Eigentaste: A Constant Time Collaborative Filtering Algorithm *

Ken Goldberg and Theresa Roeder and Dhruv Gupta and Chris Perkins
IEOR and EECS Departments
University of California, Berkeley
August 2000

Abstract

Eigentaste is a collaborative filtering algorithm that uses *universal queries* to elicit real-valued user ratings on a common set of items and applies principal component analysis (PCA) to the resulting dense subset of the ratings matrix. PCA facilitates dimensionality reduction for offline clustering of users and rapid computation of recommendations. For a database of n users, standard nearest-neighbor techniques require $O(n)$ processing time to compute recommendations, whereas Eigentaste requires $O(1)$ (constant) time. We compare Eigentaste to alternative algorithms using data from *Jester*, an online joke recommending system.

Jester has collected approximately 2,500,000 ratings from 57,000 users. We use the Normalized Mean Absolute Error (NMAE) measure to compare performance of different algorithms. In the Appendix we use Uniform and Normal distribution models to derive analytic estimates of NMAE when predictions are random. On the Jester dataset, Eigentaste computes recommendations two orders of magnitude faster with no loss of accuracy. Jester is online at:

http://eigentaste.berkeley.edu

1 Introduction

The networked world contains a vast amount of data. Visitors face the arduous task of retrieving information that matches their preferences. The term "Collaborative Filtering" (CF) describes techniques that use the known preferences of a group of users to predict the unknown preferences of a new user; recommendations for the new user are based on these predictions [26]. Other terms that have been proposed are "social information filtering" [30], and "recommender system" [28]. In each case, users *collaborate* in the sense that each rating improves the performance of the overall system. The fundamental assumption is that if users A and B rate k items similarly, they share similar tastes, and hence will rate other items similarly. Approaches differ in how they define a "rating," how they define k, and how they define "similarly."

A CF algorithm should be both *accurate* (the recommended objects should subsequently receive high ratings), and *efficient* in terms of computational complexity. A CF database represents n users and m items available for rating and recommendation. In most existing CF algorithms, online computation scales linearly with n. In this paper we propose a new algorithm, Eigentaste, designed to provide accurate and efficient recommendations to users in constant online time.[1]

Most CF systems include only *user-selected queries*: the user chooses which items to rate, yielding a sparse ratings matrix with many null values. The Eigentaste algorithm profiles user taste with *universal queries*: each item is presented with a short unbiased description (eg, book summary or film synopsis) so that users can form an opinion and respond to any query. Using univeral queries, Eigentaste presents each user with the same *gauge set* of items to rate during its profiling phase. This has the advantage that the subset of the ratings matrix containing the gauge set items is dense. Universal ratings also permit the system to collect immediate feedback on all recommended items. It is often argued that users who have not seen a film are not able to rate it effectively. The opposite can be argued: the experience of viewing a film can corrupt taste ratings by extraneous factors (indigestion, a bad seat, ...). The user's level of familiarity with each item should properly be treated as a confidence, a second-order measure notoriously difficult to collect. Consumers implicitly rate unknown items whenever they shop or make choices. Collecting ratings only of familiar items can yield a highly-biased and incomplete model of user taste. When properly designed, universal queries offer the advantage of rapid and consistent profiling of new users.

Eigentaste captures user ratings on a continuous rating scale. To rate items, users are asked to click their mouse on a horizontal "ratings bar" which returns scalar values. While technically not continuous (limited by the granularity of HTML image maps), we can distinguish approximately 200 levels of ratings in the scale. Continuous ratings avoid discretization effects in matrix computations and may offer

* UCB Electronics Research Laboratory Technical Report M00/41. Pdf format available from www.ieor.berkeley.edu/ goldberg/pubs/. For inquiries, please contact goldberg@ieor.berkeley.edu.

[1] A very brief and preliminary report on this algorithm appeared in [11]. A patent application that includes some elements of this algorithm has been filed by the UC Regents.

measurement [1] and user-interface advantages as discussed in the conclusion.

Eigentaste splits computations into offline and online phases. Offline, Eigentaste uses principal component analysis for optimal dimensionality reduction and then clusters users in the lower dimensional subspace. The online phase uses eigenvectors to project new users into clusters and a lookup table to recommend appropriate items so that run time is independent of the number of users in the database.

This paper is organized as follows. Section 2 reviews related work. Section 3 introduces the Eigentaste algorithm, PCA, and our recursive rectangular clustering method. Section 4 describes the application of Eigentaste to Jester, a CF system for recommending jokes, including a description of the bootstrapping process. Section 5 proposes the normalized Mean Absolute Error (NMAE) metric and compares performance of several algorithms on the Jester dataset in terms of accuracy and efficiency. Section 7 reviews the results and discusses future work.

2 Related Work

In this section we review only a small sample of the papers on Collaborative Filtering. Rich [29] is considered an early reference. There is a long history of patents related to CF, ranging from [14] in 1989 to [9] in 2000. In 1992, D. Goldberg et. al. coined the term "collaborative filtering" in the context of a system for filtering email using binary category flags [10]. Excellent surveys of research can be found in [31, 12, 7].

Shardanand and Maes [30] designed a collaborative filtering system for music (Ringo) and experimented with a number of measures of distance between users, including Pearson correlation, constrained Pearson correlation, and vector cosine. They compare four different recommendation algorithms based on the Mean Absolute Error of predictions. All of their neighborhood-based algorithms require time linear in the number of users.

