The Effects of Singular Value Decomposition

on

Collaborative Filtering

Computer Science Technical Report

Dartmouth College

PCS-TR98-338

Michael H. Pryor

Senior Honors Thesis

June 1998

Abstract

As the information on the web increases exponentially, so do the efforts to automatically filter out useless content and to search for interesting content. Through both explicit and implicit actions, users define where their interests lie. Recent efforts have tried to group similar users together in order to better use this data to provide the best overall filtering capabilities to everyone.

This thesis discusses ways in which linear algebra, specifically the singular value decomposition, can be used to augment these filtering capabilities to provide better user feedback. The goal is to modify the way users are compared with one another, so that we can more efficiently predict similar users. Using data collected from the PhDs.org website, we tested our hypothesis on both explicit web page ratings and implicit visits data.

Introduction

Standard content-based query filtering methods use keywords and word matching to find relevant documents for users. In contrast, collaborative filtering [5], also known as Social Information Filtering [15], attempts to filter objects based on personal taste by grouping similar people together in neighborhoods. By aligning the interests of several users, collaborative filtering methods can choose documents relevant to a group of people rather than a single user. While the filtered documents will not be tailored specifically to the individual's taste, if the user's neighbors are sufficiently similar, no difference will be detected. The advantage gained is that information gathered from other users can be used to filter documents for everyone else in the near neighborhood of similarity. The burden of providing sufficient data to define each user's tastes can then be shifted to the group, allowing a larger set of documents to be filtered with less overall individual feedback. Once the user's neighborhood is established from people with similar tastes, documents can be recommended or thrown away based on the preferences of those located in the neighborhood.

Most collaborative filtering methods find similar users by modeling users in some n-dimensional space and finding the closest neighbors. The user's position in this space is determined by their response to some subset of the documents to be filtered. This subset is usually quite small compared to the total number of documents or objects that can be rated; users must rate a considerable fraction of the collection before enough overlap can be established to reliably find similarities with others. We address this issue by modeling users not by their responses to different documents, but rather by their responses to the set of features that make up those documents. If each rating can be broken down into a linear

combination of responses to those features that make up each document, we can determine the most similar users sooner than with traditional methods.

Related Works

Ringo

The term "Social Information Filtering" was coined at the MIT Media Lab by U. Shardanand to describe a project he developed known as Ringo [15]. Ringo was a music recommendation service which generated user profiles based on explicit feedback on artists through the web and an email gateway. As users built up a profile of artists they liked and disliked, Ringo was able to find other users with similar tastes and to use their ratings as predictions. For example, if Eddie liked the Beatles and Owen liked the Beatles and Pink Floyd, then Ringo may have recommended Pink Floyd to Eddie depending on how similar he was to Owen.

Shardanand experimented with many different metrics for measuring "closeness". In each method, users were represented by a vector consisting of their ratings, and compared using the Mean Squared Difference algorithm and the Pearson correlation algorithm [15]. Shardanand considered any pair of users with a distance less than a given threshold to be close. Predictions were then generated using a weighted average of the predictions made by the user's neighbors.

Ringo gathered a very large user base, and performed rather well in predicting users' tastes in music. Since music is not amenable to parsing by a computer, Social Information Filtering provided the perfect method for predicting and finding relevant music for users. New users were given a set of artists consisting of the most often rated artists and others randomly generated from the artist database, allowing Ringo to immediately establish overlap in comparing different users.

Letizia & Let's Browse

Let's Browse [10] and its predecessor, Letizia [9], are web agents that assist a user during their browsing experience. By watching the user's behavior, Letizia learns the person's interests and browses ahead from the current web page to see how interested the user would be in documents linked to from that site. Users can then choose to follow Letizia's recommendations for visiting the links from the current site. Let's Browse is a modification of Letizia that uses a group of profiles, instead of a single profile, to determine which sites to browse next. Let's Browse is being developed to address situations in which multiple users could be viewing the same web browser at the same time (e.g., WebTV). A receiver on the computer determines which users are in the area of the monitor, and uses their tastes (as recorded in a database) to predict sites which will be most interesting to the entire group.

Letizia develops a profile of the user's interest by watching how she reacts to different web pages. If she stays at one page for a long period of time, that page is assumed to be interesting. Also if the user bookmarks a page, Letizia assumes that link is very important. By using implicit feedback measures, Letizia can find out which documents a user likes, and then look at keywords contained in those documents. Later when the user is browsing, Letizia does a breadth first search from the current page to find documents that contain those keywords, and notifies the user that they might be interested in following a particular link.

GroupLens

GroupLens [8] was developed to use collaborative filtering for helping users find interesting Usenet news articles. The system is implemented using a source server that

holds ratings and predictions for each user. These ratings are propagated to each Usenet newsreader by the GroupLens API.

From the beginning, the creators of GroupLens designed the system to be very easy to use and integrate into existing newsreaders. GroupLens supports a single keystroke rating system, making it simple for users to rate documents. Integrating the GroupLens API into any newsreader takes a minimal amount of effort; the newsreader only needs to be able to converse with a centralized server where all the predictions are computed. In order to deal with rating scarcity, comparisons are made to other readers based only on ratings within each newsgroup. Since most people only read a fraction of the Usenet news, and in particular only pay attention to a few newsgroups, comparisons between users who never read the same newsgroups would hinder the success of the system.

The GroupLens project has also demonstrated that implicit feedback can be as effective as explicit feedback in providing good predictions. By looking at the time a user spends reading each news post, GroupLens is able to predict accurately the interest of each user in the postings. Research regarding implicit feedback methods is extremely important since obtaining a significant amount of data explicitly from users is quite difficult.

PHOAKS

PHOAKS (People Helping One Another Know Stuff) [16] finds informative URLs in Usenet news postings. The system uses implicit feedback mechanisms to determine URLs that will be interesting to readers of specific newsgroups. By scanning through messages and counting the occurrences of the URLs posted, PHOAKS attempts to assemble a list of the most important URLs to readers of that group. Although the system

does not supply user specific information, PHOAKS is able to effectively form a core group of pages that interest a subsection of users.

Fab & Slider

The Fab [1] system is a web based recommendation service that incorporates both collaborative and content-based filtering methods. Users' profiles are constructed as a collection of keywords contained in those documents that each user rates highly. Documents are presented for rating when either the content of the document matches previous documents that were rated highly, or neighboring users rate a document highly. Every time a favorable or unfavorable rating is received, the profile of the user is updated to reflect the new rating.

Collection agents are sent out over the web to look for documents with specific content, each agent using a different set of keywords. After retrieving the documents, they are passed to a central server where a selection agent matched to each user's profile, scours through the documents looking for interesting material. Relevant documents are then presented to the user for rating. This rating dynamically affects the selection agents behavior and changes the user's profile. The rating also affects the collection agent that retrieved the document. Unpopular collection agents are removed and replaced with more successful ones over time.

The Fab system combines the best features of both content-based and collaborative filtering methods and also manages to keep the system dynamically updated to the current users' tastes. One potential shortcoming is Fab's reliance on explicit user feedback. The success with the initial project has led Fab's creators to begin working on a new project

called Slider, which adds the advantage of using implicit feedback methods to determine user profiles.

Siteseer

Siteseer [11] is a collaborative system using web browser bookmarks to find neighbors and recommend sites. Users with significant overlap in bookmark listings are determined to be close to one another, allowing previously unvisited sites to be recommended to one another.

The creators of the simple system recognize its limitations in providing relevant recommendations. Users bookmark pages for many different reasons, and the distinction between low interest in a page and high interest in a page cannot be determined from the bookmarks listing alone. Siteseer's success suggests that bookmarks provide a valuable form of implicit feedback that could be quite useful to other filtering systems.

Tapestry

Tapestry [5] uses collaborative filtering for delivering mail to mailing lists. As messages arrive, users can annotate the messages and later query for similar messages that others were interested in. The Tapestry Query Language provides a fairly complex tool, similar to the database language SQL, for finding messages based on feedback provided by other users. A drawback of Tapestry is that it provides no mechanism for automatically determining nearest neighbors, and it requires users to actively annotate each message they receive in order to be effective.

