance relevance, which then allows the *profile filtering* algorithm to remove both redundant and inconsistent instances from the instance base. Our experiments have shown that 24.8% of the EachMovie data and 12.2% of the MsWeb data are retained by the *profile filtering* algorithm, leading to reduction of prediction time by a factor of 4 and 8, respectively. Despite the large reduction of the instance base, predictions using the pruned data can be made with an accuracy that is almost as high as that of predictions using the complete data. These results demonstrate that the proposed *profile filtering* algorithm can effectively improve the scalability of memory-based CF. In addition, the probabilistic model of memory-based CF provides a deeper insight into the method and opens many opportunities to improve the traditional memory-based CF. One particularly interesting topic is to equip CF systems with an active learning component that can query users for information. The goal is to quickly gain the most important knowledge about the users preferences. We plan to extend our current work to this topic. As pointed out in Section 5.5, we also need to improve the *profile filtering* algorithm by alleviating its dependence on the order of observing instances.

7. REFERENCES

[Aha et al., 1991] D. W. Aha, D. Kibler and M. K. Albert, “Instance-based Learning Algorithms”, *Machine Learning*, 6: 37-66, 1991.

[Billsus & Pazzani, 1998] D. Billsus and M. J. Pazzani, “Learning Collaborative Information Filters”, In *Proceedings of the International Conference on Machine Learning*, 1998.

[Breese et al., 1998] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering”, In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998.

[Heckerman, 1995] D. Heckerman, “A Tutorial on Learning with Bayesian Networks”, Technical Report MSR-TR-95-06, Microsoft Research, 1995.

[Hettich & Bay, 1999] S. Hettich and S. D. Bay. The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science, (1999).

[Liu & Motoda, 2001] H. Liu and H. Motoda (Eds.), “Instance Selection and Construction for Data Mining”, Norvell, MA: Luwer Academic Publishers, 2001.

[Mitchell, 1997] T Michell, “Machine Learning”, McGraw Hill, 1997, pp169, 201-227

[Pennock, et al. 2000] D. M. Pennock, E. Horvitz, S. Lawrence, S., Giles, C.L.: Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach. In Proc. of the 16th Conference on Uncertainty in Artificial Intelligence (2000) 473-480

[Resnick, et al. 1994] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, “GroupLens: An Open Architecture for Collaborative Filtering of Netnews”, In *Proceedings of the 1994 Computer Supported Collaborative Work Conference*.

[Shardanand & Maes, 1995] U. Shardanand, and P. Maes, “Social Information filtering Algorithms for Automating 'Word of Mouth'”, In *Proceedings of CHI'95*.

[Wilson & Martinez, 2000] D. R. Wilson and T. R. Martinez, “Reduction Techniques for Instance-Based Learning Algorithms”, *Machine Learning*, 38-3, pp. 257-286, 2000.

[Yu et al, 2002] K. Yu, X. Xu, J. Tao, M. Ester, H.-P. Kriegel, “Instance Selection Techniques for Memory-Based Collaborative Filtering”, *Proc. 2nd SIAM Int. Conf. on Data Mining (SDM'02),* 2002

[Zhang, 1992] J. Zhang, “Selecting Typical Instances in Instance-Based Learning”, in *Proceedings of the Ninth International Conference on Machine Learning*, Aberdeen, Scotland: Morgan Kaufmann, pp. 470-479.

[Zhang & Iyengar, 2001] T. Zhang and V. S. Iyengar, “Recommender Systems Using Linear Classifiers”. *Journal of Machine Learning Research* 2 (2002) 313-33