GroupLens is a pioneering and ongoing effort in collaborative filtering [28, 18, 19, 12]. The GroupLens team initially implemented a neighborhood-based CF system for rating Usenet articles. They used a 1-5 integer rating scale and computed distance using Pearson correlations.

One of the newsgroups that GroupLens considered was rec.humor, an unmoderated newsgroup that receives hundreds of posts a day, most of them not very funny (not that rec.humor.funny is much better). This was reflected in the ratings, where 75% of the jokes received the lowest possible rating of 1 (not funny). The GroupLens team reported correlation values for 500 pairs of users; the relatively high value of these correlations was used to claim that users generally agree on the ratings of jokes, in contrast to the recipes posted on rec.food.recipes.

However, the predominance of "not funny" ratings in the data skewed all correlations dramatically upward. The GroupLens team did note the existence of a substantial number of low and negative correlations in rec.humor, suggesting that there may be indeed be some variance in user tastes. To evaluate Eigentaste, we also use humor as a domain but use jokes with a much higher variance. There are a number of differences between Grouplens and Eigentaste.

Breese et. al. [4] classify collaborative filtering algorithms into two classes: Memory-based and Model-based. Memory-based algorithms operate over the entire user database to make predictions. The most common memory-based model are based on the notion of nearest-neighbors, using a variety of distance measures. Model-based systems are based on a compact model inferred from the data. In this framework Eigentaste would be considered Model-based. Breese et. al. compare a number of algorithms including Bayesian clustering and decision-tree models. They show that Bayesian network and correlation models are the best-performing but do not discuss computational complexity.

Pennock and Horvitz [26] suggest Personality Diagnosis (PD), a latent variable approach based on computing the probability that a new user is of an underlying "personality type," and that user preferences are a manifestation of this personality type. The personality type of a given user is taken to be the vector of "true" ratings for items the user has seen. A true rating differs from the actual rating given by a user in Gaussian noise. Given the personality type of a user A, PD finds the probability that the given user is of the same personality type as other users in the system, and thus the probability that the user will like some new item. To combat the problem common to memory-based models of increased computational effort as the set of existing users grow, Pennock and Horvitz explore a Value of Information (VOI) computation which maximizes predictive value while minimizing the number of explicit ratings needed from a user. This approach requires the specification of utility functions. However, VOI can also be used offline to "prune" the data in the system in order to reduce the amount of data stored while maintaining maximum predictive power.

In a different paper, Pennock and Horvitz [25] propose an axiomatic foundation for collaborative filtering. CF makes preference predictions by combining preferences of existing users. The authors note that "aggregation" of preferences has been studied in Social Choice theory since the 1960's [2]. They argue that only a single nearest-neighbor model will satisfy the axiomatic conditions.

Delgado [7] takes an agent-based approach to CF, de-

veloping several algorithms that combine ratings data with other sources of information such as the geographic location of the user. Weighted majority voting is used to combine recommendations from different sources.

In their recent paper, Herlocker et. al. [12] divide neighbor-based CF algorithms into three steps: i) weighting possible neighbors, ii) selecting neighborhoods and, iii) producing a prediction from a weighted combination of neighbors ratings. They explore alternative methods for each step and propose Spearman (rank-based) correlation weighting as an alternative to Pearson correlations and a "significance weighting" based on the number of items two users have rated in common. To compute predictions they find that subtracting global means improves performance, while conversion to Z-scores does not. To measure accuracy, they propose Receiver Operator Characteristic (ROC) sensitivity from decision-support theory.

Many CF researchers have recognized the problem of sparseness: many values in the ratings matrix are null since all users do not rate all items. Computing distances between users is complicated by the fact that the number of items users have rated in common is not constant. An alternative to inserting global means for null values or significance weighting is Singular Value Decomposition (SVD), which reduces the dimensionality of the ratings matrix and identifies latent factors in the data.

An application of SVD in the context of document retrieval has been patented and is widely known as Latent Semantic Indexing (LSI) [21, 6, 8, 15]. In LSI, SVD is applied to factor the non-square term-document frequency matrix into othogonal factor matrices with corresponding singular values. The largest singular values correspond to the most significant factor weightings, which can be used to create a dimension-reducing linear projection of the original data.

Billsus and Pazzani [3] and Pryor [27] have applied SVD to CF in different ways. Billsus and Pazzani [3] treat CF as a classification problem and discretize ratings into a small number of classes (eg. two: like vs. dislike). Say there are n users, k items that form the basis for recommendation, and r items to consider recommending. Let $m = k + r$. They discretize the $n \times m$ original ratings matrix into a Boolean feature matrix $\mathbf{F}$, with a row for each combination of user and category. They apply SVD to $\mathbf{F}$ to reduce its dimensionality from $2n \times m$ to $v \times m$. The principal vectors are used to project each item to a point in the v-dimensional space.

Billsus and Pazzani then create n feedforward neural networks, one for each user in the database. They use backpropagation to train each network using k v-dimensional vectors (one vector for each item rated by that user). After training, each network will map a v-dimensional vector representing an unseen item to a predicted rating for that item. Using the Movielens dataset, they demonstrate that their method yields reasonable prediction accuracy but note that it is significantly more computationally expensive than other methods due to the need to train a neural network for each user.