Information Retrieval

Information retrieval methods provide a starting point in the quest for efficient collaborative filtering. Both information retrieval methods and current collaborative filtering systems find requested documents by matching them to a specific set of criteria. Information retrieval methods determine the criteria mainly by keywords and content. Collaborative filtering completes the filtering process using user profiles, which may consist of keywords, links to similar users, or responses to certain document features such as download time or HTML layout.

Many retrieval systems model documents as vectors in space and model user queries in the same space to find similar documents. Salton and McGill [13] discuss this representation in their paper on the SMART retrieval system, perhaps one of the most famous and earliest information retrieval systems. Distances between documents are measured by the cosine of the angle between the document vectors. When a user enters a query, the terms of the query are modeled in the document space as a pseudo-document. The retrieval system then returns those documents which are closest according to the vector angles.

Dumais, et. al., attempt to use Latent Semantic Analysis to respond to deficiencies in normal vector based methods. A problem known as *synonymy* arises in the case where a user searches for a term such as *TV*. Unless a thesaurus is built for the document collection, documents using only the word *television* will not show up in the result set. Another problem, known as *polysemy*, arises with words such as *mouse*, which may have multiple meanings. A user searching for the word *mouse* in an attempt to purchase a new device for her computer, could find that documents concerning the small furry rodent also were returned for the query. Normal word matching retrieval systems cannot overcome this hurdle. The keywords used to model documents are only a small subset of the total

number of terms that people will use to search for those documents. In addition, when modeling a document as a vector, there will only be one term for the word mouse, regardless of the context of that word, since no automatic methods exist to detect these ambiguities. Naturally, information can be lost in the vector model structure [2].

The information retrieval field uses linear algebra to overcome both of these problems. In particular Latent Semantic Indexing (LSI) [3] is used in discovering the underlying structure and context present in each document. Rather than searching by terms, LSI methods have allowed people to represent documents through the main factors that describe the document, and to find related documents based on those factors. In the next section, I discuss how the singular value decomposition can be used to analyze these factors and use them to estimate the underlying structure present in documents. A significant performance benefit has been discovered in using these methods for information retrieval of up to 30% better than keyword searching methods [2].

Collaborative Filtering

Collaborative filtering methods form profiles of users based on individual user data. In our case, user profiles are based on visits to a defined set of web documents, and if applicable the user's rating of those documents. We consider both the problem of predicting which sites a user will visit as well as how she will rate those sites.

In the Ringo system, Shardanand modeled users as a vector consisting of their ratings to different artists. Similar neighbors were computed using a variety of methods, all of which are similar to the cosine distance algorithm. The Mean Squared Difference algorithm measured the distance between two users as the average squared difference of each users ratings. The Pearson correlation algorithm attempted to find similar users by finding the positive and negative correlation between their ratings.

Shardanand's methods relate directly to Salton's information retrieval methods. Both methods represent objects as a vector of their components and attempt to find similar objects based on those components. In information retrieval, keyword components are quite suitable for matching similar documents. Shardanand demonstrates, through research with Ringo [15], that vector space modeling is also quite suitable for collaborative filtering methods. Since Dumais has shown that LSI methods can enhance standard information retrieval methods [3], we feel that LSI methods will also be able to enhance collaborative filtering methods.

Users can be compared with each other based on how similar their responses to a set of web documents are. A simple method is to represent each user as such:

$$U = (u_1, u_2, u_3, \ldots, u_n)$$

where the user is represented by a vector u_k – the user's rating of document k. The similarity between users, U and V, can then be defined as the distance between the users' vectors, computed as the length of $U - V$.

$$dist(U,V) = \|U - V\| = \sqrt{(u - v) \bullet (u - v)} = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \ldots + (u_n - v_n)^2}$$

In some cases, a more reliable metric, as used by Salton [13], may be the cosine of the angle between the vectors U and V, which ignores the length of the vectors:

$$\cos\Theta = \frac{U \bullet V}{\|U\|\|V\|}$$

Users with similar ratings will be near each other; users with dissimilar ratings will be far from each other. The basic assumption is that if two people give similar ratings to most of the sites they rated, then in the future their ratings will continue to be similar. Provided this assumption holds, one can predict ratings for a user Bob, by using ratings of other users who are sufficiently similar. Ratings given by Bob's neighbors for sites Bob has not visited yet can be taken to be Bob's estimated prediction for those sites.

Ringo used standard cosine analysis methods to determine similarities between users. Ratings of artists were simply modeled as vectors and similarity was then determined in the artist dimensional space, where each rating represented a reaction to a particular artist. Ringo's success prompted us to use similar methods for finding similar users based on web page ratings. Although, the method Ringo uses may not be as effective for web pages as it was for music.

Using standard cosine analysis without modification, as Ringo did, to determine neighbors may pose problems. In measuring the sum of the difference between each user's individual ratings for documents, that method is assumed that each user has defined a rating for every document. If no rating exists, then the difference between the users cannot be

computed. One possible solution to this problem is to use the average rating for documents that have not been rated. In the case where no two users share any ratings, this technique would only reveal the distance each user is from the average user. It is possible though that the users are quite similar but have not rated similar documents. In this case, the distance method will simply be adding the distance they are from the average user, and may decide that the two users are very dissimilar.

If two users are very similar but have not rated the same set of documents, we cannot determine that they are similar using the previously mentioned method. As a simple example, consider a user interested in learning Java. The sites *www.gamelan.com* and *www.sun.com* offer a wealth of information about Java. One would expect that over all ratings, users who have liked Gamelan, will also rate the Sun site highly and that people who are not interested in the Sun site also are not interested in the Gamelan site. Because of their obvious strong correlation in content, both sites will receive similar ratings.

If two users, Tom and Allison, individually rate only one of the two documents, Tom rating Gamelan and Allison rating Sun, then there is no way to tell if Tom and Allison are similar or different. Our goal is to find a way to make use of the similarity between Gamelan and Sun in our distance metric. It is simply not feasible to manually find all similar documents and determine their degrees of similarity. Another option is to compute the document similarities based on occurrences of keywords. Solutions have been found in the information retrieval field which allow searching methods to use a thesaurus to find synonymous documents. This solution works well for content-based data similarities, but content alone cannot predict user interest for all types of media.

Webmasters who visit these sites may like the layout of the Gamelan site and dislike the layout of the Sun site. Graphic designers may feel differently about the site because of the quality of the graphics. People on the West Coast may get better download times for Sun's site and therefore prefer to visit there for quick info. Too many factors define user

interest. If we assume that some factors are quite important in determining user interest (for example, content or layout), and that other factors (perhaps length) are less important, it is possible to transform our data into a dissection of those factors, and restrict our discussion to only the important factors.

Rather than looking solely at content, we seek to break down each user's interest into their response to different features of a web page. Instead comparing users based on potentially flawed rating data, ideally we hope to find similar users based on the underlying features inherent in each document. If we know everyone's priorities for each feature of a web page (such as content, layout, graphic design, download speed, etc.) and we can determine how each page matched up to those features, we should be able to predict which pages a given user will like.

Suppose that people interested in fast-breaking news are unconcerned with layout but require fast download times. Also suppose that overall ratings depend on the content of the page the most, the download time the second most, and the layout the least. Simply by looking at pages that contain fast-breaking news, we could narrow down interesting sites by weeding out the ones with slow download times and the ones with bad layout.

By breaking down each web page into its component parts, or main features, we can better predict which users will be interested in which pages. It would be difficult to try to determine what those features are, or even what pages have which specific features and in what amounts. Through statistical analysis of users' rating data, we seek to generate a statistical breakdown of those features, their weights in documents, and users' interest in the features, without ever having to specify exactly what those features represent. Predicting users' interests and finding similar users can then be reduced to simply quantifying those features. We will experiment with theses hypotheses to determine if this breakdown can be successfully accomplished, and if the meaning inherent in the numbers is related to actual user interest.

Latent Semantic Analysis

Our model assumes that documents that are found to be similar share some set of components such as content, display, or length. We further assume that people rate these documents highly because they rate those features highly. If another document were added to the collection that also displayed those features, those same people would also most likely rate this new document highly. By factoring peoples ratings into features using linear algebra, we will predict how users will react to documents they have not seen before, based on their preferences for these features. Linear algebra methods allow us to break down data sets into these components and analyze the principal components of the data. We use singular value decomposition to factor ratings into features, and their observed importance.

