Gauge Optimization for the Eigentaste Collaborative Filtering Algorithm

Jared Bauman | Graduate Student, UC Berkeley

Introduction

In this working paper I summarize the progress that I, with the regular assistance of Professor Ken Goldberg and other members of the Automation Sciences Lab at the University of California, Berkeley, have made towards developing a methodology for selecting an optimal gauge set for the Eigentaste constant-time collaborative filtering initially presented in Goldberg et al. (2001). In particular, I describe several variants on a greedy algorithm for gauge set construction with runtime either linear or quadratic in the number of jokes being considered for inclusion in the gauge set. This report is intended to be accessible for researchers with little or no knowledge of the literature on collaborative filtering. Consequently, the discussion of collaborative filtering algorithms in this paper will be largely informal, with references to more formal explanations included as appropriate.

The intuition behind the Eigentaste algorithm is that if we are able to cluster individuals into various groups based upon their opinions of a well-chosen collection of items, we should be able to use the ratings of users whose opinions are known to predict the preferences of other users in the same cluster for items to which these later users have not yet been exposed. In Goldberg et al. (2001), the set of items whose ratings are used to cluster users—a collection referred to as the gauge set—had to be selected before any data on user preferences was available, so the choice had to be made using informal knowledge of the type of item to be rated.[1] Once all of the users rating of the items in the gauge set have been accumulated, a mathematical technique known as principal component analysis (PCA) is used to reduce each user's ratings of the k jokes in the guage set to a coordinate pair in order to simplify the clustering process: if we consider our collection of gauge set evaluations as a scatter in $k-$dimensional space, PCA finds the pair of mutually perpendicular directions over which the distribution of evaluations proves most volatile. Formally, it accomplishes this by calculating the covariance matrix of item ratings, identifying the eigenvectors of the covariance matrix, and picking the two eigenvectors with the two largest (necessarily

[1] In this case, jokes.

non-negative) eigenvalues.[2]

Once these two vectors are identified, each user's k-dimensional rating is then projected onto the plane spanned by these vectors in order to obtain a two-dimensional representation of the information contained in their $k-$dimensional rating. Using their reduced ratings, users are then assigned to one of forty distinct clusters,[3] When a new user enters the system, she is asked the same gauge questions answered by past users, and her response is projected onto the two eigenvectors obtained through the analysis described above. The new user is then assigned to the nearest cluster, and the ratings of the other members of the cluster are used to predict her preferences.

To test the effectiveness of various gauge sets in clustering like users, it is customary to break the available data set into two parts—a training and a testing set. The user ratings in the training set are treated as the totality of the data available to the Eigentaste algorithm when calculating the eigenvectors to be used to represent the user ratings in two dimensions. The testing data are treated as new users, and these users actual ratings are compared to the values predicted by the cluster to which the user is assigned. The testing metric we are most interested in is the normalized mean absolute error (NMAE). Let r_{min} and r_{max} be the minimum and maximum possible ratings, and let r_{ij} and $\hat{r}_{ij}$ be the actual and predicted ratings of the i^{th} user for the j^{th} joke. Under the assumption that the data set is free of missing values, the NMAE is defined as follows:

$$NMEA = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{|r_{ij} - \hat{r}_{ij}|}{r_{max} - r_{min}}$$

When either actual or predicted ratings are not available for a new user or a cluster, M and N need to be adjusted to account for the number of values omitted from the calculation. In the following pages, I will elucidate a method for identifying a gauge set that minimizes the normalized mean absolute error in out-of-sample testing.[4]

[2]Any invertible matrix can be represented as a change of basis to it's eigenspace, a scaling by its eigenvalues, and a change of basis back to its original vector space. The eigenvectors corresponding to the largest eigenvalues are those corresponding to the greatest scalings in the eigenspace. The plane defined by these two vectors can be thought of as the cross-section of the original data that captures the greatest possible variability in user ratings. For a more thorough explanation of PCA, see Shlens 2003.

[3]See Goldberg et al. (2001) for a thorough description of the clustering process) and the mean ratings of of each item is calculated for each cluster.

[4]For more information on PCA or the Eigentaste algorithm, please consult Shlens 2003 and Goldberg et al. 2001, respectively.

Generalized Gauge Set Optimization

Data

In the following set of experiments, I used the data upon which the original Eigentaste paper was based to attempt to find the joke or jokes that yield a set of clusters that produce a minimal NMEA in out-of-sample testing. All ratings in this data set are rational values lying in the interval [-10, 10]. In this set of experiments, I did not restrict myself to using jokes contained within the original gauge set, but rather consider a large collection of jokes rated by a majority of users. It is for this reason that I restricted my analysis to the original Jester data set, as it alone contains a large subset of users who evaluated a significant sub-sample of jokes.