Pryor [27] recommends web pages based on Boolean visit patterns and 7-point discrete ratings. Applying SVD to the visit matrix produces a set of vectors corresponding to features in the matrix. He found that using only the most significant features (as measured by their singular values) reduces dimensionality and provides an effective distance metric.

In Eigentaste we address sparseness using universal queries, which insure that all users rate a common set of k gauge items. Since the resulting submatrix is dense, we directly compute the square symmetric correlation matrix and then do a linear projection using Principle Component Analysis, a closely-related factor analysis technique first described by Pearson in 1901 [24, 5, 20, 17]. Like SVD, PCA reduces dimensionality by optimally projecting highly correlated data along a smaller number of orthogonal dimensions.

3 The Eigentaste Algorithm

In this section we describe the generic Eigentaste algorithm.

3.1 Notation and Terminology

U	set of all users in database
J	set of all items to be rated and/or recommended
G	set of items in the gauge set
n	number of users, $\|U\|$
m	total number of items, $\|J\|$
k	number of items in the gauge set, $\|G\|$ (thus there are $m - k$ items available for recommendation)
R	$n \times m$ matrix of raw user ratings
A	$n \times k$ normalized matrix of user ratings of items in G
C	$k \times k$ correlation matrix of the k items in the gauge set
E	$k \times k$ orthogonal matrix of eigenvectors of C
Λ	$k \times k$ diagonal matrix of eigenvalues of C
$\tilde{r}_{ij}$	raw rating of item j by user i; $\tilde{r}_{ij} \in [\tilde{r}_{\min}, \tilde{r}_{\max}] \cup \emptyset$
r_{ij}	normalized rating of item j by user i
p_{ij}	predicted rating of item j for user i
μj	average rating of item j
J_i	set of items rated by user i
U_j	set of users having rated item j ($\forall j \in G$, $\|U_j\| = n$)

3.2 Normalizing Ratings

The $n \times m$ matrix of raw ratings from n users and m items is R. We selected k of these items to form the common gauge set (all valid users rated all items in the gauge set).

We normalize this subset of R to produce A, the $n \times k$ submatrix of gauge set ratings.

Each rating is normalized by subtracting its mean rating over all users, and then dividing by its standard deviation. Since valid users have rated all items in the gauge set, there will be no null ratings. The mean rating of the j^{th} item in the gauge set is

$$\mu_j = \frac{1}{n} \sum_{i \in U_j} \tilde{r}_{ij}$$

and the variance of the j^{th} gauge set item is

$$\sigma_j^2 = \frac{1}{n-1} \sum_{i \in U_j} (\tilde{r}_{ij} - \mu_j)^2.$$

In A, the normalized rating r_{ij} is set to $\frac{\tilde{r}_{ij} - \mu_j}{\sigma_j}$.

3.3 Pearson's Correlation Matrix

If we assume a continuous rating scale and a linear relationship between variables, we can define the global correlation matrix $C(= [c_{jk}])$ over all users.

$$C = \tfrac{1}{n-1} A^T A.$$

C is symmetric and positive definite.

3.4 Principal Component Analysis

Principal Component Analysis was first introduced in 1901 by Karl Pearson [24]. Hotelling generalized it to random variables in 1933 [16]. We apply eigen-analysis to solve for matrices E and Λ such that

$$C = E^T \Lambda E,$$

and

$$ECE^T = \Lambda.$$

Let $B = AE^T$ be a linear transform of A such that the transformed points are uncorrelated (its correlation matrix is diagonal):

$$C_B = \tfrac{1}{n-1} B^T B = ECE^T = \Lambda.$$

Each column j of B has variance λ_j. After sorting by eigenvalue, Figure 1 shows the variance in B left unaccounted for by each successive column of E in a typical dataset.

The idea is to keep only the "principal" eigenvectors. The number of eigenvectors to retain depends on the variances (eigenvalues) but is typically small. If v eigenvectors are retained, data is projected along the first v principal eigenvectors:

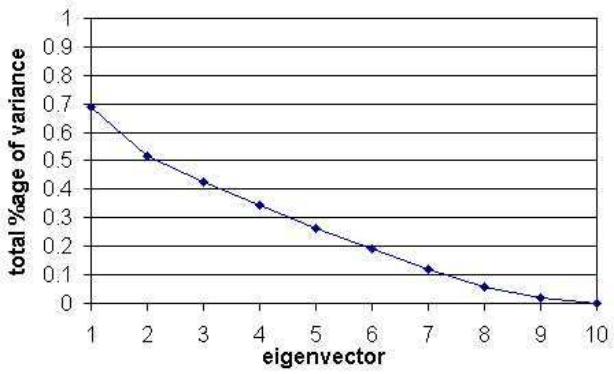

Figure 1: *Scree curve of variances explained by consecutive eigenvectors. The largest amount of variance is explained by the first eigenvector. The first two eigenvectors together account for for almost 50% of the total variance.*

$$\mathbf{x} = RE_v^T$$

On popular choice is to set $v = 2$, so that data are projected onto the "eigen-plane" for human visualization. PCA and the Eigentaste algorithm generalize easily to higher dimensions.

3.5 Recursive Rectangular Clustering

There are many ways to cluster the projected data. For the data in Section 4, after plotting in two dimensions we discovered a high concentration around the origin. We implemented a recursive rectangular clustering where cell size decreases near the origin. This can be generalized to higher dimensions and a variety of alternate clustering methods can be used with Eigentaste.