Research on Latent Semantic Indexing (LSI) for information retrieval has shown that LSI methods provide a moderate performance benefit over standard cosine analysis retrieval techniques [3]. This research indicates that LSI methods may also provide a moderate performance benefit for collaborative filtering techniques which use the standard cosine analysis for finding neighbors. Our goal is to research whether or not the Singular Value Decomposition from LSI research can enhance collaborative filtering methods on web page data.

Singular Value Decomposition

The user rating vectors can be represented as an $m \times n$ matrix, A, with m users and n documents

$$A = \left[a_{jk} \right]$$

where a_{jk} is the rating of user j for document k. Through singular value decomposition, this matrix can be factored into USV^T, where U and V are orthogonal matrices and the diagonal entries of S are the singular values of A.

U is representative of the response of each user to certain features. V is representative of the amount of each feature present in each document. S is a matrix related to the feature importance in overall determination of the rating. The S matrix is a zero matrix, except for the diagonal entries which are defined as the singular values of A.

Any specific rating can then be recomputed using the U, S, and V matrices as follows:

$$A = USV^T$$
$$A_{ij} = \sum_k \sum_l U_{ik} S_{kl} (V^T)_{lj}$$
$$A_{ij} = \sum_k \sum_l U_{ik} S_{kl} V_{jl}$$

where U_{ik} can be interpreted as user i's reaction to feature k, S_{kk} as the importance of feature k, and V_{jl} as the amount of feature l present in document j.

Due to the nature of S, in every entry for S where $i \neq j$, $S_{ij} = 0$. Therefore, every term in the above summation where $i \neq j$ can be ignored, giving:

$$A_{ij} = \sum_k U_{ik} S_{kk} V_{jk}$$

The above summation shows that the rating A_{ij} can be constructed as a sum over the features of the product of the user i's interest in each feature, the importance of the each feature, and the amount of each feature in document j.

Once the SVD is computed and these matrices are known, it is possible to predict ratings for documents users have not rated. For example, assume a new user enters the picture and rates document d. Assume for all features other than S_{11}, the user's feature weight is zero. Then for all $k \neq 1$ the above summation will be zero, and these terms of the sum can be dropped. In this way, we can compute that users response to feature one (the most important feature) using the linear equation:

$$R_d = U_{i1} S_{11} V_{j1}$$

which is just the summation for $k = 1$. Using this feature weight we can generate prediction ratings for all unrated documents simply by recomputing each a_{ij} using the generated feature weight for that user.

Consider the example of four people surfing the web:

a. Professor Russell

> Professor Russell is an anthropologist. He is also very impatient, and would rather not wait for long downloads. Web pages with good layouts and cool graphics interest him much more than those without.

b. Ed the programmer

> Ed is looking for sites on Java programming. He doesn't mind long download times as long as there is good Java content on the page.

c. Professor Davis

> Professor Davis happens to have his computer science Ph.D., but currently is reading about Pompeii. In his spare time, he dabbles in graphic design and HTML layout, and appreciates a well made site.

d. Jim the Newbie

> Jim is new to the internet and hasn't seen many web sites yet. He has heard about Java at work and is interested to learn more. He also hates to wait very long for huge pages since he only has a 1200 baud modem.

All four people have viewed the following four documents in the past:

1. "The Destruction of Pompeii". This site is well laid out and has excellent graphics, but requires long download times.

2. "Java - Just the Facts". This site is a great Java reference and has very few graphics, so downloads fairly quickly.

3. "XKernel Documentation". Only for the serious programmer. Hard to read with its poor layout and design. Not very interesting to the average user. Extremely short download time.

4. "Mayan Civilization". This site has excellent pictures of Mayan artifacts, but is relatively poor in its layout. The graphics are very high quality and therefore take a long time to download.

Using the descriptions of the first three people above, assume they rate documents 1-4 as follows:

$$M = \begin{bmatrix} 5 & 4 & 2 & 6 \\ 3 & 7 & 5 & 2 \\ 6 & 4 & 1 & 4 \end{bmatrix}$$

The singular value decomposition of M:

$$[U,S,V] = svd(M)$$

$$U = \begin{bmatrix} 0.6000 & -0.4124 & -0.6855 \\ 0.5811 & 0.8136 & 0.0192 \\ 0.5498 & -0.4099 & 0.7278 \end{bmatrix}$$

$$S = \begin{bmatrix} 14.4890 & 0 & 0 & 0 \\ 0 & 4.9324 & 0 & 0 \\ 0 & 0 & 1.6550 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.5551 & -0.4218 & 0.6023 & -0.3889 \\ 0.5982 & 0.4878 & 0.1835 & 0.6088 \\ 0.3213 & 0.5744 & -0.3306 & -0.6764 \\ 0.4805 & -0.5041 & -0.7031 & 0.1437 \end{bmatrix}$$

Generating the singular value decomposition of the rating matrix shows us that the singular values in S drop off fairly quickly. In other words, feature one described in the *S* matrix by "14.4890" is a fairly important feature. By finding a person's reaction to only the most important feature we can generate an initial crude prediction for the rest of the documents.

Assume Jim rates document one as a 2. We will find Jim's probable reaction to feature one using his rating, and assume Jim's feature weights for documents two and three are 0.

Since

$$R_d = U_{i1} S_{11} V_{j1}$$

as shown before, we solve for U_1. To predict R_2, R_3 & R_4 we substitute U_1 into the above equation.

Generating predictions for the other four documents yields:

$$P = \begin{bmatrix} 2 & 2.1554 & 1.1577 & 1.7312 \end{bmatrix}$$

Now suppose Jim visits document two and rates it a seven, because it is just the type of page he is looking for. We make use of this new info to estimate Jim's weight for feature two, U_2. By solving for both U_1 and U_2, we can recalculate the predictions. His predictions can then be adjusted as follows:

$$P = \begin{bmatrix} 2 & 7 & 5.3660 & 1.0166 \end{bmatrix}$$

After rating only one document, the predictions were not very accurate, but without knowing anything about Jim, they represent our best guess with the current information. Consider that Jim rated document one as a 2, which would signify he was similar to Ed the programmer. Ed really liked document two, and so Jim's prediction for document two was slightly higher than the other documents. As we learn more about Jim's tastes, the predictions should become even more accurate.

After rating two documents the predictions start to look more accurate. Since Jim only has a 1200 baud modem, we see that he is more interested in document three, similar to Ed, and much less interested in document four. Since Jim's taste are the most like Ed the programmer, it makes sense for his predictions to also appear fairly close. After Jim rates document three, his neighborhood will shift away from Ed as the numbers show that Jim isn't really interested in all programming content.

Under our model's assumptions, the more the user responds to, the more accurate the predictions will be. If the user responds to three documents, then we can form three linear equations and estimate the user's responses to feature one, two and three. The features are inherently ordered such that feature one exhibits the most importance in deciding the ratings and the importance decreases as you move down the diagonal of S. As

noted before, the singular values of A are directly related to the variance of the data in matrix A.

SVD analysis also offers an interesting way to compare users and find nearest neighbors. Instead of explicitly comparing users based on their ratings, comparing users based on their feature weights may be more accurate. By only comparing feature weights and individual feature importance rather than individual document ratings, we can aggregate each document in the analysis. The result will be a comparison of users based on interests alone. Multiplying both sides of the SVD equation by V yields:

$$M = USV^T$$
$$MV = USV^TV$$
$$MV = US(V^TV)$$
since V is orthogonal...
$$MV = US$$

Taking A and multiplying by the feature document matrix V, yields US, which is the user feature response matrix multiplied with the feature importance matrix. We have moved users into a feature space. By simply using normal cosine distance metrics on the feature weights instead of the explicit ratings, we may be able to find a better set of similar users. Feature space comparison will determine nearest neighbors based on their underlying reaction to the features in the documents, not their explicit reactions to each document. If we throw out the least important features, we will reduce the dimensionality of the comparison and make better use of correlation in our data.

