Learning Collaborative Information Filters

Daniel Billsus and **Michael J. Pazzani**
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
{dbillsus, pazzani}@ics.uci.edu

Abstract

Predicting items a user would like on the basis of other users' ratings for these items has become a well-established strategy adopted by many recommendation services on the Internet. Although this can be seen as a classification problem, algorithms proposed thus far do not draw on results from the machine learning literature. We propose a representation for collaborative filtering tasks that allows the application of virtually any machine learning algorithm. We identify the shortcomings of current collaborative filtering techniques and propose the use of learning algorithms paired with feature extraction techniques that specifically address the limitations of previous approaches. Our best-performing algorithm is based on the singular value decomposition of an initial matrix of user ratings, exploiting latent structure that essentially eliminates the need for users to rate common items in order to become predictors for one another's preferences. We evaluate the proposed algorithm on a large database of user ratings for motion pictures and find that our approach significantly outperforms current collaborative filtering algorithms.

1 INTRODUCTION

Research on intelligent information agents in general, and recommendation systems in particular, has recently attracted much attention. The reasons for this are twofold. First, the amount of information available to individuals is growing steadily. Information overload has become a popular buzzword of our times and people feel overwhelmed when navigating through today's information and media landscape. This leads to a clear demand for automated methods, commonly referred to as intelligent information agents, that locate and retrieve information with respect to users' individual preferences. Second, the number of users accessing the Internet is also growing. Not only does this lead to an incredible variety of subjects that can be learned about online, it opens up new possibilities to organize and recommend information. The central idea here is to base personalized recommendations for users on information obtained from other, ideally like-minded, users. This is commonly known as collaborative filtering or social filtering.

The underlying techniques used in today's recommendation systems fall into two distinct categories: content-based and collaborative methods. Content-based methods require textual descriptions of the items to be recommended and draw on results from both information retrieval and machine learning research (e.g., Pazzani and Billsus, 1997). In general, a content-based system analyzes a set of documents rated by an individual user and uses the content of these documents, as well as the provided ratings, to infer a profile that can be used to recommend additional items of interest. In contrast, collaborative methods recommend items based on aggregated user ratings of those items, i.e. these techniques do not depend on the availability of textual descriptions. Both approaches share the common goal of assisting in the user's search for items of interest, and thus attempt to address one of the key research problems of the information age: locating needles in a haystack that is growing exponentially.

In this paper we focus on collaborative filtering techniques. A variety of algorithms have previously been reported in the literature and their promising performance has been evaluated empirically (Shardanand and Maes, 1995; Hill et al. 1995; Resnick et al. 1994). These results, and the continuous increase of people connected to the Internet, led to the development and employment of numerous collaborative filtering systems. Virtually all topics that could be of potential interest to users are covered by special-purpose recommendation systems: web pages, news stories, movies, music videos, books, CDs, restaurants, and many more. Some of the best-known representatives of these systems, such as *FireFly* (www.firefly.com) or *WiseWire* (www.wisewire.com) have turned into commercial enterprises. Furthermore, collaborative filtering techniques are becoming increasingly popular as part of online shopping sites. These sites incorporate recommendation systems that suggest products to users based on products that like-minded users have ordered before, or indicated as interesting. For example, users can find out which CD they should order from an online CD store if they provide information about their favorite artists, and several online bookstores (e.g. amazon.com) can associate their available titles with other titles that were ordered by like-minded people.

Although there seems to be an increasingly strong demand for collaborative filtering techniques, only a few different algorithms have been proposed in the literature thus far. Furthermore, the reported algorithms are based on rather simple predictive techniques. Although collaborative filtering can be seen as a classification task, the problem has not received much attention in the machine learning community. It seems likely that predictive performance can be increased through the development of special-purpose algorithms that draw on results from the machine learning literature.

This paper can be outlined as follows. We briefly present the central ideas of previously reported collaborative filtering algorithms. We identify the main shortcomings of these approaches and motivate the need for techniques that do not suffer from these limitations. We then explain how the task of computing collaborative recommendations can be represented as a classification task. Within this framework we present a learning algorithm that addresses the limitations of previous approaches. The proposed method is based on dimensionality reduction through the singular value decomposition (SVD) of an initial matrix of user ratings, exploiting latent structure that essentially eliminates the need for users to rate common items in order to become predictors for one another's preferences. An artificial neural network is used to compute final recommendations. We evaluate our algorithm on a large database of user ratings for motion pictures and show that it significantly outperforms previously proposed algorithms.