Clustering

1. Start with the minimal rectangular cell that encloses all the points (user projections) in the eigenplane. This forms the outermost rectangular subdivision.
2. Bisect this cell along the x and y axes to yield 4 rectangular sub-cells.
3. For each new sub-cell that has the origin as one of its vertices, perform the operation in step 2 to generate sub-cells at the next hierarchical level.
4. Repeat Step 3 for each level until a desired depth is reached.

Figure 2 is an illustration of recursive rectangular clustering in 2 dimensions after 4 levels of recursion

We treat each cell as a cluster of neighbors in the eigen plane. For each cluster, we compute the mean for each non-gauge item, based on the number of users who have rated

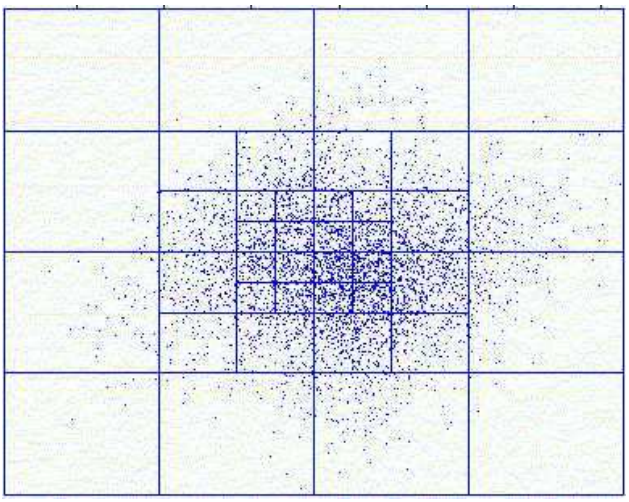

Figure 2: *4 levels of recursion, forming a total of 40 clusters.*

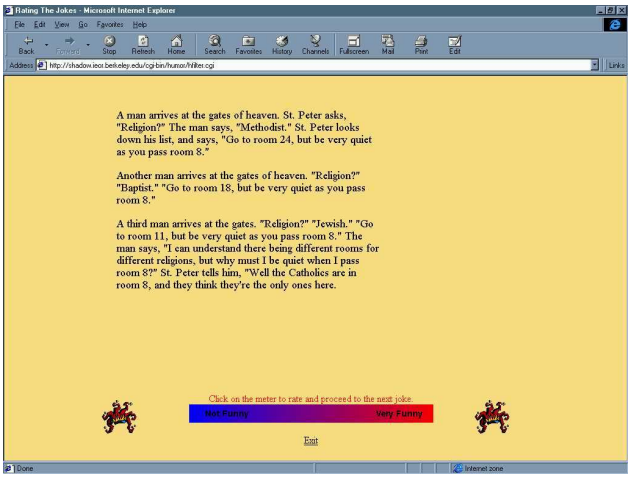

Figure 3: *The Jester interface: users are shown a joke and asked to rate it by clicking on the continuous ratings bar at the bottom of the screen.*

that item. Sorting the non-gauge items in order of decreasing mean ratings yields a lookup table of recommendations for that cluster. All of this occurs offline and periodically.

3.6 Online Computation of Recommendations

When a new user enters the system,

1. Collect ratings for all items in the gauge set.
2. Use the principal components to project this k vector onto the eigen plane.
3. Find the representative cluster.
4. Look up appropriate recommendations, present them to the new user, and collect ratings.

4 Experimental Implementation: Jester

We use humor as a domain for evaluating Eigentaste. Can an automated system recommend a funny joke? Since the criteria for humor are difficult to formalize, this is a non-trivial information retrieval problem. There have been a number of psychological studies on the human sense of humor. Ziv [33] attempted to categorize taste in humor based on social, emotional, and intellectual characteristics. These characteristics in turn depend on factors such as gender, age, social upbringing, etc. Our approach avoids such semantic categories and relies solely on numerical ratings: We treat each user and each joke as a black box. See [23] for research on humor.

We refer to the implemented CF system based on Eigentaste as *Jester*. Jester includes an HTML client interface that allows Internet users to rate jokes[2]. We chose only jokes that can fit on 1-2 screens to minimize evaluation time. We present jokes and collect real-valued ratings as users click on a rating bar implemented using the image map control provided in HTML. After the user rates each joke, another is presented. After all jokes in the gauge set are rated, Jester recommends jokes to the user and continues to collect ratings on each recommended joke. The communication between the interface and the server scripts takes place through a CGI script written in C. Figure 3 illustrates the interface.

All collaborative filtering systems experience the "cold start" problem [22]. One needs ratings to predict ratings. To address this, we started with a simple website to collect joke ratings. We chose the initial set of 40 jokes from friends and newsgroups, doing our best to avoid highly offensive jokes. We then asked 80 friends and students to rate all 40 jokes by visiting the website. We selected half of these jokes ($k = 20$) for the gauge set based on a combination of their correlations and variances. Herlocker et. al. [13] hypothesize that giving high variance items more influence in determining a correlation will improve prediction effectiveness. How to choose the gauge set will be a subject of a future paper.