Using the feature weights for comparison should yield a better determination of neighbors and also a more accurate analysis of how close each neighbor is to one another. By then weighting each neighbor's input so that it proportionally influences the data in relation to the neighbor's distance from the user, the prediction should become even more accurate. The Ringo system uses this weighting system to allow closer neighbors more

input on which artists a user may or may not like[15]. We can make a prediction of the user's ratings by taking a weighted average of the n closest neighbors' ratings as follows:

$$R_{user} = \frac{w_1 r_1 + w_2 r_2 + \ldots + w_n r_n}{w_1 + w_2 + \ldots + w_n}$$

where w_i is the i^{th} user's measured closeness to the subject.

For example, taken the given rating matrix, M:

$$M = \begin{bmatrix} 5 & 4 & 2 & 6 \\ 3 & 7 & 5 & 2 \\ 6 & 4 & 1 & 4 \\ 3 & 6 & 4 & 1 \\ 5 & 7 & 3 & 2 \\ 6 & 6 & 4 & 4 \\ 6 & 4 & 3 & 3 \end{bmatrix}$$

To compare users by features, we will solve for *US*.

$$US = \begin{bmatrix} 5 & 4 & 2 & 6 \\ 3 & 7 & 5 & 2 \\ 6 & 4 & 1 & 4 \\ 3 & 6 & 4 & 1 \\ 5 & 7 & 3 & 2 \\ 6 & 6 & 4 & 4 \\ 6 & 4 & 3 & 3 \end{bmatrix} \begin{bmatrix} 0.5682 & -0.4439 & 0.6352 & -0.2768 \\ 0.6364 & 0.4811 & 0.0294 & 0.6022 \\ 0.3696 & 0.4553 & -0.3341 & -0.7379 \\ 0.3681 & -0.6035 & -0.6958 & 0.1271 \end{bmatrix}$$

$$US = \begin{bmatrix} 8.3347 & -3.0059 & -1.5494 & 0.3112 \\ 8.7437 & 3.1051 & -0.9508 & -0.0503 \\ 7.7971 & -2.6980 & 0.8114 & 0.5182 \\ 7.3695 & 2.7723 & 0.0497 & -0.0416 \\ 9.1409 & 1.3068 & 0.9877 & 0.8719 \\ 10.1786 & -0.3700 & -0.1322 & -0.4910 \\ 8.1681 & -1.1839 & 0.8389 & -1.0847 \end{bmatrix}$$

Assume a new user enters with the following ratings:

$$r = [5 \quad 6 \quad 3 \quad 3]$$

We'll place the user's ratings into the feature space. Multiplying r by V.

$$r' = [8.8727 \quad 0.2222 \quad 0.2625 \quad 0.3967]$$

r' represents the user's response to features one through four, weighted by the feature importance.

Since we are comparing features and not ratings, we compare users only be the most important features. By examining the S matrix from the singular value decomposition, we can decide which features are the most important:

$$S = \begin{bmatrix} 22.6915 & 0 & 0 & 0 \\ 0 & 6.0735 & 0 & 0 \\ 0 & 0 & 2.3796 & 0 \\ 0 & 0 & 0 & 1.5961 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Since the singular values, which represent the feature importance, drop off rather quickly, the first two should be adequate enough in finding the nearest neighbor. Using these first two features only in the comparison yields:

$$neighbors = \begin{bmatrix} 0.9319 & 0.9504 & 0.9365 & 0.9445 & 0.9932 & 0.9981 & 0.9858 \end{bmatrix}$$

where *neighbors* is a vector of the cosines of the angles between the user and each neighbor (i.e., neighbors(1) = 0.9319 = cosine(angle between the user and neighbor 1)) which places the neighbors from closest to furthest in the following order:

$$neighbors = [6 \quad 5 \quad 7 \quad 2 \quad 4 \quad 3 \quad 1]$$

Normal neighbor analysis of the ratings yields a neighbor matrix as follows:

$$neighbors = \begin{bmatrix} 0.9126 & 0.9409 & 0.9346 & 0.9430 & 0.9891 & 0.9929 & 0.9682 \end{bmatrix}$$

and places the neighbors in the following order:

$$neighbors = [6 \quad 5 \quad 7 \quad 4 \quad 2 \quad 3 \quad 1]$$

Using only half the data from the ratings to make comparisons, we have produced an almost exact approximation to the original data.

The advantage we obtain by transforming to feature space is that SVD analysis eliminates the error involved when two documents are highly correlated and yet accurate ratings cannot be given because a user has not rated both documents (as long as others have established the correlation between the documents). Using SVD analysis, these highly correlated documents will have high presence of the same features, and a user rating that feature high will find the prediction appropriate to the other document. Also initial predictions for a user who has not rated many documents should be more meaningful, especially if the documents she rates has a high amount of important features present in it. If she rates a document that has a high amount of important features, then her rating will be very meaningful because it will give a description of her response to mainly those features which are the principal components of user interest. Since those features are ordered in S, those features account for most of the variance in the ratings.

Visits

We also used SVD analysis to measure the value of linear algebra analysis on predicting user visits. As before, each user was represented as a vector consisting of user data. To analyze the visit data, each element of the vector holds a one if the user visited the site and a zero if the user did not visit the site. Our model assumes that users who visit similar sites will have similar tastes in which sites they visit. By using normal cosine analysis and latent semantic analysis we can find similar users, and then predict which sites a user will be most likely to visit.

If a user choose which sites to visit and which not to visit in a conscious effort to avoid non-interesting sites, then perhaps we can use their choice to visit a site as an indication of preference for that site. After logging almost 17,000 unique visits, we backtested our theory to try and predict which links users would follow. We assume that similar users will follow similar links. SVD analysis should allow us to find those similar users so we can predict the links they might follow.

Note that by not following a link, the user is not explicitly stating that they chose not to follow that link. Rather it may be the case that they have simply not seen the page that link is on, or had the time to visit the link. Therefore we need a metric that measures predictive quality, but yet did not penalize users for not following links we thought they may have. Instead if a user followed a link and we **did not** predict that they would follow a link, the result was a decrease in our measuring metric. If the user **did** follow the link that we predicted, the value of the metric went up. If the user did not follow a link then it had no effect on the value of the metric. With the rating data we could just compare our predictions with the actual data to see our error, but with the visit data this is more complicated.

Using the nearest neighbors as a predictor, we asked, given the user visited n sites, what was the probability that the user visited exactly the *n* sites they visited and not the sites they didn't visit? The probability that the user visited such sites is given by the following:

```
P(user visited certain sites, and not others) = getProbability(1,n)

      function getProbability(site, totalSites) {
            if (site > totalSites)
                  return 1
            if  (user visited site)
                  return (P(site)*getProbability(site+1,totalSites))
            else
                  return ((1-P(site))*getProbability(site+1,totalSites))
            fi
      end function
```

For example, if out of ten sites, the user visited sites 1,3,5,7, and 9, the probability for those visits is:

```
P(1)*(1-P(2))*P(3)*(1-P(4))*P(5)*(1-P(6))*P(7)*(1-P(8))*P(9)*(1-P(10))
```

We are more interested in discovering the probability, given the user visited *n* sites, that they would visit the sites they visited. In this way we avoid the situation of when the user actually did not visit the sites. This conditional probability is:

```
P(user visited certain sites, given they visited n sites) =

P(user visited certain sites and the user visited n sites)/P(user visited n sites) =

P(user visited certain sites)/P(user visited n sites)
```

We use the log of this probability (for convenience) as our error metric.