2 COLLABORATIVE FILTERING ALGORITHMS

In this section we briefly outline the main ideas of collaborative filtering algorithms reported in the literature. Shardanand and Maes, 1995, discuss a variety of social filtering algorithms and evaluate them in the context of their music recommendation system *Ringo* (predecessor to *FireFly*). These algorithms are based on a simple intuition: predictions for a user should be based on the similarity between the interest profile of that user and those of other users. Therefore, the first step of these algorithms is to compute similarities between user profiles. Suppose we have a database of user ratings for items, where users indicate their interest in an item on a numeric scale. It is now possible to define similarity measures between two user profiles, *U* and *J*, where a user profile simply consists of a vector of numeric ratings. A measure proposed by Shardanand and Maes is the *Pearson correlation coefficient*, r_{UJ}. Once the similarity between profiles has been quantified, it can be used to compute personalized recommendations for users. All users whose similarity is greater than a certain threshold *t* are identified and predictions for an item are computed as the weighted average of the ratings of those similar users for the item, where the weight is the computed similarity. Note that this prediction scheme leads to cases where predictions cannot be computed for all items in the database. If the threshold *t* is set to a high value, only a few very similar users are considered and it becomes increasingly likely that ratings for some specific item are not available. In order to avoid this problem, (Resnick et al., 1994) compute predictions according to the following formula, where U_x is a rating to be predicted for User *U* on item *x* and r_{UJ} is the correlation between users *U* and *J*.

$$U_x = \overline{U} + \frac{\sum_{J \in Raters\ of\ x} (J_x - \overline{J})\, r_{UJ}}{\sum_{J \in Raters\ of\ x} |r_{UJ}|}$$

where

$$r_{UJ} = \frac{\sum (U - \overline{U})(J - \overline{J})}{\sqrt{\sum (U - \overline{U})^2 \cdot \sum (J - \overline{J})^2}}$$

If no ratings for item *x* are available, the prediction is equivalent to the mean of all ratings from user *U*. Similar algorithms were reported and evaluated in (Hill et al. 1995).

While these correlation-based prediction schemes were shown to perform well, they suffer from several limitations. Here, we identify three specific problems: First, correlation between two user profiles can only be computed based on items that both users have rated, i.e. the summations and averages in the correlation formula are only computed over those items that both users have rated. If users can choose among thousands of items to rate, it is likely that overlap of rated items between two users will be small in many cases. Therefore, many of the computed correlation coefficients are based on just a few observations, and thus the computed correlation cannot be regarded as a reliable measure of similarity. For example, a correlation coefficient based on three observations has as much influence on the final prediction as a coefficient based on 30 observations. Second, the correlation approach induces one global model of similarities between users, rather than separate models for classes of ratings (e.g. positive rating vs. negative rating). Current approaches measure whether two user profiles are positively correlated, not correlated at all or negatively correlated. However, ratings given by one user can still be good predictors for ratings of another user, even if the two user profiles are not correlated. Consider the case where user A's positive ratings are a perfect predictor for a negative rating from user B. However, user A's negative ratings do not imply a positive rating from user B, i.e. the correlation between the two profiles could be close to zero, and thus potentially useful information is lost. Third, and maybe most importantly, two users can only be similar if there is overlap among the rated items, i.e. if users did not rate any common items, their user profiles cannot be correlated. Due to the enormous number of items available to rate in many domains, this seems to be a serious stumbling block for many filtering services, especially during the startup phase. However, just knowing that users did not rate the same items does not necessarily mean that they are not like-minded. Consider the following example: Users A and B are highly correlated, as are users B and C. This relationship provides information about the similarity between users A and C as well. However, in case users A and C did not rate any common items, a correlation-based similarity measure could not detect any relation between the two users. We believe that potentially useful information is lost if this kind of transitive similarity relation cannot be detected.

3 COLLABORATIVE FILTERING AS A CLASSIFICATION PROBLEM

In this section we present collaborative filtering in a machine learning framework and suggest the use of an algorithm that specifically addresses the aforementioned limitations of correlation-based approaches.

Collaborative filtering can be seen as a classification task. Based on a set of ratings from users for items, we are trying to induce a model for each user that allows us to classify unseen items into two or more classes, for example *like* and *dislike*. Alternatively, if our goal is to predict user ratings on a continuous scale, we have to solve a regression problem.

Our initial data exists in the form of a sparse matrix, where rows correspond to users, columns correspond to items and the matrix entries are ratings. Note that *sparse* in this context means that most elements of the matrix are empty, because every user typically rates only a very small subset of all possible items. The prediction task can now be seen as filling in the missing matrix values. Since we are interested in learning personalized models for each user, we associate one classifier (or regression model) with every user. This model can be used to predict the missing values for one row in our matrix.