We next implemented Jester 1.0, a naive recommender system based only on the single nearest neighbor in Euclidean space. We added 30 new jokes to the system ($m = 70$). New users were asked to rate each joke in the gauge set and 5 non-gauge jokes selected at random to seed future recommendations. For each user, the single nearest neighbor was used to generate recommendations from the remaining

[2] See http://eigentaste.berkeley.edu

jokes. Figure 2 is an illustration of the recursive rectangular clustering scheme in 2 dimensions, using 4 levels.

We registered the Jester site with a number of search engines. By the end of November 1998, about a month and a half of the system inception, we had around 150 registered users. On 2 December, at 9:25am, Jester was featured in the Culture section of Wired News [32]. Online news site like Yahoo, Excite, and Netscape News went up with the same story.. This produced a sudden influx of 41350 page requests. Since system process time grew linearly with the number of users, Jester 1.0 was quickly overwhelmed. It crashed and was offline for several days.

Our experience with this traffic overload motivated us to develop a more scalable algorithm, Eigentaste. Jester 2.0 was released on March 1st, 1999. Its graphics were redesigned, we added 30 new jokes to the system so that $m = 100$, and the gauge set was reduced to 10 jokes.

5 Results

5.1 Normalized Mean Absolute Error (NMAE)

The error metric used most often in the CF literature is the Mean Absolute Error (MAE) [30, 4, 26]. If p_{ij} is the prediction for how user i will rate item j, the MAE for user i is defined as

$$\mathrm{MAE} = \frac{1}{\mathrm{c}} \sum_{\mathrm{j}=1}^{\mathrm{c}} |\tilde{\mathrm{r}}_{\mathrm{ij}} - \mathrm{p}_{\mathrm{ij}}|$$

where c is the number of items user i has rated. MAE for a set of users is the average MAE over all members of that set.

Since our numerical rating scale gives ratings over the range $[-10, +10]$, we normalize to express errors as percentages of full scale: Normalized Mean Absolute Error, is:

$$\mathrm{NMAE} = \frac{\mathrm{MAE}}{\tilde{\mathrm{r}}_{\max} - \tilde{\mathrm{r}}_{\min}}$$

Herlocker et al. [12] discusses a variety of other error measures. In the Appendix, we consider NMAE from a theoretical perspective.

5.2 The Jester Dataset

Since March 1999, Jester has collected approximately 2,500,000 ratings from 57,000 users. There are 10 jokes in the gauge set and 90 non-gauge jokes. The average number of ratings per user is 46. We based our experiments on 18,000 random users from this sample. The Jester Dataset, including anonymous ratings from these users, is available upon request.[3]

For the experiments below, we randomly divide the users into two disjoint sets: training and test. We use data from the training set to compute predictions using each algorithm (ie, to "train" the system). Data from the test set is then used to evaluate efficiency and accuracy.

5.3 POP Algorithm

The simplest recommendation algorithm is to treat all users as coming from the same global cluster and to base recommendations for all users on global mean ratings. We used this "POP" algorithm as our control case. (Note: The name "POP" is taken from [4]).

POP predicts ratings for every joke based on its global average. We use the training set to compute the global average and the test set to evaluate the predictions. POP yields NMAE of 0.203.

5.4 Nearest Neighbor Algorithms

The nearest neighbor algorithm and its variants are the ones most widely referenced in the literature. The formula generally used to find the predicted rating p_{ij} for user i and item j is

$$p_{ij} = \bar{r}_i + \kappa \sum_{k=1}^{n} w(i,p)(\tilde{r}_{pj} - \bar{r}_p)$$

where $\bar{r}_i$ is the average joke rating for user i, and κ is a normalizing factor ensuring that the absolute value of the weights sum to 1. The weights $w(i,p)$ can reflect distances, correlations, or similarities between user i and other users that have rated the same items. Most commonly, $w(i,p)$ is the Pearson correlation coefficient between users i and p:

$$w(i,p) = \frac{\sum_j (\tilde{r}_{ij} - \bar{r}_i)(\tilde{r}_{pj} - \bar{r}_p)}{\sqrt{\sum_j (\tilde{r}_{ij} - \bar{r}_i)^2 \sum_j (\tilde{r}_{pj} - \bar{r}_p)^2}}$$

where the summations over j include items that both user i and user p have rated in common [4, 30].

We also implemented a weighted nearest neighbor algorithm. We used a function of Euclidean distance from user i to user p as the weight $w(i,p)$, and $\kappa = \sum_p w(i,p)$. Specifically, if we are interested in q nearest neighbors, $w(i,p) = d(i,q+1) - d(i,p)$. This ensures that i's closest neighbor has the largest weight.

[3] To request this data, please email your contact information and a description of intended research to the first author.

Using only one nearest neighbor (1-NN) proved to be a less accurate predictor than the global average joke rating (POP). For nearest neighbor calculations, the predicted rating for a given test user was the actual rating of his/her nearest neighbor in the training set. Using one nearest neighbor, the normalized MAE was 0.238, an increase over POP.

In addition to finding the prediction error for one nearest neighbor, we calculated errors when recommendations were based on q or more nearest neighbors (q-NN). In these cases, the predicted rating for a user in the test set was the (unweighted) average of the ratings of his q nearest neighbors (one can view this as a special case of the similarity-based weighting, where $w(i,k) = 1/d$ for all $d < q$ neighbors that have rated the item). For $q = 2$, there is sharp improvement: NMAE = 0.224. In addition, the error monotonically decreases until $q = 80$. After this point, the error increases again, and asymptotically approaches the POP value as expected. See Figures 4 and 5.

