Brewery Boys

Predicting Beer Ratings through Singular Value Decomposition and Collaborative Filtering

Matt McFarland and Theodore Owens

March 11, 2016

Using a dataset of beer reviews from **Beer Advocate**, we attempt to predict a reviewer's scoring of an unencountered beer based on tastes expressed through their previous reviews. We use two collaborative filtering approaches to make predictions: *Singular Value Decomposition* and *Item-to-Item Collaborative Filtering*.

We find that *Singular Value Decomposition* can generate predicts 2% better than the average baseline predictions. *Item-to-Item Collaborative Filtering* produces a 1% improvement compared to the baseline, conditional on limiting the dataset to users that have made many reviews.

1 Preface

To keep our terminology consistent with existing literature in collaborative filtering, we will take *users* to mean reviewers on Beer Advocate and *items* to mean the beers under review. For a user i and an item j, let Y_{ij} and $\hat{Y}_{ij}$ give the actual and predicted rating of that user on that item, respectively.

2 Problem

We are presented with Y, an $n \times d$ matrix of n users and their ratings of d beers. This matrix contains no other information about the users or beers and is very sparse, as most users have only rated a small subset of items. Our problem can be phrased as: **given a user i and an beer j that user i has not rated, compute $\hat{Y}_{ij}$.**

We use item-based collaborative filtering and single value decomposition to predict how a user will score a beer. These approaches predict review scores based on other user-item interactions in the dataset known to be similar to the target user and item. The accuracy of these prediction algorithms depends on having a large library of historical ratings to use as similarity references [5].

3 Data

Our dataset contains $1,586,599$ reviews concerning $m = 33,388$ users and $n = 65,680$ beers. Each review contains a primary rating on a scale from 0 to 5 by intervals of .5. It also provides ratings in several other metrics (palate, taste, appearance, aroma) and information about the beer itself (brewery, style, ABV). For more information about the raw dataset, see the Appendix. We limit our analysis to predicting the primary rating ("overall review").

Pre-Processing The dataset is comprised by a long list of ratings, where each row represents one user's rating of one beer. To generate our user-by-beer matrix Y, we rearranged these reviews into an n by d matrix, where each row represents a user and each column represents a beer in the catalog. The intersection between a user and a beer contains that user's review score of that beer if available.

To avoid noise and reduce computational complexity, we only include users who have reviewed at least 5 beers and beers that have at least 50 reviews. This leaves us with approximately $5,000$ items and $13,000$ users, where the resulting matrix is 1.8% filled (but still represents the bulk of the data set with more than 1.2 million reviews included).

Splitting Testing and Training Data To partition the dataset into testing and training segments, we randomly separated out a percentage (15% - 20%) of all observed ratings and held those points from the training process. Our prediction models trained on the remaining set of reviews. We used the held-out set of testing data points to evaluate the predictive power of the trained model.

Error Measurement We used the Mean Squared Error (MSE) measurement to calculate the fit of our predictive model. While training, S consists of the training data set. To calculate testing error, we used S as the set of held-out testing data.

$$\mathbf{MSE} = \frac{\sum\limits_{i,j \in S} \left|\mathbf{Y_{ij}} - \hat{\mathbf{Y}}_{\mathbf{ij}}\right|^2}{\left|\mathbf{S}\right|^2}$$

Tools We implemented the data pre-processing and transformation of review list into a user-item interaction matrix with the Pandas library in Python. Item-Based Collaborative Filtering analysis was also done in Python. The Single Value Decomposition analysis was conducted in Matlab.

4 Methods

We use three methods to make predictions:

1. **Baseline**: uses four types of rating means to establish baseline predictions against which we can compare our more advanced approaches.

2. **Singular Value Decomposition**: uses feature reduction to expose principle features in the items and user preferences.

3. **Item-Based Collaborative Filtering**: uses a pairwise correlation matrix of items and predicts based on ratings of similar items

5 Baseline Predictors

To establish a threshold of success for our algorithms, we first calculate a series of average baselines. The most basic predictor consists of predicting the global average rating μ_{global}. Predicting the user's average rating μ_{user} and beer's average score μ_{item} are two other basic predictors.