Table 1: Exemplary User Ratings

	I_1	I_2	I_3	I_4	I_5
U_1	*4*		*3*		
U_2		*1*		*2*	
U_3	*3*	*4*	*2*		*4*
U_4	*4*	*2*	*1*		*?*

With respect to Table 1, consider that we would like to predict user 4's rating for item 5. We can train a learning algorithm with the information that we have about user 4's previous ratings. In this example user 4 has provided 3 ratings, which leads to 3 training examples: I_1, I_2, and I_3. These training examples can be directly represented as feature vectors, where users correspond to features (U_1, U_2, U_3) and the matrix entries correspond to feature values. User 4's ratings for I_1, I_2 and I_3 are the class labels of the training examples. However, in this representation we would have to address the problem of many missing feature values. If the learning algorithm to be used cannot handle missing feature values, we can apply a simple transformation. Note that we cannot introduce an additional numeric value that indicates a missing feature, because this would conflate the new value and the observed ratings. However, every user can be represented by up to n Boolean features, where n is the number of points on the scale that is used for ratings. For example, if the full n-point scale of ratings is used to represent ratings from m users, the resulting Boolean features are of the form "User m's rating was i", where $0 < i \leq n$. We can now assign Boolean feature values to all of these new features. If this representation leads to an excessive number of features that only appear rarely throughout the data, the rating scale can be further discretized, e.g. into the two classes *like* and *dislike*. The resulting representation is simple and intuitive: a training example E corresponds to an item that the user has rated, the class label C is the user's discretized rating for that item, and items are represented as vectors of Boolean features F_i.

Table 2: Exemplary Feature Vectors

	E_1	E_2	E_3
U_1like	*1*	*0*	*1*
U_1dislike	*0*	*0*	*0*
U_2like	*0*	*0*	*0*
U_2dislike	*0*	*1*	*0*
U_3like	*1*	*1*	*0*
U_3dislike	*0*	*0*	*1*
Class	*like*	*dislike*	*dislike*

Table 2 shows the resulting Boolean feature vectors (*true = 1* and *false = 0*) for user 4, where a rating of either *1* or *2* corresponds to the class *dislike*, and a rating of either *3* or *4* corresponds to the class *like*.

After converting a data set of user ratings for items into this format, we can draw on the machine learning literature and apply virtually any supervised learning algorithm that, through analysis of a labeled training sample $T = \{E_j, C_j\}$, can induce a function $f : E \rightarrow C$.

However, if we look back at the correlation-based approaches described earlier and express them in our learning framework, we notice that these algorithms solve a classification problem in a somewhat unconventional way. If features and classes are represented as ordinal values (no discretization), these algorithms measure the degree of correlation between features and class labels. Predictions for unseen examples are then computed as a weighted average of feature values. While this approach seems to work reasonably well for the domain at hand, it is not supported by a sound theory that we could use to motivate the algorithms' use for either a classification or regression task. It comes as no surprise that researchers in machine learning have thus far not attempted to solve any task with this algorithm. It seems likely that theoretically well-founded algorithms that have the discrimination between classes as their specific goal, can outperform correlation-based approaches.

3.1 REDUCING DIMENSIONALITY

Our goal is to construct or apply algorithms that address the previously identified limitations of correlation-based approaches. As mentioned earlier, the computation of correlation coefficients can be based on too little information, leading to inaccurate similarity estimates. When applying a learning algorithm, we would like to avoid this problem. In particular, we would like to discard information that we do not consider informative for our classification task. Likewise, we would like to be able to take possible interaction and dependencies among features into account, as we regard this as an essential prerequisite for users to become predictors for one another's preferences even without rating common items. Both of these issues can be addressed through the application of appropriate feature extraction techniques. Furthermore, the need for dimensionality reduction is of particular importance if we represent our data in the proposed learning framework. For large databases containing many users we will end up with thousands of features while our amount of training data is very limited. Learning under these conditions is not practical, because the amount of data points needed to approximate a concept in *d* dimensions grows exponentially with *d*, a phenomenon commonly referred to as the *curse of dimensionality* (Bellman, 1961). This is, of course, not a problem unique to collaborative filtering. Other domains with very similar requirements include the classification of natural language text or, in general, any information retrieval task. In these domains the similarity among text documents needs to be measured. Ideally, two text documents should be similar if they discuss the same subject or contain related information. However, it is often not sufficient to base similarity on the overlap of words. Two documents can very well discuss similar subjects, but use a somewhat different vocabulary. A low number of common words should not imply that the documents are not related. This is very similar to the problem we are facing in collaborative filtering: the fact that two users rated different items should not imply that they are not like-minded. Researchers in information retrieval have proposed different solutions to the text version of this problem. One of these approaches, Latent Semantic Indexing (LSI) (Deerwester et al., 1990) is based on dimensionality reduction of the initial data through singular value decomposition (SVD). We will now show how the SVD can be used as a dimensionality reduction technique for our collaborative filtering task. A more detailed description of underlying algebraic principles can be found in (Berry et al., 1994).

