Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-043

Application of Dimensionality Reduction in Recommender System - A Case Study

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl

July 14, 2000

Application of Dimensionality Reduction in Recommender System -- A Case Study

Badrul M. Sarwar, George Karypis, Joseph A. Konstan, John T. Riedl

GroupLens Research Group / Army HPC Research Center
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
+1 612 625-4002
{sarwar, karypis, konstan, riedl}@cs.umn.edu

Abstract

We investigate the use of dimensionality reduction to improve performance for a new class of data analysis software called "recommender systems". Recommender systems apply knowledge discovery techniques to the problem of making product recommendations during a live customer interaction. These systems are achieving widespread success in E-commerce nowadays, especially with the advent of the Internet. The tremendous growth of customers and products poses three key challenges for recommender systems in the E-commerce domain. These are: producing high quality recommendations, performing many recommendations per second for millions of customers and products, and achieving high coverage in the face of data sparsity. One successful recommender system technology is *collaborative filtering*, which works by matching customer preferences to other customers in making recommendations. Collaborative filtering has been shown to produce high quality recommendations, but the performance degrades with the number of customers and products. New recommender system technologies are needed that can quickly produce high quality recommendations, even for very large-scale problems.

This paper presents two different experiments where we have explored one technology called *Singular Value Decomposition (SVD)* to reduce the dimensionality of recommender system databases. Each experiment compares the quality of a recommender system using SVD with the quality of a recommender system using collaborative filtering. The first experiment compares the effectiveness of the two recommender systems at predicting consumer preferences based on a database of explicit ratings of products. The second experiment compares the effectiveness of the two recommender systems at producing *Top-N* lists based on a real-life customer purchase database from an E-Commerce site. Our experience suggests that SVD has the potential to meet many of the challenges of recommender systems, under certain conditions.

1 Introduction

Recommender systems have evolved in the extremely interactive environment of the Web. They apply data analysis techniques to the problem of helping customers find which products they would like to purchase at E-Commerce sites. For instance, a recommender system on Amazon.com (www.amazon.com) suggests books to customers based on other books the customers have told Amazon they like. Another recommender system on CDnow (www.cdnow.com) helps customers choose CDs to purchase as gifts, based on other CDs the recipient has liked in the past. In a sense, recommender systems are an application of a particular type of Knowledge Discovery in Databases (KDD) (Fayyad et al. 1996) technique. KDD systems use many subtle data analysis techniques to achieve two unsubtle goals. They are: *i)* to save money by discovering the potential for efficiencies, or *ii)* to make more money by discovering ways to sell more products to customers. For instance, companies are using KDD to discover which products sell well at which times of year, so they can manage their retail store inventory more efficiently, potentially saving millions of dollars a year (Brachman et al. 1996). Other companies are using KDD to discover which customers will be most interested in a special offer, reducing the costs of direct mail or outbound telephone campaigns by hundreds of thousands of dollars a year (Bhattacharyya 1998, Ling et al. 1998). These applications typically involve using KDD to discover a new model, and having an analyst apply the model to the application. However, the most direct benefit of KDD to businesses is increasing sales of existing products by matching customers to the products they will be most likely to purchase. The Web presents

new opportunities for KDD, but challenges KDD systems to perform interactively. While a customer is at the E-Commerce site, the recommender system must learn from the customer's behavior, develop a model of that behavior, and apply that model to recommend products to the customer. Recommender systems directly realize this benefit of KDD systems in E-Commerce. They help consumers find the products they wish to buy at the E-Commerce site. *Collaborative filtering* is the most successful recommender system technology to date, and is used in many of the most successful recommender systems on the Web, including those at Amazon.com and CDnow.com.

The earliest implementations of collaborative filtering, in systems such as Tapestry (Goldberg et al., 1992), relied on the opinions of people from a close-knit community, such as an office workgroup. However, collaborative filtering for large communities cannot depend on each person knowing the others. Several systems use statistical techniques to provide personal recommendations of documents by finding a group of other users, known as *neighbors* that have a history of agreeing with the target user. Usually, neighborhoods are formed by applying proximity measures such as the Pearson correlation between the opinions of the users. These are called *nearest-neighbor techniques.* Figure 1 depicts the neighborhood formation using a nearest-neighbor technique in a very simple two dimensional space. Notice that each user's neighborhood is those other users who are most similar to him, as identified by the proximity measure. Neighborhoods need not be symmetric. Each user has the best neighborhood for him. Once a neighborhood of users is found, particular products can be evaluated by forming a weighted composite of the neighbors' opinions of that document.

These statistical approaches, known as *automated collaborative filtering*, typically rely upon *ratings* as numerical expressions of user preference. Several ratings-based automated collaborative filtering systems have been developed. The GroupLens Research system (Resnick et al. 1994) provides a pseudonymous collaborative filtering solution for Usenet news and movies. *Ringo* (Shardanand et al. 1995) and *Video Recommender* (Hill et al. 1995) are email and web systems that generate recommendations on music and movies respectively. Here we present the schematic diagram of the architecture of the GroupLens Research collaborative filtering engine in figure 2. The user interacts with a

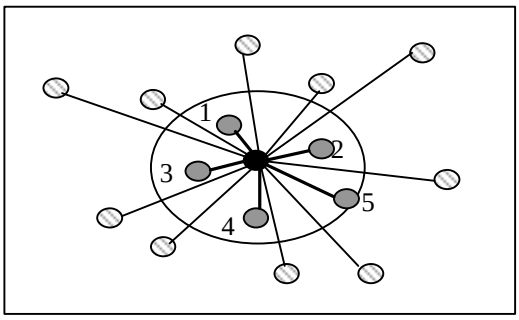

Figure 1: Illustration of the neighborhood formation process. The distance between the target user and every other user is computed and the closest-k users are chosen as the neighbors (for this diagram k = 5).