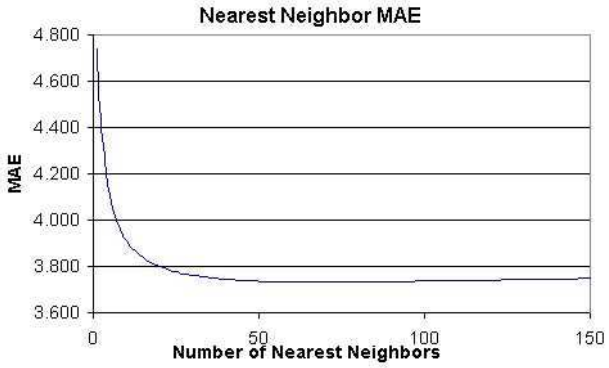

Figure 4: *MAE as a function of the number of nearest neighbors used, for* $1 \leq q \leq 150$.

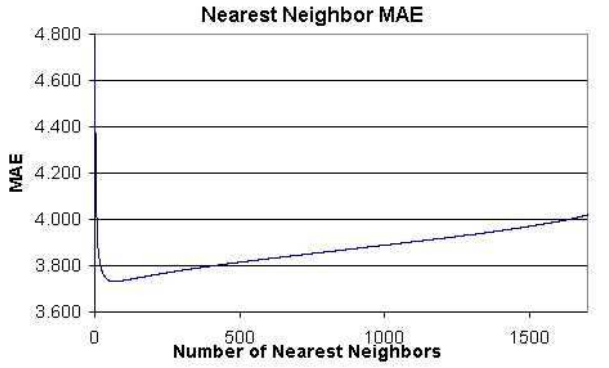

Figure 5: *MAE as a function of the number of nearest neighbors used, for* $1 \leq q \leq 1700$. *The error decreases sharply, then steadily increases for* $q \geq 90$.

As visible in Figure 4, the lowest NMAE is 0.187, for approximately 80 nearest neighbors.

5.5 Eigentaste

For the Jester dataset, the first two eigenvectors accounted for nearly 50% of all the variance (see Figure 1). We took the first two principal components ($v = 2$) and projected the data onto the eigenplane:

$$\mathbf{x} = AE_2^T$$

Each user represents a point in this two-dimensional eigenplane. After computing predictions using the training set and comparing with actual ratings in the test set, the NMAE for Eigentaste is 0.187.

6 Computational Complexity

Recall that the ratings database contains n users and m items for rating and recommendation. In Eigentaste there are $k << m$ items in the gauge set.

6.1 POP Algorithm

The computational complexity for the POP algorithm is $O(nm)$ to compute global means and $O(1)$, constant time, to provide recommendations.

6.2 Nearest Neighbor Algorithms

For 1-NN, the online time needed to find recommendations for a new user is $O(n)$. In an implementation of finding the nearest neighbor, all training set users are scanned into memory, and then traversed for each test set user. The time required to scan all 8853 training set users was 31.64 seconds. To find the nearest neighbor and generate the predictions for one test user took 438 msec. It is important to note that the overriding factor is scanning in the training set users. For a user online, it would take over 32 seconds to get his/her recommendations—assuming there are no other processes running on the system. It may also become necessary to scan training set users from a file if their number grows larger than is reasonable to have in memory. This would significantly increase the time necessary to find predictions.

For q-NN (where $q > 1$) the time required will slightly increase as q grows, but the time to find the q nearest neighbors remains $O(n)$.

6.3 Eigentaste Algorithm

For the offline phase, finding the correlation matrix, calculating the eigenvectors, projecting users into the eigen plane, and clustering them takes $O(k^2 n)$ time. Note that

k is small (in our case $k = 10$), so this phase is not too demanding. Using a similar implementation to the one used for the nearest neighbor algorithm above, it took 27.85 seconds to scan all 8853 training set users, 444 msec to generate the eigenvectors/eigenvalues, and 298 msec to generate the clusters and the predictions for each cluster. The total offline time needed for 8853 training set users was 29.59 seconds.

Online, however, recommendations can be made in constant time: $O(k)$. Thus, there is no increase in time required to find a recommendation as the number of users in the system increases. For Jester, the time to project the ratings from the gauge set and to look up the recommendations is 3.22 msec per user. The Eigentaste algorithm provides a significant decrease in online computation time.

7 Discussion

In this paper we describe Eigentaste, a new CF algorithm that applies PCA to a dense subset of the ratings matrix. Eigentaste uses universal queries to elicit real-valued user ratings on a common set of items. PCA facilitates dimensionality reduction for offline clustering of user and rapid online cluster assignment. Accuracy and efficiency results on the Jester dataset are summarized in the table below.

Algorithm	Accuracy (NMAE)	Offline	Online	Online time per user
POP	0.203	$O(nm)$	$O(1)$	-
1-NN	0.237	-	$O(nk)$	350 msec
80-NN	0.187	-	$O(nk)$	350 msec
Eigentaste	0.187	$O(k^2n)$	$O(k)$	3.2 msec

These Normalized Mean Absolute Error (NMAE) values indicate that predicted ratings values will be within roughly 20% of the true ratings values for each algorithm. So items with predicted ratings well above the mean for a new user will in many cases correspond to desireable items for that user.