3.2 COLLABORATIVE FILTERING AND THE SVD

We start our analysis based on a rectangular matrix containing Boolean values that indicate user ratings for items (see Table 2). This matrix is typically very sparse, where *sparse* means that most elements are zero, because each item is only rated by a small subset of all users. Furthermore, many features appear infrequently or do not appear at all throughout this matrix. However, features will only affect the SVD if they appear at least twice. Therefore, we apply a first preprocessing step and remove all features that appear less than twice in our training data. The result of this preprocessing step is a matrix *A* containing zeros and ones, with at least two ones in every row. Using the SVD, the initial matrix *A* with *r* rows, *c* columns and rank *m* can be decomposed into the product of three matrices:

$$A = U \Sigma V^T$$

where the columns of *U* and *V* are orthonormal vectors that define the *left* and *right* singular vectors of *A*, and Σ is a diagonal matrix containing corresponding singular values. Since the derived vectors are orthonormal, no vector can be reconstructed as a linear combination of the others. *U* is an $m \times c$ matrix and the singular vectors correspond to columns of the original matrix. *V* is an $r \times m$ matrix and the singular vectors correspond to rows of the original matrix. The singular values quantify the amount of variance in the original data captured by the singular vectors. This representation provides an ideal framework for dimensionality reduction, because one can now quantify the amount of information that is lost if singular values and their corresponding singular vector elements are discarded. The smallest singular values are set to zero, reducing the dimensionality of the new data representation. The underlying intuition is that the *n* largest singular values together with their corresponding singular vector elements capture the important "latent" structure of the initial matrix, whereas random fluctuations are eliminated. The usefulness of the SVD for our task can be further explained by its geometric interpretation. If we choose to retain the *k* largest singular values, we can interpret the singular vectors, scaled by the singular values, as coordinates of points representing the rows and columns of the original matrix in *k* dimensions. In our context, the goal of this transformation is to find a spatial configuration such that items and user ratings are represented by points in *k*-dimensional space, where every item is placed at the centroid of every user rating that it received and every user rating is placed at the centroid of all the items that it was assigned to. While the position of vectors in this *k*-dimensional space is determined through the assignment of ratings to items, items can still be close in this space even without containing any common ratings. Likewise, user ratings can be close to each other, although they were never assigned to a common set of items. Many different strategies for classification of items are theoretically possible using this *k*-dimensional representation. We will now describe the complete algorithm for item classification that we used in our experiments.

3.3 USING SINGULAR VECTORS AS TRAINING EXAMPLES

Our training data is a set of rated items, represented as Boolean feature vectors (see Table 2). We compute the SVD of the training data and discard the *n* smallest singular values, reducing the dimensionality to *k*. Currently, we set *k* to $0.9 \cdot m$, where *m* is the rank of the initial matrix. This value was chosen because it resulted in the best classification performance (evaluated using a tuning set, see Section 4). The singular vectors of matrix *U* scaled by the remaining singular values represent rated items in *k* dimensions. These vectors become our new training examples. Since we compute the SVD of the training data, resulting in real-valued feature vectors of size *k*, we need to specify how we transform examples to be classified into this format. Based on the geometric interpretation of the SVD, the solution to this problem is straightforward. We compute a *k*-dimensional vector for an item, so that with appropriate rescaling of the axes by the singular values, it is placed at the centroid of all the user ratings that it contains. Mathematically, we can compute this vector as:

$$v_k = v^T U_k \Sigma_k^{-1}$$

where *v* is a Boolean feature vector containing user ratings, U_k is a matrix of singular vectors with *k* elements in each vector, and Σ_k is a diagonal matrix containing the *k* largest singular values.