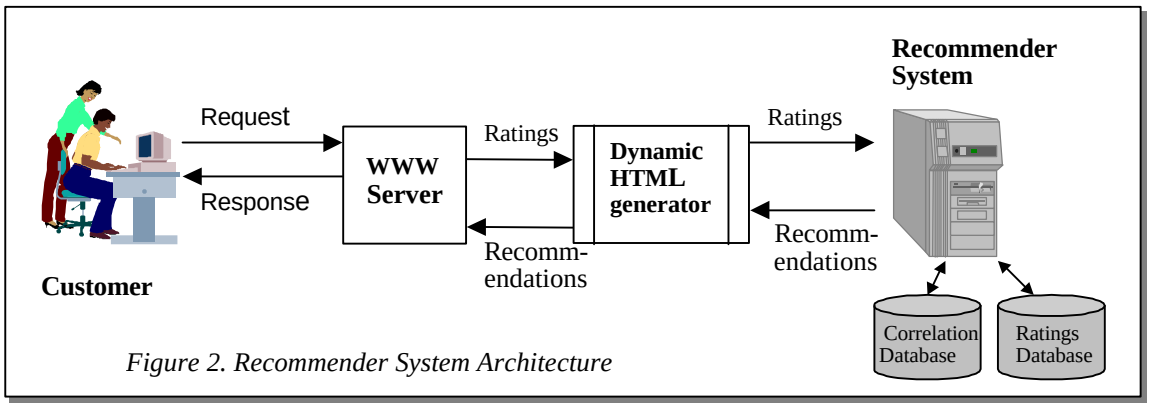

Figure 2. Recommender System Architecture

Web interface. The Web server software communicates with the recommender system to choose products to suggest to the user. The recommender system, in this case a collaborative filtering system, uses its database of ratings of products to form neighborhoods and make recommendations. The Web server software displays the recommended products to the user.

The largest Web sites operate at a scale that stresses the direct implementation of collaborative filtering. Model-based techniques (Fayyad et al., 1996) have the potential to contribute to recommender systems that can operate at the scale of these sites. However, these techniques must be adapted to the real-time needs of the Web, and they must be tested in realistic problems derived from Web access patterns. The present paper describes our experimental results in applying a model-based technique, Latent Semantic Indexing (LSI), that uses a dimensionality reduction technique, Singular Value Decomposition (SVD), to our recommender system. We use two data sets in our experiments to test the performance of the model-based technique: a movie dataset and an e-commerce dataset.

The contributions of this paper are:

1. The details of how one model-based technology, LSI/SVD, was applied to reduce dimensionality in recommender systems for generating predictions.

2. Using low dimensional representation to compute neighborhood for generating recommendations.

3. The results of our experiments with LSI/SVD on two test data sets—our MovieLens test-bed and customer-product purchase data from a large E-commerce company, which has asked to remain anonymous.

The rest of the paper is organized as follows. The next section describes some potential problems associated with correlation-based collaborative filtering models. Section 3 explores the possibilities of leveraging the latent semantic relationship in customer-product matrix as a basis for prediction generation. At the same time it explains how we can take the advantage of reduced dimensionality to form better neighborhood of customers. The section following that delineates our experimental test-bed, experimental design, results and discussion about the improvement in quality and performance. Section 5 concludes the paper and provides directions for future research.

2 Existing Recommender Systems Approaches and their Limitations

Most collaborative filtering based recommender systems build a neighborhood of likeminded customers. The Neighborhood formation scheme usually uses Pearson correlation or cosine similarity as a measure of proximity (Shardanand et al. 1995, Resnick et al. 1994). Once these systems determine the proximity neighborhood they produce two types of recommendations.

1. *Prediction* of how much a customer *C* will like a product *P*. In case of correlation based algorithm, prediction on product '*P*' for customer '*C*' is computed by computing a weighted sum of co-rated items between *C* and all his neighbors and then by adding *C's* average rating to that. This can be expressed by the following formula (Resnick et al., 1994):

$$C_{P_{pred}} = \overline{C} + \frac{\sum_{J \in rates} (J_P - \overline{J}) r_{CJ}}{\sum_J |r_{CJ}|}$$

   Here, r_{CJ} denotes the correlation between user *C* and neighbor *J*. J_P is *J's* ratings on product *P*. $\overline{J}$ and $\overline{C}$ are *J* and *C's* average ratings. The prediction is personalized for the customer *C*. There are, however, some naive *non-personalized prediction* schemes where prediction, for example, is computed simply by taking the average ratings of items being predicted over all users (Herlocker et al., 1999).

2. *Recommendation* of a list of products for a customer *C*. This is commonly known as *top-N* recommendation. Once a neighborhood is formed, the recommender system algorithm focuses on the products rated by the neighbors and selects a list of *N* products that will be liked by the customer.