It is interesting to note that these accuracies are comparable with those reported for a completely different data set (movies); the algorithms in Herlocker et al [12], when normalized to the 4 unit rating scale (1-5), yield NMAE from 0.192 to 0.207.

The POP (global mean) algorithm offers a useful baseline for accuracy (see Appendix on other baseline comparisons). In terms of NMAE the POP algorithm performs reasonably well, as other researchers have found [12]. It is computationally efficient but completely ignores differences between users.

Nearest-neighbor methods offer improved accuracy, unless not enough neighbors are considered (eg. 1-NN), which makes individual recommendations highly susceptible to to noise. If the 80 nearest neighbors are considered (80-NN), noise is reduced and accuracy improves about 8% over POP, but at the cost of considerable online computational as the number of users grows. Of course it may be possible to preprocess the user group to select or create a small number of representative users ("mentors") to keep n small.

For this dataset and implementation, Eigentaste's accuracy is as good as 80-NN but its online computation is faster by two orders of magnitude. These results suggest that speedup can be achieved without compromising prediction accuracy.

We are experimenting with a number of variations, such as offline k-means clustering with cosine distance measures, and hybrid approaches with adaptive online weighting to further improve accuracy without altering online computation time.

Eigentaste addresses the sparseness problem with *universal queries* instead of user-selected queries; each query contains a short unbiased description (eg, book summary or film synopsis) so that users can form an immediate opinion. Using univeral queries, Eigentaste presents each user with the same *gauge set* of items to rate during its profiling phase. The resulting subset of the ratings matrix is dense. As discussed in Section 1, it can be argued that universal queries are less effective than user-selected queries. But universal queries are particularly appropriate for some domains, such as jokes, news articles, images, and music clips, where a brief sample is available to be evaluated by all users. Although we applied universal queries in the domain of jokes (where the query is simply the joke itself), we are in the process of testing Eigentaste in other domains including books. We plan to write another paper on the design of universal queries and the choice of which items to include in the gauge set. When properly designed, universal queries offer the advantage of rapid and consistent profiling and the ability to collect immediate feeback on all recommended items. The effectiveness of universal queries, confidence queries, and the potential for hybrid query models depends on the domain and is a subject for future study.

Eigentaste captures user ratings on a continuous rating scale using the HTML image map protocol. Continuous ratings have three advantages: 1) they avoid discretization effects in matrix computations 2) they capture taste with finer granularity [1], and 3) users report that they find the continuous rating bar easier to use. Users often report a desire to choose a value "between" two discrete options. The etymology of the word "taste" suggests the digestive system: a user's rating is literally a "gut reaction". If so, the continuous rating bar may offer a more visceral interface. Last, users may provide more data if the interface feels more like navigating a video game than answering a questionnaire. More research on this issue is needed.

In the Appendix, we consider the accuracy metric from a theoretical perspective. We have reported Normalized Mean Absolute Error values for our experiments and noted that these values compare surprisingly well to values reported in other domains and experiments. How significant is a NMAE of 20%? To compare the CF performance to random guessing, we use Uniform and Normal noise distribution models to derive analytic estimates of NMAE. We find that if user ratings are uniformly distributed, random predictions yield NMAE = 33%. This suggests that there is room for improved accuracy for all current CF algorithms.

A Appendix: NMAE for Random Predictions

To compare the CF performance to random guessing, we use Uniform and Normal distribution models to estimate Normalized Mean Absolute Error (NMAE) analytically. Let X be the user's rating and Y be the predicted rating. Let X and Y be independent random variables.

Uniform Distribution

Let X and Y be *uniform* random variables on the interval $[-10, 10]$. The probability distribution of the error, $X - Y$, is a triangular function over the range $(-20, 20)$. Taking the absolute value folds this function onto the positive axis. Normalizing to integrate to 1, the MAE density function, $|X - Y|$, is $f(x) = 0.1 - 0.005x, 0 \leq x \leq 20$. The expected value for the $MAE_{|X-Y|}$ is

$$E[MAE] = \int_0^{20} (0.1 - 0.005x)x dx = 6.667$$

Normalizing over the range of values, NMAE = 0.333. That is, if actual and predicted values are uniformly distributed, we'd expect the random predictions, on average, to be off by a third of full scale.

To confirm, we performed a Monte Carlo experiment where the actual and predicted ratings are random numbers uniformly distributed between -10 to +10. The NMAE we found was 0.320, very close to the analytic prediction.

Normal Distribution

Assume now that both the measured and predicted ratings are Normally distributed random variables with the same mean, μ, and variances σ_1^2 and σ_2^2, respectively, X and Y are, again, independent. From the moment-generating function of random variables,

$$M_{X-Y}(t) = M_X(t)M_{-Y}(t) = e^{\frac{1}{2}\sigma_1^2 t^2 + \mu t} e^{\frac{1}{2}\sigma_2^2 t^2 - \mu t} =$$

$$= e^{(\sigma_1^2 + \sigma_2^2)t^2/2 + (\mu - \mu)t}.$$

Therefore, $X - Y$ is also Normally distributed, with mean 0 and variance $(\sigma_1^2 + \sigma_2^2)$.