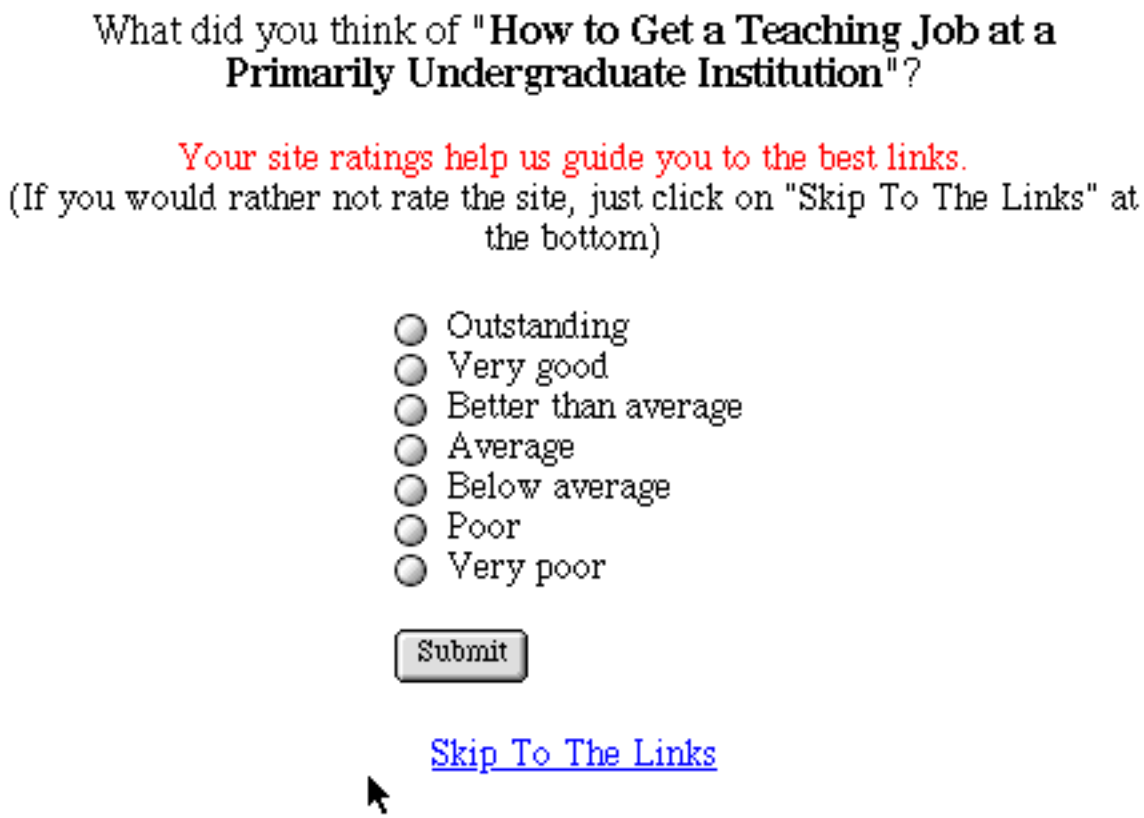
Experiments

All of our testing was done using the PhDs.org site at www.PhDs.org. The site is a web resource for anyone interested in careers in math or science. It has a basic Yahoo-like index setup, where links are grouped hierarchically under subcategories. Links are added by editors to the site.

When users visit the site, a cookie is inserted into their browser to identify them as a unique user. This allows us to keep information from session to session. Users can click through to external links and view the data contained there. Upon returning to www.PhDs.org, they are asked to rate the site they visited on a scale from "Outstanding" to "Very poor". They are also given the option of skipping the rating section if they choose to do so.

Many different setups were experimented with to try and find the most unobtrusive and effective way of getting ratings from users. Initially, we decided to frame the external sites with a small frame on the side that asked for the rating. This presented problems since after they entered the rating we had to eliminate the frame and replace the entire window with the page that they were currently viewing in the right frame. Unfortunately if they had skipped from any page other than the initial page that was linked to www.PhDs.org, there was no way to get that information due to the security sandbox present in the browsers. This setup was replaced with a pop-up window that displayed similar to the frame. After the rating was received the pop-up window would disappear. This solved the problem of removing the rating dialog after the rating was received.

Unfortunately, many of the users found that the abundance of windows was quite annoying. Finally we decided to store the last link that each user visited from www.PhDs.org. When the user returned we would display a page that asked for the rating. This allowed the user to fully explore the site before having to commit to assigning it a rating, and also did not add any extra windows to the browsing environment.



Sample Rating Web Page at www.PhDs.org

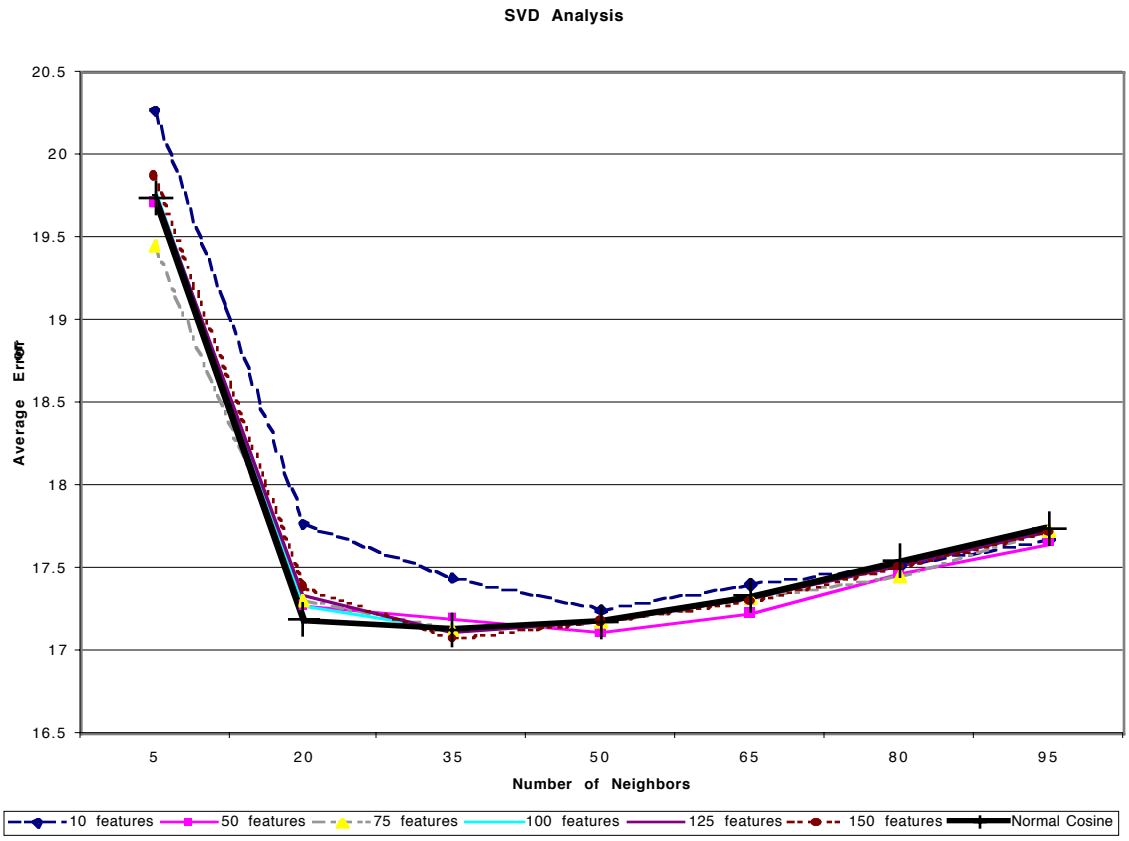
All of the databases are stored in Microsoft Access and are accessed through Cold Fusion, which is a very extendable processing language. SVD analysis was performed using primarily MATLAB 5.2.0, and also with custom tags written in C, and dynamically linked into the Cold Fusion server. The SVD code was taken from *Numerical Recipes in C*.

Description

We conducted four sets of experiments to gauge the success of the SVD analysis. The experiments used visit data collected from users' sessions at www.PhDs.org. The visit data was split into two halves: the first half was used to find nearest neighbors, who were then used to predict probable visits for the second half of the data. We compared the predicted visits with the actual visits to determine our error. For the first set of experiments measured SVD performance against normal cosine analysis. The second set of experiments weighted the predictions such that closer neighbors had more input on the outcome of the prediction. The third set of experiments used the rating data also collected from www.PhDs.org to weight the visit data. If a user visited a link and liked it, we raised the value for their visit to give it more weight. If the user visited the link and disliked it, we lowered the rating to decrease its weight. The final set of experiments combined both the weighted neighbor analysis allowing closer neighbors more input with the rating weighted method which adjusted the visit data based on user ratings.