These systems have been successful in several domains, but the algorithm is reported to have shown some limitations, such as:

- *Sparsity*: Nearest neighbor algorithms rely upon exact matches that cause the algorithms to sacrifice recommender system coverage and accuracy (Konstan et al., 1997. Sarwar et al., 1998). In particular, since the correlation coefficient is only defined between customers who have rated at least two products in common,

many pairs of customers have no correlation at all (Billsus et al., 1998). In practice, many commercial recommender systems are used to evaluate large product sets (e.g., Amazon.com recommends books and CDnow recommends music albums). In these systems, even active customers may have rated well under 1% of the products (1% of 2 million books is 20,000 books--a large set on which to have an opinion). Accordingly, Pearson nearest neighbor algorithms may be unable to make many product recommendations for a particular user. This problem is known as *reduced coverage*, and is due to sparse ratings of neighbors. Furthermore, the accuracy of recommendations may be poor because fairly little ratings data can be included. An example of a missed opportunity for quality is the loss of neighbor transitivity. If customers Paul and Sue correlate highly, and Sue also correlates highly with Mike, it is not necessarily true that Paul and Mike will correlate. They may have too few ratings in common or may even show a negative correlation due to a small number of unusual ratings in common.

- *Scalability*: Nearest neighbor algorithms require computation that grows with both the number of customers and the number of products. With millions of customers and products, a typical web-based recommender system running existing algorithms will suffer serious scalability problems.
- *Synonymy:* In real life scenario, different product names can refer to the similar objects. Correlation based recommender systems can't find this latent association and treat these products differently. For example, let us consider two customers one of them rates 10 different *recycled letter pad* products as "high" and another customer rates 10 different *recycled memo pad* products "high". Correlation based recommender systems would see no match between product sets to compute correlation and would be unable to discover the latent association that both of them like *recycled office products.*

3 Applying SVD for Collaborative Filtering

The weakness of Pearson nearest neighbor for large, sparse databases led us to explore alternative recommender system algorithms. Our first approach attempted to bridge the sparsity by incorporating semi-intelligent filtering agents into the system (Sarwar et al., 1998, Good et al., 1999). These agents evaluated and rated each product, using syntactic features. By providing a dense ratings set, they helped alleviate coverage and improved quality. The filtering agent solution, however, did not address the fundamental problem of poor relationships among like-minded but sparse-rating customers. We recognized that the KDD research community had extensive experience learning from sparse databases. After reviewing several KDD techniques, we decided to try applying Latent Semantic Indexing (LSI) to reduce the dimensionality of our customer-product ratings matrix.

LSI is a dimensionality reduction technique that has been widely used in information retrieval (IR) to solve the problems of *synonymy* and *polysemy* (Deerwester et al. 1990). Given a term-document-frequency matrix, LSI is used to construct two matrices of reduced dimensionality. In essence, these matrices represent latent attributes of terms, as reflected by their occurrence in documents, and of documents, as reflected by the terms that occur within them. We are trying to capture the relationships among pairs of customers based on ratings of products. By reducing the dimensionality of the product space, we can increase density and thereby find more ratings. Discovery of latent relationship from the database may potentially solve the synonymy problem in recommender systems. LSI, which uses singular value decomposition as its underlying matrix factorization algorithm, maps nicely into the collaborative filtering recommender algorithm challenge. Berry et al. (1995) point out that the reduced orthogonal dimensions resulting from SVD are less noisy than the original data and capture the latent associations between the terms and documents. Earlier work (Billsus et al. 1998) took advantage of this semantic property to reduce the dimensionality of feature space. The reduced feature space was used to train a neural network to generate predictions. The rest of this section presents the construction of SVD-based recommender algorithm for the purpose of generating predictions and *top-N recommendations*; the following section describes our experimental setup, evaluation metrics, and results.

3.1 Singular Value Decomposition (SVD)

SVD is a well-known matrix factorization technique that factors an $m \times n$ matrix R into three matrices as the following:

$$R = U \cdot S \cdot V'$$

Where, U and V are two orthogonal matrices of size $m \times r$ and $n \times r$ respectively; r is the rank of the

matrix R. S is a diagonal matrix of size $r \times r$ having all singular values of matrix R as its diagonal entries. All the entries of matrix S are positive and stored in decreasing order of their magnitude. The matrices obtained by performing SVD are particularly useful for our application because of the property that SVD provides the best lower rank approximations of the original matrix R, in terms of Frobenius norm. It is possible to reduce the $r \times r$ matrix S to have only k largest diagonal values to obtain a matrix S_k, $k < r$. If the matrices U and V are reduced accordingly, then the reconstructed matrix $R_k = U_k.S_k.V_k'$ is the closest rank-k matrix to R. In other words, R_k minimizes the Frobenius norm $||R- R_k||$ over all rank-k matrices.

We use SVD in recommender systems to perform two different tasks: First, we use SVD to capture latent relationships between customers and products that allow us to compute the predicted likeliness of a certain product by a customer. Second, we use SVD to produce a *low-dimensional* representation of the original customer-product space and then compute neighborhood in the reduced space. We then used that to generate a list of *top-N* product recommendations for customers. The following is a description of our experiments.

3.1.1 Prediction Generation

We start with a customer-product ratings matrix that is very sparse, we call this matrix R. To capture meaningful latent relationship we first removed sparsity by filling our customer-product ratings matrix. We tried two different approaches: using the average ratings for a customer and using the average ratings for a product. We found the product average produce a better result. We also considered two normalization techniques: conversion of ratings to z-scores and subtraction of customer average from each rating. We found the latter approach to provide better results. After normalization we obtain a filled, normalized matrix R_{norm}. Essentially, $R_{norm} = R+NPR$, where NPR is the fill-in matrix that provides *naive non-personalized recommendation.* We factor the matrix R_{norm} and obtain a *low-rank* approximation after applying the following steps described in (Deerwester et al. 1990):

- factor R_{norm} using SVD to obtain U, S and V.
- reduce the matrix S to dimension k
- compute the square-root of the reduced matrix S_k, to obtain $S_k^{1/2}$
- compute two resultant matrices: $U_kS_k^{1/2}$ and $S_k^{1/2}V_k'$