At this point we need to pick a suitable learning algorithm that takes real-valued feature vectors as its input and learns a function that either predicts class membership or computes a score a user would assign to an item. Ideally, we would like to use a learning paradigm that allows for maximum flexibility in evaluating this task as either a regression or classification problem. Therefore, we selected artificial neural networks as the method of choice for our purposes (Rumelhart and McLelland, 1986). It can be shown that neural networks with linear output units and a single hidden layer can approximate any continuous function *f* by increasing the size of the hidden layer (Ripley, 1996). This allows us to solve a regression problem. Alternatively, if we replace the linear output units by logistic units, we can use the same framework to perform logistic regression, or learn to discriminate between classes. We ran various experiments on a tuning set of the data available to us, to determine a network topology and learning paradigm that resulted in good performance (see Section 4 for details on the experimental evaluation). The winning approach was a feed-forward neural network with *k* input units, 2 hidden units and 1 output unit. The hidden units use sigmoid functions, while the output unit is linear. Weights are learned with backpropagation. Although the task at hand might suggest using a user's rating as the function value to predict, we found that a slightly different approach resulted in better performance. We determined the average rating for an item[1] and trained the network on the difference between a user's rating and the average rating. This function appeared to be easier to learn, presumably because the function values take on extreme values less frequently and in these cases express a user's individual taste. In order to predict scores for items, the output of the network needs to be added to the mean of the item. We then used a threshold *t* (depending on the rating scale of the domain, see next section) to convert the predicted rating to a binary class label. In summary, our algorithm for collaborative filter induction proceeds in the following steps:

Training:

- Convert the training data, a sparse matrix of user ratings, to Boolean feature vectors, resulting in a matrix filled with zeros (*false*) and ones (*true*).
- Compute the SVD of the training data.
- Select *k*, the number of dimensions to retain, and reduce the extracted singular vectors accordingly.
- Train a neural network with singular vectors scaled by singular values.

[1] The average is computed using ratings from all users who rated the item, except the user whose rating is to be predicted.

Predicting:

- Convert the item's user ratings to a Boolean feature vector.
- Scale the feature vector into the k-dimensional space.
- Feed the resulting real-valued vector to the trained neural network to compute a prediction.

4 EXPERIMENTAL EVALUATION

In this section we report results of the experimental evaluation of our proposed algorithm. We describe the data set used, the experimental methodology, as well as performance measures we consider appropriate for this task.

4.1 THE EACHMOVIE DATABASE

We ran experiments using data from the *EachMovie* collaborative filtering service. The *EachMovie* service was part of a research project at the Systems Research Center of Digital Equipment Corporation. The service was available for a period of 18 months and was shut down in September 1997. During that time the database grew to a fairly large size, containing ratings from 72,916 users on 1,628 movies. User ratings were recorded on a numeric six-point scale (0.0, 0.2, 0.4, 0.6, 0.8, 1.0). The data set is publicly available and can be obtained from Digital Equipment Corporation (McJones, 1997).

Although data from 72,916 users is available, we restrict our analysis to the first 2,000 users in the database. These 2,000 users provided ratings for 1,410 different movies. We restricted the number of users considered, because we are interested in the performance of the algorithm under conditions where the ratio of users to items is low. This is a situation that every collaborative filtering service has to go through in its startup-phase, and in many domains we cannot expect to have significantly more users than items. We also believe that the deficiencies of correlation-based approaches will be more noticeable under these conditions, because it is less likely to find users with considerable overlap of rated items.

4.2 PERFORMANCE MEASURES

We are most interested in a system that can accurately distinguish between movies a user would like and all other movies rather than a method that accurately predicts the numeric rating of every movie. Of course, a method that predicts the actual ratings most exactly could also be the best classifier for this classification task. To analyze this, we defined two classes, *hot* and *cold,* that were used to label movies. When transforming movies to training examples for a particular user, we label movies as *hot* if the rating for the movie was *0.8* or *1.0*, or *cold* otherwise. We decided to use this threshold since we are interested in identifying movies the user would like and feel strongly about. Since the correlation-based approaches as well as the neural network predict numeric ratings, we base the classification of movies on this numeric prediction, and classify them as *hot* if the predicted rating exceeds the threshold *0.7* (midpoint between the two possible user ratings *0.6* and *0.8*). At the same time, we can still use the predicted score to rank-order classified movies. Not only does assigning class labels allow us to measure classification accuracy, we can also apply additional performance measures, such as *precision* and *recall*, commonly used for information retrieval tasks. In our domain, *precision* is the percentage of movies classified as *hot* that are *hot*, and *recall* is the percentage of *hot* movies that were classified as *hot*. We believe that these measures are appropriate for our study, because we would like to quantify performance for a task that has the identification of relevant items as its goal.

It is important to evaluate *precision* and *recall* in conjunction, because it is easy to optimize either one separately. However, for a classifier to be useful for our purposes we demand that it be precise as well as have high recall. In order to quantify this with a single measure, (Lewis and Gale, 1994) proposed the *F-measure,* a weighted combination of precision and recall that produces scores ranging from 0 to 1. Here we assign equal importance to precision and recall:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

In summary, we measure the overall performance of the algorithms using classification accuracy and the F-measure. Since we see the F-measure as a useful construct to compare classifiers, but think that it is not an intuitive measure to indicate a user's perception of the usefulness of an actual system, we use an additional measure: precision at the top *n* ranked items (here, we report scores for *n = 3* and *n = 10*).