Additionally, we implemented a baseline predictor used by Simon Funk in the Netflix Prize.[1] First, we calculate the mean for each item (μ_{item}) for each column j. *After* subtracting μ_{item} (each beer's average) from Y, we calculate the mean bias (μ_{bias}) for each user above or below the beer average by averaging across row i. We then construct $\mu_{baseline_{ij}} = \mu_{bias_i} + \mu_{item_j}$ and use $\mu_{baseline_{ij}}$ as the predictor as $\hat{Y}_{ij}$.

Table 1: Results of Mean Predictions

Predictor	μ_{global}	μ_{user}	μ_{item}	$\mu_{baseline}$
MSE	0.4900	0.4193	0.3550	0.3458

Because $\mu_{baseline}$ performs the best out of these baseline predictors, we will compare the predictive success of our machine learning algorithms against the error of this baseline.

6 Singular Value Decomposition

Algorithm

Training The goal of the SVD analysis is to factor Y into U $(n \times k)$ and V $(d \times k)$ matrices whose product (UV^T) well appoximates Y, and where k is the number of Y's largest eigenvalues incorporated into U and V.

To maximize the potential predictive power of this approach, we fit U and V to the residuals of the training data (matrix R) after subtracting the $Y_{baseline}$ prediction from Y.

$$R_{ij} = Y_{ij} - Y_{baseline_{ij}}$$

By centering the data on $Y_{baseline}$, we can regularize our predictive error function to avoid overfitting. (Fitting U and V to non-centered data took far longer to train and produced less accurate results. See TODO cite figure in appendix.) To find a U and V matrix that approximates R, we minimized the difference between the observed training residuals and the predicted residuals.

$$[\hat{U}, \hat{V}] \leftarrow \arg\min_{U,V} \sum_{i,j \epsilon S} \|R_{ij} - U_i V_j^T\|^2 + w_U \|U\|^2 + w_V \|V\|^2$$

Because Y includes missing data, we cannot solve for the closed form solution and must train U and V with a gradient descent minimization of the prediction difference function. S is a set of observed beer ratings, and I is the indicator of S.

$$\frac{\partial E}{\partial U_i} = -2 * \sum_{i,j \epsilon S} (R^{ij} - U_i V_j^T) V_j + w_U U_i$$

$$\frac{\partial E}{\partial V_j} = -2 * \sum_{i,j \epsilon S} (R^{ij} - U_i V_j^T) U_i + w_V V_j$$

Where U and V were updated with schotastic gradient descent (and where λ is the learning rate / step size). While training U and V, this update process continued until the error started to increase.

$$U_{i+1} = U - \lambda * \frac{\partial E}{\partial U} \qquad V_{i+1} = V - \lambda * \frac{\partial E}{\partial V}$$

Prediction To predict a residual for user i and beer j, we take the product of U_i and V_j^T. By adding the baseline prediction to the predicted residual, we get the overall predicted score.

$$\hat{Y}_{ij} = Y_{baseline_{ij}} + \hat{R}_{ij} = Y_{baseline_{ij}} + U_i V_j^T$$

Hyper-Parameter Tuning

The SVD prediction required four hyperparameters: The learning rate (μ), the regularization weights for U (w_U) and V (w_V), and k which determines the number of eigenvectors U and V include. In each training run, we selected the λ possible without causing divergence (varying from 0.5 to .001). Because cross-validate testing among the remaining three hyper parameters would be too time intensive, we opted to select the regularization weights $w_U = 20.0$ and $w_V = 10.0$; such large values helped prevent overtraining. Having decided on a non-divergent μ and weights w_U and w_V, we then executed a form of cross validation to find the best k.