These resultant matrices can now be used to compute the recommendation score for any customer c and product p. We observe that the dimension of $U_kS_k^{1/2}$ is $m \times k$ and the dimension of $S_k^{1/2}V_k'$ is $k \times n$. To compute the prediction we simply calculate the dot product of the c^{th} row of $U_kS_k^{1/2}$ and the p^{th} column of $S_k^{1/2}V_k'$ and add the customer average back using the following:

$$C_{P_{pred}} = \overline{C} + U_K . \sqrt{S_k}' (c) \cdot \sqrt{S_k} . V_k' (P).$$

Note that even though the R_{norm} matrix is dense, the special structure of the matrix NPR allows us to use sparse SVD algorithms (*e.g.,* Lanczos) whose complexity is almost linear to the number of non-zeros in the original matrix R.

3.1.2 Recommendation generation

In our second experiment, we look into the prospects of using low-dimensional space as a basis for neighborhood formation and using the neighbors' opinions on products they purchased we recommend a list of N products for a given customer. For this purpose we consider customer preference data as binary by treating each non-zero entry of the customer-product matrix as "1". This means that we are only interested in whether a customer consumed a particular product but not how much he/she liked that product.

Neighborhood formation in the reduced space:

The fact that the reduced dimensional representation of the original space is less sparse than its high-dimensional counterpart led us to form the neighborhood in that space. As before, we started with the original customer-product matrix A, and then used SVD to produce three decomposed matrices U, S, and V. We then reduced S by retaining only k eigenvalues and obtained S_k. Accordingly, we performed dimensionality reduction to obtain U_k and V_k. Like the previous method, we finally computed the matrix product $U_kS_k^{1/2}$. This $m \times k$ matrix is the k dimensional representation of m customers. We then performed vector similarity (cosine similarity) to form the neighborhood in that reduced space.

Top-N Recommendation generation:

Once the neighborhood is formed we concentrate on the neighbors of a given customer and analyze the products they purchased to recommend N products the target customer is most likely to purchase. After computing the neighborhood for a given customer C, we scan through the purchase record of each of the k neighbors and perform a frequency count on the

products they purchased. The product list is then sorted and most frequently purchased N items are returned as recommendations for the target customer. We call this scheme *most frequent item recommendation.*

3.1.3 Sensitivity of Number of Dimensions *k*

The optimal choice of the value k is critical to high-quality prediction generation. We are interested in a value of k that is large enough to capture all the important structures in the matrix yet small enough to avoid overfitting errors. We experimentally find a good value of k by trying several different values.

3.1.4 Performance Implications

In practice, e-commerce sites like amazon.com experiences tremendous amount of customer visits per day. Recommending products to these large number of customers in real-time requires the underlying recommendation engine to be highly scalable. Recommendation algorithms usually divide the prediction generation algorithm into two parts: *offline component* and *online component.* Offline component is the portion of the algorithm that requires an enormous amount of computation e.g., the computation of customer-customer correlation in case of correlation-based algorithm. Online component is the portion of the algorithm that is dynamically computed to provide predictions to customers using data from stored offline component. In case of SVD-based recommendation generation, the decomposition of the customer-product matrix and computing the reduced user and item matrices i.e., $U_k S_k^{1/2}$ and $S_k^{1/2} V_k'$ can be done offline.

Offline computation is not very critical to the performance of the recommender system. But there are some issues with the memory and secondary storage requirement that need to be addressed. In case of SVD, the offline component requires more time compared to the correlation-based algorithm. For an $m \times n$ matrix the SVD decomposition requires a time in the order of $O((m+n)^3)$ (Deerwester et. al., 1990). Computation of correlation takes $O(m^2.n)$. In terms of storage, however, SVD is more efficient, we need to store just two reduced customer and product matrices of size $m \times k$ and $k \times n$ respectively, a total of $O(m+n)$, since k is constant. But in case of the correlation-based CF algorithm, an $m \times m$ all-to-all correlation table must be stored requiring $O(m^2)$ storage, which can be substantially large with millions of customers and products.

So, we observe that as a result of dimensionality reduction SVD based online performance is much better than correlation based algorithms. For the same reason, neighborhood formation is also much faster when done in low dimensional space.

4 Experiments

4.1 Experimental Platform

4.1.1 Data sets

As mentioned before we report two different experiments. In the first experiment we used data from our MovieLens recommender system to evaluate the effectiveness of our SVD-based prediction generation algorithm. MovieLens (www.movielens.umn.edu) is a web-based research recommender system that debuted in Fall 1997. Each week hundreds of users visit MovieLens to rate and receive recommendations for movies. The site now has over 35000 users who have expressed opinions on 3000+ different movies.[1] We randomly selected enough users to obtain 100,000 rating-records from the database (we only considered users that had rated twenty or more movies). Rating-record in this context is defined to be a triplet <*customer, product, rating*>. We divided the rating-records into training set and a test set according to different ratios. We call this *training ratio* and denote it by x. A value of x=0.3 indicates that we divide the 100,000 ratings data set into 30,000 train cases and 70,000 test cases. The training data was converted into a user-movie matrix R that had 943 rows (i.e., 943 users) and 1682 columns (i.e., 1682 movies that were rated by at least one of the users). Each entry $r_{i,j}$ represented the rating (from 1 to 5) of the i^{th} user on the j^{th} movie.