4.3 EXPERIMENTAL METHODOLOGY

Since we are interested in the performance of the algorithms with respect to the number of ratings provided by users, we report learning curves where we vary the number of rated items from 10 to 50. For each user we ran a total of 30 paired trials for each algorithm. For an individual trial of an experiment, we randomly selected 50 rated items to use as a training set, and 30 as a test set. We then started training with 10 examples out of the set of 50 and increased the training set incrementally in steps of 10, measuring the algorithms' performance on the test set for each training set size. Final results for one user are then averaged over all trials. We repeated this for 20 users and

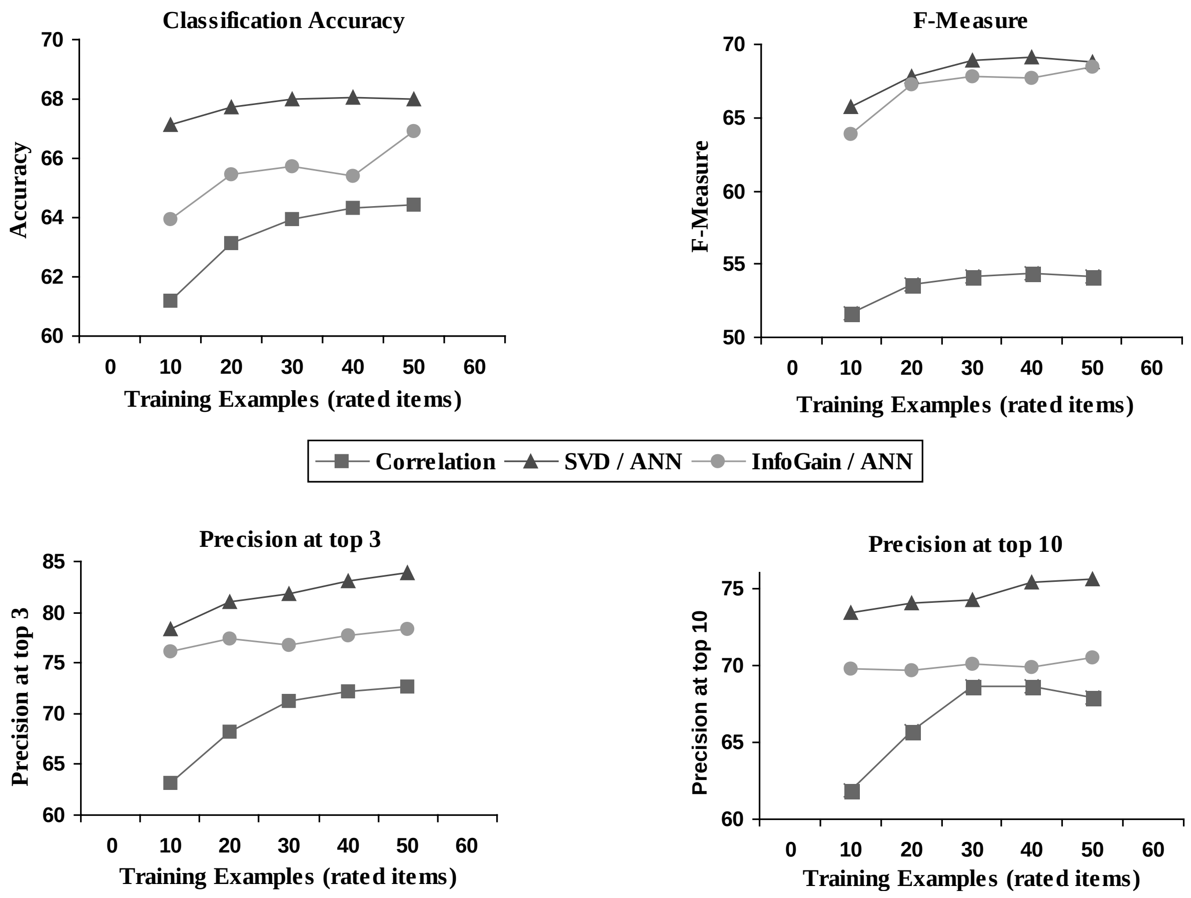

Figure 1: Learning Curves

the final curves reported here are averaged over those 20 users.