Table 2: Cross Validation of K Results

k	3	5	7	10	15	20
Avg MSE	0.3084	0.3103	0.3110	0.3116	0.3124	0.3148

The Avg MSE is the averaged MSE of the three validation tests for the given K. In the validation test, 20% of the training data was randomly partitioned into a validation set. SVD trained on the remaining portion of the training set, and validation error was calculated with the found U and V on the validation data. Note: We cross validated on a smaller subet of the dataset (7,000 users and 800 items) because using the full dataset would have been too computationally intensive for such analysis.

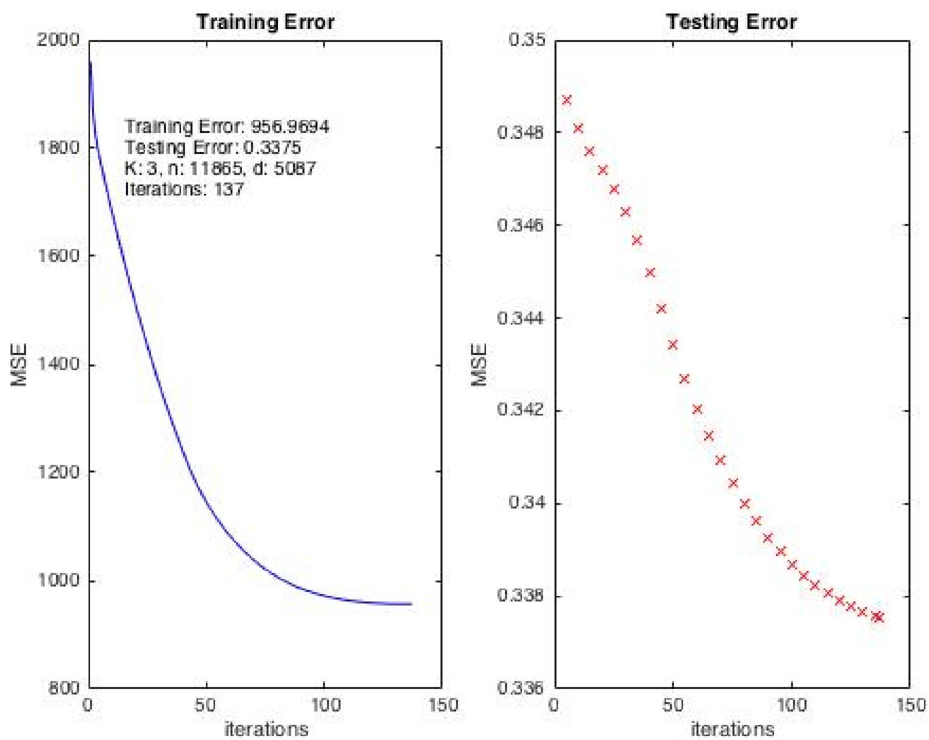
Cross validation results showed that a small $K = 3$ generated the lowest average MSE.

SVD Results

Initially, we tried to fit U and V to the raw scores instead of to the residuals (with the found best $k = 3$). Using the product of the trained U_i and V_j as the predicted score $\hat{Y}_{ij}$ produced a testing error of 0.4584 which was unacceptably higher than the best baseline.

To improve performance, we altered our training algorithm to fit U and V to the residual matrix R, and summed the predicted residual and predicted baseline to get the review prediction. Using the found best $k = 3$, regularization weights $w_U = 20.0$ and $w_K = 10.0$, and learning rate $\mu = .5$, we trained the SVD on the full dataset of review residuals (reserving 15% of the data as testing data). We were able to realize an MSE of 0.3375 which represented an improvement of 2.4% over the bias baseline predictor (MSE of 0.3458).

Figure 1: SVD Best Prediction Results



7 Item-Based Collaborative Filtering

Algorithm

Training For this method, we also from the residual set of ratings of R, where the user and item biases have been removed. We generate an $d \times d$ correlation matrix C from R. The entry C_{ij} describes the similarity of residuals between items i and j. We use the Pearson Correlation to determine these similarity scores, which fall in the range $[-1, 1]$.