The second experiment is designed to test the effectiveness of "neighborhood formed in low dimensional space". In addition to the above movie data, we used historical catalog purchase data from a large e-commerce company. This data set contains purchase information of 6,502 users on 23,554 catalog items. In total, this data set contains 97,045 purchase records. In case of the commerce data set, each record is a triplet <*customer, product, purchase amount*>. Since, purchase amount can't be meaningfully converted to user rating, we didn't use the second data set for prediction experiment. We

[1] In addition to MovieLens' users, the system includes over two million ratings from more than 45,000 EachMovie users. The EachMovie data is based on a static collection made available for research by Digital Equipment Corporation's Systems Research Center.

converted all purchase amounts to "1" to make the data set binary and then used it for recommendation experiment. As before, we divided the data set into a train set and a test set by using similar notion of training ratio, *x*.

4.1.2 Benchmark recommender systems

To compare the performance of SVD-based prediction we also entered the training ratings set into a collaborative filtering recommendation engine that employs the Pearson nearest neighbor algorithm. For this purpose we implemented *CF-Predict*, a flexible recommendation engine that implements collaborative filtering algorithms using C. We tuned CF-Predict to use the best published Pearson nearest neighbor algorithm and configured it to deliver the highest quality prediction without concern for performance (i.e., it considered every possible neighbor to form optimal neighborhoods). To compare the quality of SVD neighborhood-based recommendations, we implemented another recommender system that uses *cosine-similarity* in high dimensional space to form neighborhood and returns *top-N* recommendations, we call it *CF-Recommend*. We used cosine measure for building neighborhood in both cases because in the low dimensional space proximity is measured only by computing the cosine.

For each of the ratings in the test data set, we requested a prediction from CF-Predict and also computed the same prediction from the matrices $U_k S_k^{1/2}$ and $S_k^{1/2} V_k'$ and compared them. Similarly, we compared two *top-N* recommendation algorithms.

4.2 Evaluation Metrics

Recommender systems research has used several types of measures for evaluating the success of a recommender system. We only consider the quality of prediction or recommendation, as we're only interested in the output of a recommender system for the evaluation purpose. It is, however, possible to evaluate intermediate steps (e.g., the quality of neighborhood formation). Here we discuss two types of metrics for evaluating predictions and *top-N* recommendations respectively.

4.2.1 Prediction evaluation metrics

To evaluate an individual item prediction researchers use the following metrics:

- Coverage metrics evaluate the number of products for which the system could provide recommendations. Overall coverage is computed as the percentage of customer-product pairs for which a recommendation can be made.
- Statistical accuracy metrics evaluate the accuracy of a system by comparing the numerical recommendation scores against the actual customer ratings for the customer-product pairs in the test dataset. *Mean Absolute Error (MAE)*, *Root Mean Squared Error (RMSE)* and *Correlation* between ratings and predictions are widely used metrics. Our experience has shown that these metrics typically track each other closely.
- Decision support accuracy metrics evaluate how effective a prediction engine is at helping a user select high-quality products from the set of all products. These metrics assume the prediction process as a binary operation—either products are predicted (good) or not (bad). With this observation, whether a product has a prediction score of 1.5 or 2.5 on a five-point scale is irrelevant if the customer only chooses to consider predictions of 4 or higher. The most commonly used decision support accuracy metrics are *reversal rate*, *weighted errors* and *ROC sensitivity* (Le et al., 1995)

We used MAE as our choice of evaluation metric to report prediction experiments because it is most commonly used and easiest to interpret directly. In our previous experiments (Sarwar et al., 1999) we have seen that MAE and ROC provide the same ordering of different experimental schemes in terms of prediction quality.

4.2.2 Recommendation evaluation metrics

To evaluate *top-N* recommendation we use two metrics widely used in the information retrieval (IR) community namely recall and precision. However, we slightly modify the definition of recall and precision as our experiment is different from standard IR. We divide the products into two sets: the *test* set and *top-N* set. Products that appear in both sets are members of the *hit set.* We now define recall and precision as the following:

- Recall in the context of the recommender system is defined as:

$$Recall = \frac{size\ of\ hit\ set}{size\ of\ test\ set} = \frac{\left|test \bigcap top\ N\right|}{\left|test\right|}$$

- Precision is defined as:

$$Precision = \frac{size\ of\ hit\ set}{size\ of\ topN\ set} = \frac{\left|test \bigcap topN\right|}{N}$$

These two measures are, however, often conflicting in nature. For instance, increasing the number N tends to increase recall but decreases precision. The fact that both are critical for the quality judgement leads us to use a combination of the two. In particular, we use the standard F1 metric (Yang et. al., 1999) that gives equal weight to them both and is computed as follows:

$$F1 = \frac{2 * Recall * Precsion}{(Recall + Precision)}$$

We compute F1 for each individual customer and calculate the average value to use as our metric.

4.3 Experimental Steps

4.3.1 Prediction Experiment.

Each entry in our data matrix R represents a rating on a 1-5 scale, except that in cases where the user i didn't rate movie j the entry $r_{i,j}$ is null. We then performed the following experimental steps.

We computed the average ratings for each user and for each movie and filled the null entries in the matrix by replacing each null entry with the column average for the corresponding column. Then we normalized all entries in the matrix by replacing each entry $r_{i,j}$ with $(r_{i,j} - \underline{r}_i)$, where, $\underline{r}_i$ is the row average of the i^{th} row. Then MATLAB was used to compute the SVD of the filled and normalized matrix R, producing the three SVD component matrices U, S and V'. S is the matrix that contains the singular values of matrix R sorted in decreasing order. S_k was computed from S by retaining only k largest singular values and replacing the rest of the singular with 0. We computed the square root of the reduced matrix and computed the matrix products $U_k S_k^{1/2}$ and $S_k^{1/2} V'_k$ as mentioned above. We then multiplied the matrices $U_k S_k^{1/2}$ and $S_k^{1/2} V'_k$ producing a 943 x 1682 matrix, P. Since the inner product of a row from $U_k S_k^{1/2}$ and a column from $S_k^{1/2} V_k$ gives us a prediction score, each entry p_{ij} of this resultant matrix P holds the prediction score for each user-movie pair i,j. We then de-normalized the matrix entries by adding the user average back into each prediction scores and loaded the training set ratings into *CF-Predict* and request prediction scores on each of the test set ratings. Computed MAE of the SVD and the *CF-Predict* prediction scores and compare the two sets of results.