The actual size of the feature vectors used to train the neural network depends on the number of rated items in the current training set, as well as the particular rated items. Initially, every training example consists of 4000 Boolean values (2000 users * 2 features per user). Deleting all features that appear less than twice reduces the number of features approximately by a factor of 4 (see section 3.2), i.e. if we start to train our algorithm with 10 examples, we have an initial 10×1000 matrix of training data. After decomposing this matrix using the SVD, the matrix U that represents rated items in a space of lower dimensions is a 10 x 10 matrix (because the initial matrix has 10 columns and this is also the rank of the matrix). Since we keep only 90% of the singular values, the resulting feature vectors consist of 9 real values. Likewise, if we have 50 examples in the training set, the resulting size of every training example after dimensionality reduction is 45.

We determined parameters for our algorithms using a tuning set of 20 randomly selected users. The results reported here are averaged over 20 different users. The training data for these users is based on ratings from the first 2000 users of the database, as described earlier. We selected users randomly, but with the following constraints. First, only users whose prior probability of liking a movie is below 0.75 are considered. Otherwise, scores that indicate high precision of our algorithms might be biased by the fact that there are some users in the database who either like everything or just gave ratings for movies they liked. Second, only users that rated at least 80 movies were selected, so that we could use the same number of training and test examples for all users.

4.4 SUMMARY OF RESULTS

Figure 1 summarizes the performance of three different algorithms. The algorithm labeled *Correlation* is the correlation-based approach that performed best on this data out of the strategies described in Section 2. This approach uses the prediction formula as described in (Resnick et al 1994) and summarized in Section 2. We consider all correlations, i.e. we do not require correlations to be above a certain threshold. The algorithm labeled *SVD/ANN* is our dimensionality reduction approach coupled with a neural network as described in Section 3.3. Since this algorithm is a combination of a feature extraction technique (SVD) and a learning algorithm (ANN), the observed performance does not allow us to infer anything about the relative importance of each technique individually. Therefore, we report the performance of a third algorithm, labeled *Info-*

Gain/ANN, in order to quantify the importance of our proposed feature extraction technique. *InfoGain/ANN* uses the same neural network setup as *SVD/ANN*, but applies a different feature selection algorithm. Here, we compute the expected information gain (Quinlan, 1986) of all the initial features and then select the *n* most informative features, where *n* is equivalent to the number of features used by *SVD/ANN* for each training set size. Since expected information gain cannot detect interaction and dependencies among features, the difference between *SVD/ANN* and *InfoGain/ANN* allows us to quantify the utility of the SVD for this task.

The results show that both *SVD/ANN*, as well as *InfoGain/ANN*, performed better than the correlation approach. In addition, *SVD/ANN* is more accurate and substantially more precise than *InfoGain/ANN*. At 50 training examples *Correlation* reaches a classification accuracy of 64.4%, vs. 67.9% for *SVD/ANN*. While predictive accuracy below 70% might initially seem disappointing, we need to keep in mind that our goal is not the perfect classification of all movies. We would like to have a system that identifies many interesting items and does this with high precision. This ability is measured by the F-measure and we can see that *SVD/ANN* has a significant advantage over the correlation approach (at 50 examples 54.2% for *Correlation* vs. 68.8% for *SVD/ANN*). Finally, if we restrict our analysis to the top 3 or top 10 suggestions of each algorithm, we can see that *SVD/ANN* is much more precise than the other two algorithms. At 50 training examples *Correlation* reaches a precision of 72.6% at the top 3 suggestions, *InfoGain/ANN's* precision is 78.3% and *SVD/ANN* reaches 83.9%. These results are encouraging and provide empirical evidence that the use of theoretically well-founded learning algorithms can lead to improved predictive performance on collaborative filtering tasks. Furthermore, we have shown that an additional performance increase can be obtained through the use of appropriate dimensionality reduction techniques, such as the SVD.

5 DISCUSSION AND FUTURE WORK

Our experiments illustrate the potential of dimensionality reduction techniques that exploit the underlying "latent structure" of user ratings. The key to success of this method is that it can utilize information from users whose ratings are not correlated, or who have not even rated anything in common. However, since we are computing the SVD of the training data, i.e. a matrix consisting only of feature vectors for all items a user has rated, we might not be exploiting the full potential of the method. Including feature vectors of items that the user has not rated in the matrix to decompose will affect the position of the singular vectors corresponding to labeled training examples in k-dimensional space. Future experiments will reveal if further performance improvements can be achieved through the addition of unlabeled training data.