For each graph the x-axis plots the number of neighbors used to make the predictions. The y-axis plots the average error for the predictions over all users. Each point is then the average error for the predictions using the given number of neighbors. The bold line on each graph is the result using the normal cosine analysis. Other lines are the results of using SVD analysis with differing number of features.

Normal Cosine Analysis vs. Normal SVD Analysis



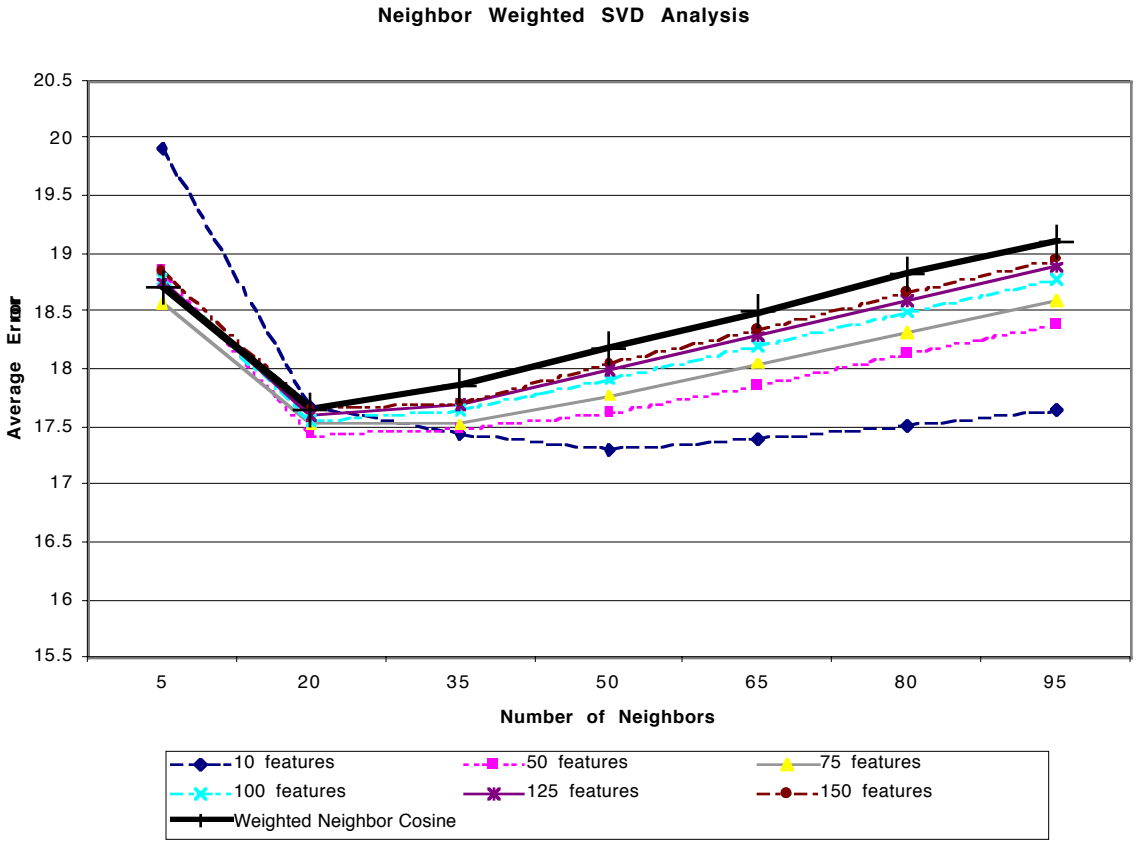
Notice in the above graph that the performance for both the normal cosine analysis and the SVD analysis all follow a particular curve shape. When the number of neighbors is extremely small, the error is extremely high. Most likely a small number of neighbors is over-specific and their similarity on the first half of the data is not indicative of their similarity on the second half. As the number of neighbors increases though, the overall predictive ability of the neighborhood group becomes better until it reaches an optimal number around 20-35 neighbors. Out of a possible 250 neighbors, this suggests that using the nearest 10% of the neighbors in our data to generate predictions is the most successful. As the number of neighbors increases past the optimal number, the predictions gradually worsen. When every user is included as a neighbor, the neighbors predictions become

over-generalized and predict that the user will visit the most popular sites. Predicting the user will visit popular sites performs moderately well in most cases and so the error shown is not as erratic as the over-specific case of too few neighbors.

As we increase the number of features to the total number (250) we are using more and more of the data. Remember that to obtain the feature space, we multiplied the *V* matrix found from Singular Value Decomposition by the visit data. Once we use all 250 features for the predictions, the SVD analysis will behave exactly the same the normal cosine analysis since the distance between two vectors is not affected by the multiplication of an orthogonal matrix, in our case *V*. Therefore as more and more features are added to the SVD analysis the results become closer and closer to those of the normal cosine analysis.

The SVD analysis provides only very small improvements over standard cosine analysis for our test data. The experiments do show that using only 50 features out of a total of 250 for our data set approximates the cosine analysis fairly accurately. This result is significant in showing that the SVD analysis can perform at least as well with only 20% of the data. In cases where the number of users is rather large and speed is an issue, the SVD algorithm could provide improvements over standard cosine analysis. To compute neighbors as efficiently as the cosine analysis, algorithms could use only the most significant features, thereby reducing the computation time.

In two experiments done by Dumais using LSI analysis to improve standard IR techniques, one experiment performed as well as the IR techniques and the other performed moderately better [3]. Dumais mentions that by adding standard enhancements to these techniques could improve the results even more, and so our other experiments focus on weighting neighbor input and using ratings to weight visit data.

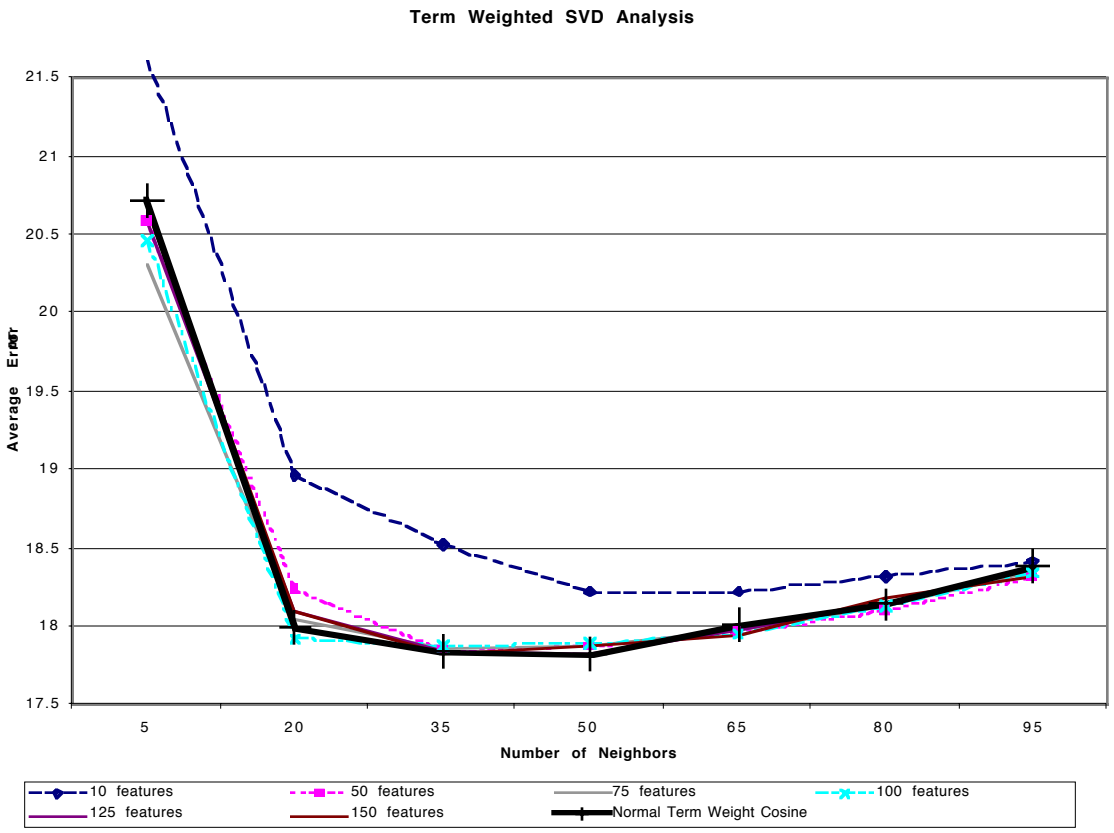


The second experiment consisted of allowing closer neighbors to exert a larger influence on the predictions, and further neighbors to exert less of an influence. The weighting was computed using the cosine of the angle between the neighbor and the user. Since 0 <= cosine Θ <= 1, we were able to form the weighted average probability using the weighting formula given earlier in the paper by multiplying each visit term by the cosine for that neighbor and dividing by the sum of the cosines.