We repeated the entire process for k = 2, 5-21, 25, 50 and 100, and found 14 to be the most optimum value (Figure 3(a)). We then fixed k at 14 and varied the train/test ratio x from 0.2 to 0.95 with an increment of 0.05 and for each point we run the experiment 10 times each time choosing different training/test sets and take the average to generate the plots. Note that the overall performance of the SVD-based prediction algorithm does significantly change for a wide range of values of k.

4.3.2 Top-N recommendation experiment:

We started with a matrix as the previous experiment but converted the rating entries (i.e., non-zero entries) to "1". Then we produced *top-10* product recommendations for each customer based on the following two schemes:

- High dimensional neighborhood: In this scheme we built the customer neighborhood in the original customer-product space and used *most frequent item* recommendation to produce *top-10* product list. We then used our F1 metric to evaluate the quality.
- Low dimensional neighborhood: We first reduce the dimensionality of the original space by applying SVD and then used $U_k S_k^{1/2}$ (i.e., representation of customers in k dimensional space) matrix to build the neighborhood. As before we used *most frequent item* recommendation to produce *top-10* list and evaluated by using F1 metric.

In this experiment our main focus was on the E-

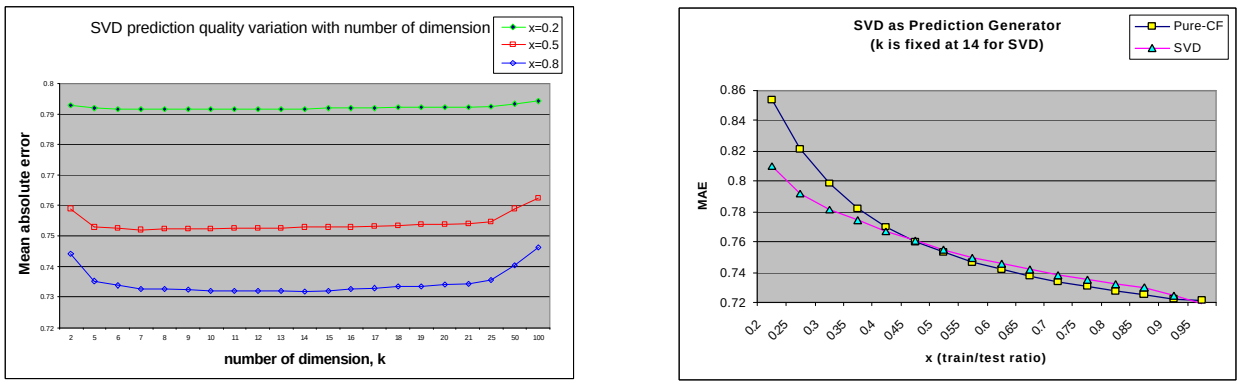

Figure 3. (a) Determination of optimum value of k. (b) SVD vs. CF-Predict prediction quality

commerce data. We also report our findings when we apply this technique on our movie preference data.

4.4 Results

4.4.1Prediction experiment results

Figure 3(b) charts our results for the prediction experiment. The data sets were obtained from the same sample of 100,000 ratings, by varying the sizes of the training and test data sets, (recall that *x* is the ratio between the size of the training set and the size of the entire data set). Note that the different values of *x* were used to determine the sensitivity of the different schemes on the sparsity of the training set.

4.4.2 Top-N recommendation experiment results

For the recommendation experiment, we first determined the optimum *x* ratio for both of our data sets in high dimensional and low dimensional cases. At first we run the high dimensional experiment for different *x* ratio and then we perform low dimensional experiments for different *x* values for a fixed dimension (*k*) and compute the F1 metric. Figure 4 shows our results, we observe that optimum *x* values are 0.8 and 0.6 for the movie data and the E-commerce data respectively.

Once we obtain the best *x* value, we run high dimensional experiment for that *x* and compute F1 metric. Then we run our low-dimensional experiments for that *x* ratio, but vary the number of dimension, *k*. Our results are presented in figures 5 and 6. We represent the corresponding high dimensional results (i.e., results from CF-recommend) in the chart by drawing vertical lines at their corresponding values.