Our prediction step requires knowing which items should be considered "similar" enough to each other to use as a basis for a prediction. We must discretize C such that correlation scores above a certain threshold s^* are considered "similar".[5] By applying this threshold to all entries in C, we generate an $d \times d$ matrix S, where:

$$\mathbf{S}_{ij} = \begin{cases} 1 & \text{if items } i \text{ and } j \text{ are similar } (C_{ij} > s^*) \\ 0 & \text{otherwise} \end{cases}$$

Prediction Armed with the similarity matrix S, we can make predictions. For a given user i, we wish to predict his rating on an item j that he has not yet rated, given his past ratings. Letting S be a set of items similar to the predicted item j that the user has also rated, We predict $\hat{Y}_{ij}$, where:

$$\hat{Y}_{ij} = \mu_{baseline_{ij}} + \frac{\sum\limits_{s \in S} Y_{is}}{|S|}$$

We predict our baseline plus an item-based collaborative filtering term.[2] This term sums the users ratings for items similar to j and divides by the number of similar items that the user has rated (takes an average).

Hyper-Parameter Tuning

Item-Based Collaborative Filtering requires setting s^* to a threshold to determine whether two items are similar or not. If Figure 2, we calculate the prediction error for various similarity thresholds. We observe that this algorithm has the lowest prediction error where $s^* = 0$. As the s^* increases, we have fewer similiar items from which to generate the collaborative filtering term, leading to greater variation in the "average of similar items". Even though non-correlated items are viewed as "similar" when $s^* = 0$, our results improve on the best baseline by removing all *dissimilar* items from the comparison.

Item Based Collaborative Filtering Results

Given the dependency of Item-Based Collaborative Filtering on a user's ratings of similar items, we run the algorithm against two datasets. The first dataset is consistent with our approach in SVD, and considers items with at least 50 ratings and users that have given at least 5 ratings. Under these conditions, we cannot with any regularity beat the baseline.

In the second data set, we restrict the second dataset to users that have made at least 500 ratings (with the same item requirement of 50 ratings). With this restriction in place, we find a modest 1% improvement compared to our baseline.

When the feature matrix is denser (as in the user restricted dataset), our collaborative filtering approach strengthens relative to the baseline. There is a greater chance that any given item is both similar to the current predicted item *and* the user has rated that similar item.

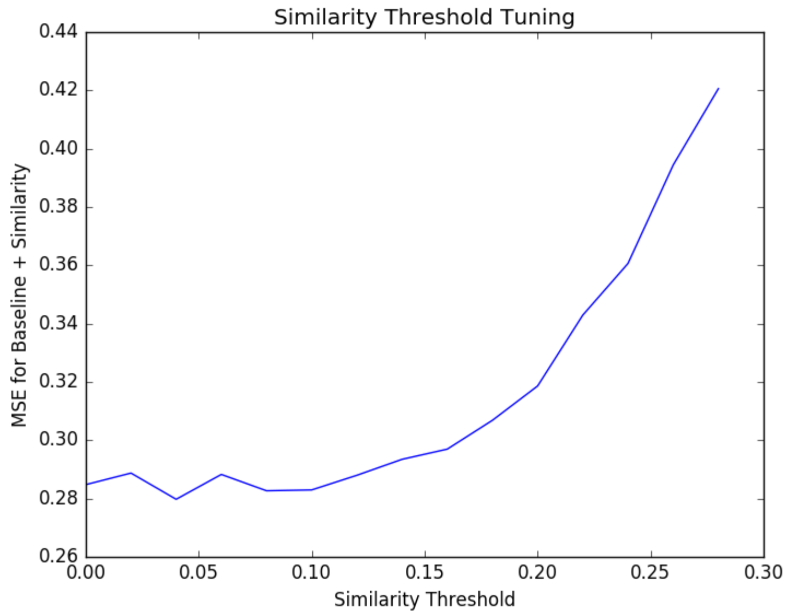


Figure 2: Prediction Error for Similarity Thresholds

Table 3: Item-Based Collaborative Filtering Results

	Baseline (Test)	CF (Test)	Baseline (Train)	CF (Train)
Unrestricted Data	0.3629	0.3440	0.3304	0.2359
User Restricted Data	0.3045	0.3011	0.3042	0.1918

We also do note from testing our method against the training set that the collaborative filtering approach yields results that are significantly better than the baseline, suggesting a degree of overfitting. Regularization of the similarity matrix might alleviate the overfitting issue.