For the Jester dataset, the average standard deviation is $\sigma \approx 5$. Assume both the actual and predicted ratings have $\sigma = 5$, the density function for $|X - Y|$ is

$$f(x) = \frac{2}{\sqrt{2\pi}\sigma} e^{-x^2/100}.$$

The expected MAE, then, is:

$$E[MAE] = \int_0^{\infty} \frac{1}{5\sqrt{\pi}} e^{-x^2/100} x dx = 5.642.$$

The expected NMAE is 0.282. That is, if actual and predicted values are normally distributed, we'd expect the predictions, on average, to be off by 28%.

B Acknowledgments

This work was supported in part by NSF Award IRI-9553197. We thank Mark Digiovanni and Hiro Narita for their help in developing the initial version of Jester 1.0, and Matthias Runte for early comparisons with nearest-neighbor methods. We are grateful for the insightful input we received from from Hal Varian, Gordon Rios, Ming Zhang, Adam Jacobs, John Canny, Mike Jordan, Richard Wallace, Alper Atmturk, Marti Hearst, Bob Farzin, Rashmi Sinha, Steve Bui, and Ilan Adler as we developed the algorithm. Thanks to David Pescovitz and Malcolm Gladwell for covering this work in the press. We were fortunate to have received a server from Vivek Sanghi of Intel and excellent real world advice from Leonard Shlain, Reed Gaither, Alan Eyzaguirre, Bob Stein, Pam Aranow, and Jordan Shlain.

References

[1] G. Albaum, R. Best, and D. Hawkins. Continuous vs discrete semantic differential ratings scales. *Psychological Reports*, 49:90–97, 1981.

[2] K.J. Arrow. *Social Choice and Individual Values*. Yale University Press, 2 edition, 1963.

[3] Daniel Billsus and Michael Pazzani. Learning collaborative information filters. In *AAAI Workshop on Recommender Systems*, August 1998.

[4] Breese, Heckermen, and Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *Microsoft Research Technical Report*, (MSR-TR-98-12), October 1998.

[5] B.V. Dasarathy. *NN Pattern Classification Techniques*. IEEE Computer Society Press, CA, 1991.

[6] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391-407, 1990.

[7] Joaquin A. Delgado. *Agent-Based Information Filtering and Recommender Systems on the Internet*. PhD thesis, Nagoya Institute of Technology, February 2000.

[8] C.H.Q. Ding. A similarity-based probability model for latent semantic indexing. In *Proceedings of the SIGIR*. ACM, August 1999.

[9] Alexander Chislenko et al. Us patent 6092049: Method and apparatus for efficiently recommending items using automated collaborative filtering and feature-guided automated collaborative filtering. 18 July 2000.

[10] David Goldberg, David Nichols, Brian Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61-70, 1992.

[11] D. Gupta and K. Goldberg. Jester 2.0: A linear time collaborative filtering algorithm applied to jokes. In *Proceedings of the SIGIR*. ACM, August 1999.

[12] Jonathan Herlocker, Joseph Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the SIGIR*. ACM, August 1999.

[13] Jonathan Herlocker, Joseph Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the SIGIR*. ACM, August 1999.

[14] John Hey. Us patent 4870579: System and method of predicting subjective reactions. 26 September 1989.

[15] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the SIGIR*. ACM, August 1999.

[16] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Eduational Psychology*, 24:417–441, 1933.

[17] J.E. Jackson. *A User Guide to Principal Components: A Problem Solving Approach*. John Wiley and Sons, New York, 1991.

[18] J.A. Konstan and K. Bharat. Integrated personal and community recommendations in collaborative filtering. In *CSCW Workshop*, 1996.

[19] Joseph Konstan, Bradley Miller, David Maltz, Jonathan Herlacker, Lee Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77-87, March 1997.

[20] P.R. Krishnaiah and L.N. Kanal. Classification, pattern recognition and reduction in dimensionality. In *Handbook of Statistics 2*. North-Holland Publishing Company, Amsterdam New York Oxford, 1982.

[21] T. Landauer, M. Littman, and Bell Communications Research (Bellcore). Us patent 5301109: Computerized cross-language document retrieval using latent semantic indexing. 5 April 1994.

[22] D.A. Maltz and K. Ehlrich. Pointing the way: Active collaborative filtering. In *Chi'95 Proceedings Papers*, 1995.

[23] Don Nilsen, editor. *International Journal of Humor Research*. Walter de Gruyter Inc., New York.

[24] K. Pearson. On lines and planes of closest fit to systems of points in space. *Phil. Mag.*, 2:559–572, 1901.

[25] D.M. Pennock and E. Horvitz. Analysis of the axiomatic foundations of collaborative filtering. In *AAAI Workshop on Artificial Intelligence for Electronic Commerce*, Orlando, Florida, July 1999. National Conference on Artificial Intelligence.

[26] D.M. Pennock and E. Horvitz. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *IJCAI Workshop on Machine Learning for Information Filtering*, Stockholm, Sweden, August 1999. International Joint Conference on Artificial Intelligence.

[27] Michael Pryor. The effects of singular value decomposition on collaborative filtering. *Dartmouth College CS Technical Report*, PCS-TR98-338, 1998.

[28] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 1994.

[29] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3:335-366, 1979.

[30] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word of mouth. In *ACM Conference on Computer Human Interaction (CHI)*, 1995.

[31] Hal Varian and Paul Resnick. Special issue on cf and recommender systems. *Communications of the ACM*, 40(3), March 1997.

[32] www.wired.com/news/news/culture/story/_16567.html.

[33] A. Ziv. *Personality and Sense of Humor*. Springer Publishing Co., New York, 1984.