This set of experiments gave better results than standard cosine analysis with the same weighting. Around the optimal number of neighbors (20-35) for the unweighted analysis method our experiments shows that using 20% of the features (50 features) adds a

slight improvement to standard methods. In addition, using only 1/5 of the features for finding neighbors yields the previously mentioned benefits of speedup in neighbor processing and data storage. Analysis with only 10 features yields significant performance benefits over normal weighted neighbor cosine analysis at 50 neighbors.

Term Weighted Cosine Analysis vs. Term Weighted SVD Analysis



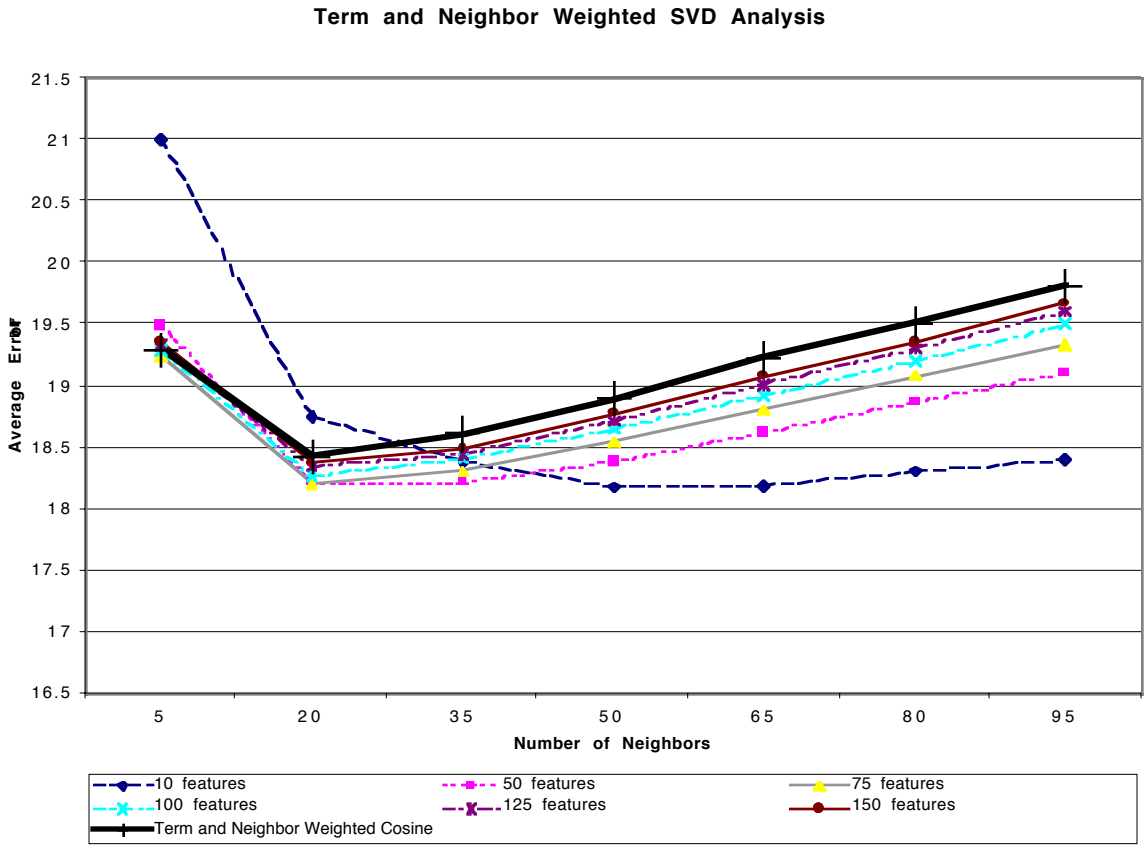
By taking the small number of ratings that we had for our visit data, we weighted the terms of the visit matrix to enhance the implicit visits feedback data with the explicit rating feedback data. If a user had rated a site highly we increased their visit data from 1 to 2. If the user had rated the site poorly we decreased their visit data from 1 to –2. The performance of both methods is rather similar to the normal analysis with no weighting.

The number of rating points that were actually useable from the set of visits data that we used were rather meager. The added effect on the data may not have been significant enough to alter the outcome of the neighbor analysis.

The most likely reason for the decrease in performance is that by weighting certain visits we were explicitly increasing their influence on the neighbor analysis. We explicitly made some visits more powerful in both the positive and negative influence. Ideally we would have had ratings for most of the visit points and weighted them all. Unfortunately, since we did not have the rating data for all points, those without ratings were left at their normal magnitude. In effect, this made these points less influential in comparison to the weighted points. This increase and decrease in the influence of the data points was an arbitrary result of which ratings we had. This arbitrary changing of the influence may have harmed the neighbor analysis more than helped.

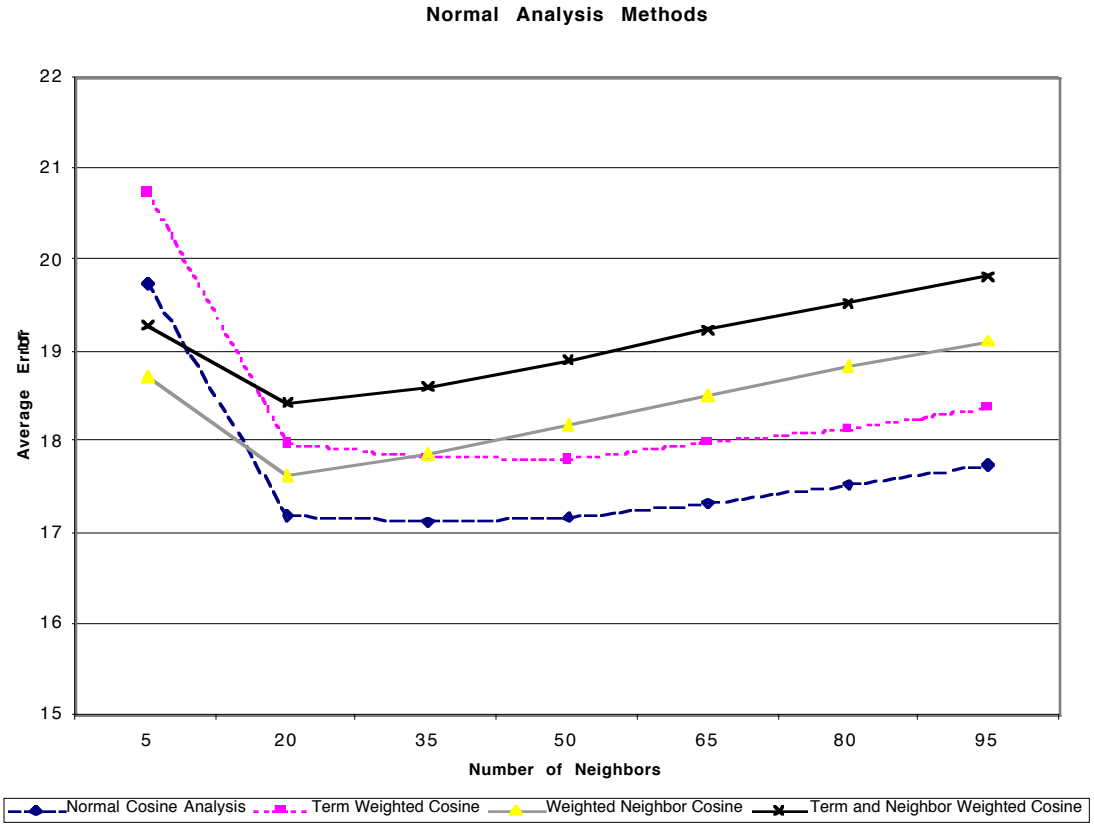
Combined Term Weight and Neighbor Weighted Cosine Analysis

vs. Combined Term Weight and Neighbor Weighted SVD Analysis



In the final experiment we looked at the combined performance of both neighbor weighted predictions with the rating weighting of the visits data. The results were slightly poorer than normal neighbor weighting predictions alone. This is most likely a direct result of the addition of the term weighting. As mentioned previously, the selection of points to be rated was arbitrarily left up to which sites a user had actually rated and was not definitely indicative of a more meaningful visit.