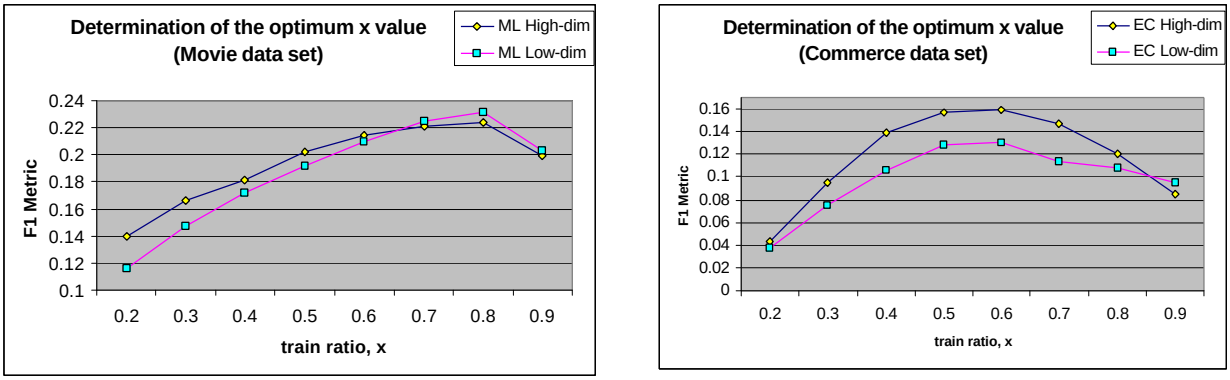

Figure 4. Determination of the optimum value of x. *a) for the Movie data b) for the Commerce data*

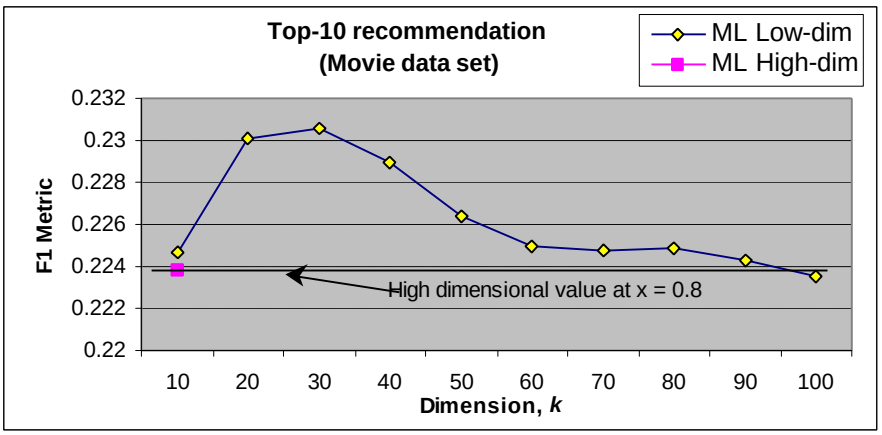

Figure 5. Top-10 recommendation results for the MovieLens data set.

4.5 Discussion

In case of the prediction experiment, we observe that in Figure 3(b) for *x<0.5* SVD-based prediction is better than the CF-Predict predictions. For *x>0.5*, however, the CF-Predict predictions are slightly better. This suggests that nearest-neighbor based collaborative filtering algorithms are susceptible to data sparsity as the neighborhood formation process is hindered by the lack of enough training data. On the other hand, SVD based prediction algorithms can overcome the sparsity problem by utilizing the latent relationships. However, as the training data is increased both SVD and CF-Predict prediction quality improve but the improvement in case of CF-Predict surpasses the SVD improvement.

From the plots of the recommender results (Figures 5 and 6), we observe that for the movie data the best result happens in the vicinity of *k=20* and in case of the e-commerce data the recommendation quality keeps on growing with increasing dimensions. The movie experiment reveals that the low dimensional results are better than the high dimensional counterpart at all values of *k*. In case of the e-commerce experiment the high dimensional result is always better, but as more and more dimensions are added low dimensional values improve. However, we increased the dimension values up to 700, but the low dimensional values were still lower than the high dimensional value. Beyond 700 the entire process becomes computationally very expensive. Since the commerce data is very high dimensional (6502x23554), probably such a small *k* value (up to 700) is not sufficient to provide a useful approximation of the original space. Also another factor to consider is the amount of sparsity in the data sets, the movie data is 95.4% sparse (100,000 nonzero entries in 943x1,682 matrix), while the e-commerce data is 99.996% sparse (97,045 nonzero entries in 6,502x23,554 matrix). To test this hypothesis we deliberately increased sparsity of our movie data (i.e., remove nonzero entries) and repeated the experiment and observed dramatic reduction in F1 values!

Overall, the results are encouraging for the use of SVD in collaborative filtering recommender systems. The SVD algorithms fit well with the collaborative filtering data, and they result in good quality predictions. And SVD has potential to provide better online performance than correlation-based systems. In case of the *top-10* recommendation experiment we have seen even with a small fraction of dimension, i.e., 20 out of 1682 in movie data, SVD-based recommendation quality was better than corresponding high dimensional scheme. It indicates that neighborhoods formed in the reduced dimensional space are better than their high dimensional counterparts.[2]

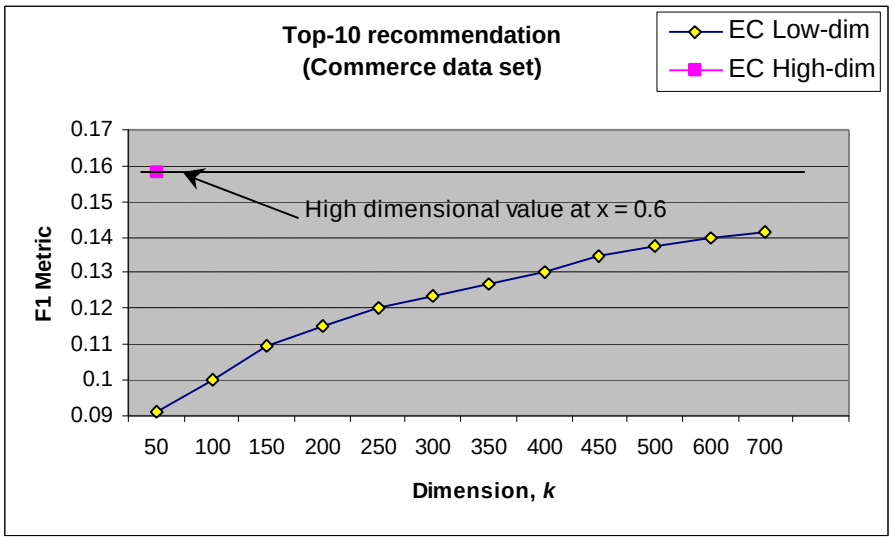

Figure 6. Top-10 recommendation results for the E-Commerce data set.