The original Jester data set contains information on 73,421 users, 48,483 of whom have rated at least 36 of the Jester website's collection of 150 jokes. Usings the sub-sample of users that rated at least 36 jokes, I then identified those jokes that had been rated by at least 90% of respondants. Removing all users that failed to rate at least one of the 36 jokes identified in this manner, I was left with a sub-sample containing the ratings of 39,422 respondants—a collection which included 53.7% of the original users. For each of the following trials, this data set was randomly broken into three equally sized subsets, which were used as training, selection, and testing sets, respectively. The training set is used to identify the first two principal components corresponding to a particular gauge, the selection set is used to compare the performance of potential gauge sets containing the same number of jokes, and the testing set is used to evaluate the performance of the optimal gauge set out-of-sample.

Methodology and Results

As a first step in the evaluation process, it was necessary to establish a series of benchmarks against which all subsequent results could be compared. The first benchmark is the out-of-sample NMAE produced by training our dimensionality-reduction algorithm using the gauge set used in Goldberg et al. 2001.[5] The second, tougher-to-acheive benchmark is the out-of-sample NMAE produced by training our dimensionality-reduction algorithm when all 36 jokes rated by all users in our sub-sample are included in the gauge set.

Having established these benchmarks, I then attempted to determine how well we could predict user ratings using just a single joke.[6] Each of the 36 possible jokes was used as the basis for classification

[5]Note that the jokes in the gauge set are necessarily rated by all respondants.

[6]In the single joke case, no PCA is required to reduce dimension. Clustering was accomplished by dividing the interval

in turn, and the effectiveness of each classifier was measured by its out-of-sample performance on the selection set. The most effective single-joke gauge set was then selected, and evaluated on the test set in order to rule out overfitting that might result from the selection process. In order to see how closely I could approach the performance of the 36-joke PCA, I then tested a greedy algorithm that considered all possible pairs of jokes that might be included in the gauge set.[7] Fixing the first optimal pair, all possible third and fourth jokes were considered for inclusion in the gauge set, and the best pair was then chosen based upon performance on the selection set. This process was also repeated a third time to create a heuristically ideal six-joke gauge set.

All of these evaluations were repeated using the median ratings of each cluster as predictors instead the mean ratings, to see if the NMAE significantly changed when our estimators were made robust to outlying ratings. A similar test was run to see if prediction accuracy improved when inlying values (between -2 and 2 on the -10 to 10 scale) were ignored when training our dimensionality reduction algorithm, under the hypothesis that individuals who did not feel particularly strongly about a joke often chose such intermediate values rather than thinking critically about they thought the joke compared to those that they had previously seen. The results of these experiments are included in the following table.

Method	Mean	Mean, DZ[-2, 2]	Median	Joke(s) Used
10-Joke PCA	0.2022		0.2010	Original Gauge
36-Joke PCA	0.1845		0.1854	Complete Set
Single Joke	0.1995		0.1968	39
Single Joke		0.2117		38
Two Joke (Two at a Time)	0.1927	0.1972	0.1916	38,39
Four Joke (Two at a Time)	0.1898	0.1887	0.1859	14, 28, 38, 39
Six Joke (Two at a Time)	0.1869	0.1886		14, 21, 26, 28, 38, 39
Six Joke (Two at a Time)			0.1837	14, 26, 28, 38, 39, 61

Note in particular how using a singe joke which had not been included in the original gauge set (joke 39) as the basis for clustering proved more effective that using the principal componenets of the original gauge set for this purpose. Furthermore, note that the performance of our six-joke guage set is almost identical to that of the PCA using all ratings of all 36 jokes (marginally worse when using means,

between the largest and smallest observed ratings into twenty sub-intervals, with the lengths of the intervals close to the origin significantly smaller than those located farther away.

[7] As the universe of possible gauge set jokes was quite small, this quadratic-time algorithm proved tractable.

marginally better when using medians). Next, note that using medians in place of means generally leads to improved performance. This should not be too surprising, as the median, rather than the mean, is the minimizer of least absolute deviation, of which the NMAE is one measure. Finally, note that removing ratings in the interval [-2, 2] generally reduces the accuracy of the algorithm. This does not imply that all such methods for removing inliers will prove unhelpful, but suggest that removing a smaller interval (such as [-1, 1]) or a more advanced re-weighting scheme might prove more useful.

Future Work

Going forward, it would be interesting to see how the one-joke-at-a-time gauge set building heuristic performs relative to the two-jokes-at-a-time method when building 4 or 6 joke gauge sets—I've already tried building a two-joke gauge set with the one-at-a-time heuristic–it selected the same jokes as the two-at-a-time method and consequently had near-identical NMAEs. It would also be quite helpful to see if inlier removal worked better if the intervals [-3, 3], [-1, 1], or just [0] were removed instead of [-2, 2]. We could also see if inlier removal strategies combined with outlier removal—such as removing the highest and lowest 2.5% of ratings—perform better than just plain vanilla inlier removal strategies.[8] Finally, something more complicated, such as de-weighting inliers and outliers during the construction of the covariance matrix might prove more effective than simply removing a whole set of suspect observations.