Comparison of Normal Cosine, Weighted Cosine, and Term Weighted Cosine Analysis



In comparing the normal cosine analysis with the weighed analysis, we expect the weighted neighbor analysis to perform at least as well if not better. The weighted neighbor analysis should have given more influence to those neighbors closer to the user and therefore more likely to be correct in their predictions. Our results in the above graph show that this is the case for a small number of neighbors, but once the number of neighbors reaches 20 and beyond, the weighted neighbor analysis performs worse than the analysis which treats all neighbors equally.

Our results suggest that finding a more appropriate approach to term weighting is an important open problem. We produced our most favorable results for the SVD analysis using weighted neighbor analysis. If the weighted SVD neighbor analysis performed better than the weighted neighbor cosine analysis, the SVD could prove to be significantly better at finding neighbors compared to normal methods.

Possible Problems

All of our testing and methodology are based on the assumption that peoples' interests can be expressed as a linear combination of document features, and that those features can be quantified in documents through a simple analysis of peoples' ratings. In order for our predictions to be successful, we further assumed that there were a few significant features which quantified most of the reaction toward any given document.

If people are searching for a specific subject and therefore rate pages that are not geared toward that subject as poor, then they are distinguishing that page from others solely by content. I may find pages about gardening boring, and so I rate them poorly. My mother on the other hand, has no clue what Quake or Doom is and therefore would most likely rate sites on that topic poorly. This distinction is fine, and entirely appropriate, except that the distinction creates a multitude of features that represent each document. My mother will like pages that exhibit the gardening feature. I will like pages that exhibit the Quake feature. Eventually this leads to a multitude of features which distinguish pages by content. This invalidates the assumption that a few features will matter the most. If the document set is limited to a specific group of semi-similar documents, we hope this will not happen. Similar to the solution PHOAKS presents, by grouping ratings for each Usenet group, we hope that on a large scale basis, in order to counter sparse ratings, this system would be implemented in the same way.

Another problem is that when we ask people, "How would you rate this site?", the question can be interpreted in many different ways. Some people could have randomly clicked buttons just to get past the ratings screen. Others may have never given documents low ratings since all the documents at www.PhDs.org shared a similar theme, and users visiting the site may have already had a preference for the material.

Overlap also may have caused a serious problem. Our data was **very** sparse, which in itself is not a problem for our method, but this also meant we had very little overlap, since we had no controls on which pages were rated. In fact, only 26 people rated more than 10 documents, and only 119 number of people rated more than 5, out of a total 1405 users. Most of the visitors to the site probably only visited once to check it out after they heard it mentioned on NPR or read about it in the VOX. In fact we had 765 people only give us one rating (54 %). This may be an artifact of the fact that browsers do not always accept cookies; the number is not entirely indicative of a user only rating one document, since if the person's browser was refusing cookies, they would appear as a new user when they returned. But for the most part, this high number of single ratings is due to the huge amount of browsers who just are passing through, and will never return to this site again. One rating from a given person provides us with no significant data other than a better average rating for the document.

Without overlap in our data, SVD analysis will never be able to describe documents accurately by a small number of features. Each document will be expressed as its own feature. Take for example the case where five people rate five different documents:

$$M = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 3 & 0 & 0 \end{bmatrix}$$

$$[U,S,V] = svd(M)$$

$$U = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$S = \begin{bmatrix} 7 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$V = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Each document simply becomes a feature if there is no overlap. No additional knowledge can be inferred from the data.

We collected ratings for four months and also recorded visits to external sites during that time. Every visit that does not have a rating is a result of the user choosing not to rate the site. We had 16,943 visits and 3307 ratings (20% feedback ratio).

Unfortunately our data is very sparse. In fact, the user by document matrix (377 users by 1405 documents) is only .6% full. This makes it very hard to get any meaningful predictions out of the data. More implicit feedback methods need to be examined to determine efficient ways for collecting the data without forcing the users to become annoyed with our methods.

Summary of Experiments

Our results show that SVD analysis performs at least as well as normal neighbor analysis methods. These results are promising in that they suggest comparative predicting results can be found using a small subset of the data, allowing for faster algorithms to find neighbors and less storage space for each user. The current trend on the web is to provide tailored information to users through collaborative filtering. Sites such as amazon.com and sixdegrees.com are building up user profiles to recommend movies, books, and music to their audiences. These sites generate considerable traffic and need an efficient and fast way to generate predictions. Using Latent Semantic Analysis for these programs could improve their performance.

SVD analysis may even provide moderate improvement in predictive power over normal methods. Depending on the effect of weighting schemes on our data, the SVD method may perform moderately better than the normal analysis methods. The results are indefinite and require further research to resolve.

Acknowledgements

I would like to thank Geoff Davis and Jay Aslam for their guidance and help during the research and writing of this paper. Their clear explanations and constructive feedback were invaluable to the completion of this project.

Bibliography

[1] M. Balabanovic and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation," *Communications of the ACM*, vol. 40, pp. 66-72, 1997.

[2] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using Linear Algebra for Intelligent Information Retrieval," in *Department of Computer Science*. Tennessee, Knoxville: University of Tennessee, 1994, pp. 24.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391-407, 1990.

[4] P. W. Foltz and S. T. Dumais, "Personalized Information Delivery: An Analysis of Information Filtering Methods," *Communications of the ACM*, vol. 35, pp. 51-60, 1992.

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry," *Communications of the ACM*, vol. 35, pp. 61-70, 1992.

[6] K. S. Jones, "Search Term Relevance Weighting Given Little Relevance Information," *Journal of Documentation*, vol. 35, pp. 329-338, 1979.

[7] H. Kautz, B. Selman, and M. Shah, "Referral Web: Combining Social Networks and Collaborative Filtering," *Communications of the ACM*, vol. 40, pp. 63-65, 1997.

[8] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: Applying Collaborative Filtering to Usenet News," *Communications of the ACM*, vol. 40, pp. 77-87, 1997.

[9] H. Lieberman, "Letizia: An Agent That Assists Web Browsing," in *MIT Media Lab*. Cambridge, MA: MIT, 1996, pp. 6.

[10] H. Lieberman, N. V. Dyke, and A. Vivacqua, "Let's Browse: A Collaborative Web Browsing Agent," in *MIT Media Lab*. Cambridge, MA: MIT, 1997, pp. 5.

[11] J. Rucker and M. J. Polanco, “Siteseer: Personalized Navigation on the Web,” *Communications of the ACM*, vol. 40, pp. 73-75, 1997.

[12] G. Salton and C. Buckley, “Improving Retrieval Performance by Relevance Feedback”

[13] G. Salton and M. J. McGill, “The SMART and SIRE Experimental Retrieval Systems”

[14] B. Shapira, P. Shoval, and U. Hanani, “Stereotypes in Information Filtering Systems,” *Information Processing & Management*, vol. 33, pp. 273-287, 1997.

[15] U. Shardanand, “Social Information Filtering for Music Recommendation,” in *Department of Electrical Engineering and Computer Science*. Cambridge, MA: MIT, 1995, pp. 93.

[16] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter, “PHOAKS: A System for Sharing Recommendations,” *Communications of the ACM*, vol. 40, pp. 59-62, 1997.