We believe that additional knowledge about the similarity of users and items can be gained through the analysis of textual descriptions of items. Our long-term goal of this work is to combine collaborative and content-based filtering techniques. Similarity between users could then be influenced by similarity between descriptions of rated items. This is a very desirable characteristic, as it would further reduce the need for ratings of common items. We believe that content-based techniques will fit nicely into the learning framework presented in this paper. Since items correspond to feature vectors, one could extend these feature vectors to contain content-based features. We started to run initial experiments using textual descriptions of movies, extending feature vectors with Boolean features indicating the presence or absence of words. These experiments have not yet led to significant performance improvements. However, we assume that the reason for this is the form of textual movie descriptions available to us for these first experiments, rather than the viability of the method itself.

While the proposed SVD/ANN approach leads to performance gains, it is significantly more computationally expensive than the other approaches discussed here. The SVD implementation used in our experiments is a single-vector Lanczos method which is part of the publicly available software package *SVDPACKC* (Berry, 1992). Its computational complexity is *O(3Dz)*, where *z* is the number of non-zero elements in the matrix and *D* is the number of dimensions to be computed. In our experiments we observed training times (SVD + network training) ranging from 0.4 seconds for 10 training examples to 2.3 seconds for 50 training examples[2]. While these times would allow for the application of the algorithm as part of an intelligent information agent operating under real-time conditions, we need to keep in mind that we restricted our experiments to 2000 users. Including more users leads to larger matrices to be decomposed and the algorithm will slow down. Therefore, it remains to be seen if similar techniques could be applied to collaborative-filtering services that have accumulated large amounts of data and need to compute predictions under real-time conditions. However, note that the SVD would not have to be recomputed for each user. The SVD of large portions of the available data could be precomputed, and new items that were not part of this analysis could be scaled into the k-dimensional space as described in Section 3.3. The viability, performance and complexity of this approach will be the subject of future research.

[2] Measured on a 200Mhz Pentium Pro system.

6 SUMMARY AND CONCLUSIONS

In this paper we have identified the shortcomings of correlation-based collaborative filtering techniques and shown how these problems can be addressed through the application of classification algorithms. We believe that the contributions of this paper are twofold. First, we have presented a representation for collaborative filtering tasks that allows the use of virtually any machine learning algorithm. We hope that this will pave the way for further analysis of the suitability of learning algorithms for this task. Second, we have shown that exploiting latent structure in matrices of user ratings can lead to improved predictive performance. In a set of experiments with a database of ratings for motion pictures, we used the singular value decomposition to project user ratings and rated items into a lower dimensional space. This allows users to become predictors for one another's preferences even without any overlap of rated items. Since our society is already being characterized as an information society that suffers from steadily increasing information overload, we regard the automated induction of personalized information filters as an important research problem. The Internet opens up new possibilities to collect enormous amounts of information about users' likes and dislikes. We hope this paper will help develop new ideas for more effective use of this information.

Acknowledgements

We would like to thank the System Research Center of Digital Equipment Corporation for making the *EachMovie* database available for research.

References

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour.* New Jersey: Princeton University Press.

Berry, M. W. (1992). Large scale singular value computations. *International Journal of Supercomputer Applications*, 6(1), 13-49.

Berry, M. W., Dumais, S. T., and O'Brien, G.W. (1995). "Using linear algebra for intelligent information retrieval." *SIAM Review, 37(4)*, 1995, 573-595.

Deerwester, D., Dumais, S. T., Landauer, T. K., Furnas, G.W., and Harshman, R.A. (1990). "Indexing by latent semantic analysis." *Journal of the Society for Information Science, 41(6)*, 391-407.

Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and Evaluating Choices in a Virtual Community of Use. In Proceedings of the Conference on Human Factors in Computing Systems (CHI95), 194-201, Denver, CO, ACM Press.

Lewis, D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, 3-12, London, Springer-Verlag.

McJones, P. (1997). EachMovie collaborative filtering data set. DEC Systems Research Center. http://www.research.digital.com/SRC/eachmovie/.

Pazzani M., and Billsus, D. (1997). Learning and Revising User Profiles: The identification of interesting web sites. *Machine Learning 27*, 313-331.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Resnick, P., Neophytos, I., Mitesh, S. Bergstrom, P. and Riedl, J. (1994) GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of CSCW94: Conference on Computer Supported Cooperative Work, 175-186, Chapel Hill, Addison-Wesley.

Rumelhart, D. E. and McClelland, J. L. (eds) (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.* Cambridge, MA: The MIT Press.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge: Cambridge University Press.

Shardanand, U. and Maes, P. Social Information Filtering: Algorithms for Automating 'Word of Mouth', In Proceedings of the Conference on Human Factors in Computing Systems (CHI95), 210-217, Denver, CO, ACM Press.