Intra-Gauge Set Optimization

An earlier set of experiments that I conducted using the Jester2+ data set attempted to construct a smaller gauge set using only the 8 questions originally posed to all users. Consequently, no data was thrown out, but approaching the performance of the original 8-joke PCA proved to be much more difficult. When performing these experiments, I did not have the presence of mind to make a distinction between selection and test sets, so the NMAEs that I report are not quite out-of-sample errors in that the process of selecting an optimal gauge set may have introduced some degree of overfitting. However, if the relative performances on the selection and test sets of various gauge sets are any indication, the degree of overfitting introduced in this manner is very close to zero.

[8]Note that inlier removal strategies cannot be combined with median methods, hence the need to be more creative regarding outlier removal.

Method	Mean	Median	Joke(s) Used
8-Joke PCA	0.17507	0.1735963	Original Gauge
Single Joke	0.1875	0.1844	Gauge[8]
Two Joke (Two at a Time)	0.1802	0.1784	Gauge[2,8]
Four Joke (Two at a Time)	0.1771	0.1774	Gauge[2,6,7,8]

Note that, as before, using medians generally leads to better predictions than using means. Furthermore, note that by using just half of our original gauge set we are able to nearly replicate the error rates of our original PCA. This indicates that the ratings that users assign to some jokes are clearly more informative than those that they assign to others, and that we should think of the value of adding additional jokes to the gauge set as having significant diminishing returns.

Effect of Inlier and Outlier Removal on Baseline PCA

Prior to starting my work on gauge-set optimization, I also tinkered with removing inliers and outliers from the Jester2+ data set in order to observe the effect of such a removal on the accuracy of the baseline 8-joke PCA. In addition, I examined what happened when I normalized the covariance matrix prior to performing the PCA, and what happened when principal components other than the first and second were used to perform our dimensionality reduction. Finally, I measured the effect of using a clustering algorithm that concentrated clusters more tightly around the origin–a technique that I shall refer to trisection clustering.[9] The following tables contain the results of these experiments.

Table 1: Effect of Deadzones on NMAE

Principal Componenets	Normalized?	Deadzone?	Outlier Removal?	Clustering	NMAE
1,2	No	None	No	Bisection	0.1836
1,2	No	[0]	No	Bisection	0.1834
1,2	No	[-1, 1]	No	Bisection	0.1826
1,2	No	[-2, 2]	No	Bisection	0.1818
1,2	No	[-3, 3]	No	Bisection	0.1823
1,2	No	[-4, 4]	No	Bisection	0.1835

The takeaway from this table is that there appear to be modes gains to predictive accuracy from removing inliers, and that accuracy is highest when data in the interval [-2, 2] is removed. Note how

[9]Whereas in the original Eigentaste paper the boundries of each cluster were determined by iteratively dividing by two the maximum and minimum values of each two-dimensional coordinates in the reduced representation of each user's ratings, in the 'trisection' clustering scheme, these values were divided by three at each iteration.

this contrasts with my results from the generalized gauge set optimization section, where outlier removal seemed to decrease accuracy.[10]

Table 2: Effect of Deadzones when Observations with $|z| > 1.96$ Are Dropped

Principal Componenets	Normalized?	Deadzone?	Outlier Removal?	Clustering	NMAE
1,2	No	None	No	Bisection	0.1862
1,2	No	[0]	Yes	Bisection	0.1831
1,2	No	[-1, 1]	Yes	Bisection	0.1848
1,2	No	[-2, 2]	Yes	Bisection	0.1852
1,2	No	[-3, 3]	Yes	Bisection	0.1832

Compare the results included in table two those in table one. Note in particular that the NMAEs in table two are generally higher than their corresponding values in table one, indicating that removing the highest and lowest 2.5% of observations (before deadzone values are removed) does not lead to higher predictive accuracy.

Table 3: Effect of Normalization, Clustering Strategies, and Principal Components

Principal Componenets	Normalized?	Deadzone?	Outlier Removal?	Clustering	NMAE
1,2	No	None	No	Bisection	0.1836
1,2	Yes	None	No	Bisection	0.1850
1,2	No	None	No	Trisection	0.1849
1,2	Yes	None	No	Trisection	0.1845
2,3	No	None	No	Bisection	0.1918
2,3	Yes	None	No	Bisection	0.1949
2,3	No	None	No	Trisection	0.1912
2,3	Yes	None	No	Trisection	0.1931

The most obvious result from this series of experiments is that using the second and third principal components is strictly inferior to using the first and second principal components. This should be intuitive, as together the second and third principal components together explain less variability in the distribution of user ratings than the first and second components do together. Second, note that there does not appear to be either a consistent benefit or a consistent penalty from normalizing the covariance matrix before performing a PCA on the gauge set data when the first and second eigenvectors are used. Finally, note that trisection clustering seems to lead to strictly inferious NMAEs than those yielded by the clustering algorithm originally developed in Goldberg et al. (2001).

[10] In particular, [-2, 2] was removed in those experiments.