[2] We're also working with experiments to use the reduced dimensional neighborhood for prediction generation using classical CF algorithm. So far, the results are encouraging.

5 Conclusions

Recommender systems are a powerful new technology for extracting additional value for a business from its customer databases. These systems help customers find products they want to buy from a business. Recommender systems benefit customers by enabling them to find products they like. Conversely, they help the business by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web.

Recommender systems are being stressed by the huge volume of customer data in existing corporate databases, and will be stressed even more by the increasing volume of customer data available on the Web. New technologies are needed that can dramatically improve the scalability of recommender systems.

Our study shows that Singular Value Decomposition (SVD) may be such a technology in some cases. We tried several different approaches to using SVD for generating recommendations and predictions, and discovered one that can dramatically reduce the dimension of the ratings matrix from a collaborative filtering system. The SVD-based approach was consistently worse than traditional collaborative filtering in se of an extremely sparse e-commerce dataset. However, the SVD-based approach produced results that were better than a traditional collaborative filtering algorithm some of the time in the denser MovieLens data set. This technique leads to very fast online performance, requiring just a few simple arithmetic operations for each recommendation. Computing the SVD is expensive, but can be done offline. Further research is needed to understand how often a new SVD must be computed, or whether the same quality can be achieved with incremental SVD algorithms (Berry et. al., 1995).

Future work is required to understand exactly why SVD works well for some recommender applications, and less well for others. Also, there are many other ways in which SVD could be applied to recommender systems problems, including using SVD for neighborhood selection, or using SVD to create low-dimensional visualizations of the ratings space.

6 Acknowledgements

Funding for this research was provided in part by the National Science Foundation under grants IIS 9613960, IIS 9734442, and IIS 9978717 with additional funding by Net Perceptions Inc. This work was also supported by NSF CCR-9972519, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. We thank anonymous reviewers for their valuable comments.

References

1. Berry, M. W., Dumais, S. T., and O'Brian, G. W. 1995. "Using Linear Algebra for Intelligent Information Retrieval". SIAM Review, 37(4), pp. 573-595.
2. Billsus, D., and Pazzani, M. J. 1998. "Learning Collaborative Information Filters". In Proceedings of Recommender Systems Workshop. Tech. Report WS-98-08, AAAI Press.
3. Bhattacharyya, S. 1998. "Direct Marketing Response Models using Genetic Algorithms." In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 144-148.
4. Brachman, R., J., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G., and Simoudis, E. 1996. "Mining Business Databases." *Communications of the ACM*, 39(11), pp. 42-48, November.
5. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. 1990. "Indexing by Latent Semantic Analysis". *Journal of the American Society for Information Science*, 41(6), pp. 391-407.
6. Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., Eds. 1996. "Advances in Knowledge Discovery and Data Mining". *AAAI press/MIT press*.
7. Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. 1992. "Using Collaborative Filtering to Weave an Information Tapestry". *Communications of the ACM*. December.
8. Good, N., Schafer, B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J. 1999. "Combining Collaborative Filtering With Personal Agents for Better Recommendations." In *Proceedings of the AAAI-'99 conference, pp 439-446.*
9. Heckerman, D. 1996. "Bayesian Networks for Knowledge Discovery." In *Advances in Knowledge Discovery and Data Mining*. Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., Eds. *AAAI press/MIT press*.

10. Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. 1999. "An Algorithmic Framework for Performing Collaborative Filtering." In *Proceedings of ACM SIGIR'99. ACM press.*

11. Hill, W., Stead, L., Rosenstein, M., and Furnas, G. 1995. “Recommending and Evaluating Choices in a Virtual Community of Use”. In *Proceedings of CHI ’95.*

12. Le, C. T., and Lindgren, B. R. 1995. “Construction and Comparison of Two Receiver Operating Characteristics Curves Derived from the Same Samples”. *Biom. J.* 37(7), pp. 869-877.

13. Ling, C. X., and Li C. 1998. “Data Mining for Direct Marketing: Problems and Solutions.” In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pp. 73-79.

14. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994. “GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of CSCW ’94,* Chapel Hill, NC.

15. Sarwar, B., M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., and Riedl, J. 1998. “Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System.” In *Proceedings of CSCW ’98,* Seattle, WA.

16. Sarwar, B.M., Konstan, J.A., Borchers, A., and Riedl, J. 1999. "Applying Knowledge from KDD to Recommender Systems." *Technical Report TR 99-013,* Dept. of Computer Science, University of Minnesota.

17. Schafer, J. B., Konstan, J., and Riedl, J. 1999. “Recommender Systems in E-Commerce.” In *Proceedings of ACM E-Commerce 1999 conference*.

18. Shardanand, U., and Maes, P. 1995. “Social Information Filtering: Algorithms for Automating ‘Word of Mouth’.” In *Proceedings of CHI ’95.* Denver, CO.

19. Yang, Y., and Liu, X. 1999. "A Re-examination of Text Categorization Methods." In *Proceedings of ACM SIGIR'99 conferenc,* pp 42-49.

20. Zytkow, J. M. 1997. “Knowledge = Concepts: A Harmful Equation.” In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining.*