8 Comparison of Methods

We have demonstrated how both SVD and item-based collaborative filtering result in modest improvements compared to the best baseline prediction. Given how little variance exists in the distribution of reviews (see the Appendix section of data characterization), we consider any improvement on the best baseline to be a success. Comparing the two prediction methods against one another, we find SVD advantageous due to its feature reduction principle.

In SVD, we expose latent features that group the items along certain unkown criteria based on correlations in user ratings. By simplifying items and users into similar groups, we dimensionally reduce the enormity of the beer catalog and user population to a few stereotypes that illuminate common traits between beers and users.

Oppositely, item-based collaborative filtering relies only on direct comparisons between two items. Given that the number of pairings grows with the square of the number of items, we require a vast number of comparisons to make meaningful observations about the similarity of the two items.

Further, in the prediction stage, item-based filtering only takes advantage of knowledge drawn from "similar" items. However, knowing a user's opinion of *dissimilar* items could be useful as well. If a user likes a dissimilar item, we may infer the user will dislike the predicted item.

So while both methods can contribute to more accurate predictions beyond the best baseline, SVD is superior in its ability to find defining common traits among users and beers whereas item based collaborative filtering can only examine items known to be similar to each other.

In futher exploration, it would be interesting to see how SVD and item-based collaborative filtering could be used to augment one another's predictions in an ensemble prediction. Understanding where each method is uniquely strong and weak would show how to ocmbine the metho

9 Contributions

Matt implemented SVD algorithm and worked with Matlab. Ted focused on pre-processing features in Python and explored the item-based collaborative filtering approach. We each wrote the sections of the report that are relevant to our respective algorithms. Ted drafted opening and closing remarks, edited and finalized by Matt.

References

[1] Funk, Simon, "Netflix Update: Try This at Home" The Evolution of Cybernetics. Web. 11 Dec. 2006.

[2] Gower, Stephen. "Netflix Prize and SVD." (n.d.): n. pag. 18 Apr. 2014. Web. 9 Mar. 2016.

[3] Ma, Chih-Chao. "A Guide to Singular Value Decomposition for Collaborative Filtering." (n.d.): n. pag. Depart of Computer Science, National Taiwan University. Web. 9 Mar. 2016.

[4] Paterek, Arkadiusz. "Improving Regularized Singular Value Decomposition for Collaborative Filtering." Institute of Informatics, Warsaw University. Web. 12 Aug. 2007.

[5] Sarwar, Badrul, George Karypis, Joseph Konstan, and John Reidl. "Item-based Collaborative Filtering Recommendation Algorithms." Proceedings of the Tenth International Conference on World Wide Web - WWW '01 (2001): n. pag. Web.

Appendices

A Dataset Characterization

Figure 3: Sample Ratings

brewery_name	overall score	aroma	appearance	reviewer	beer_style	palate	taste	beer_name	beer_abv
Vecchio Birraio	1.5	2	2.5	stcules	Hefeweizen	1.5	1.5	Sausa Weizen	5%
Vecchio Birraio	3	2.5	3	stcules	English Strong Ale	3	3	Red Moon	6.2%
Vecchio Birraio	3	2.5	3	stcules	Foreign / Export Stout	3	3	Black Horse Black Beer	6.5%
Vecchio Birraio	3	3	3.5	stcules	German Pilsener	2.5	3	Sausa Pils	5%
Caldera Brewing Company	4	4.5	4	johnmichaelsen	American Double	4	4.5	Cauldron DIPA	7.7%

Table 4: Whole Dataset Summary Statistics

Number of Reviews	1,586,599
Number of Items	65,680
Number of Users	33,388
Rating Minimum	5.0
Rating Maximum	0.0
Rating Mean	3.82
Rating Variance	0.52
Rating Standard Deviation	0.72

Figure 4: Rating Distribution

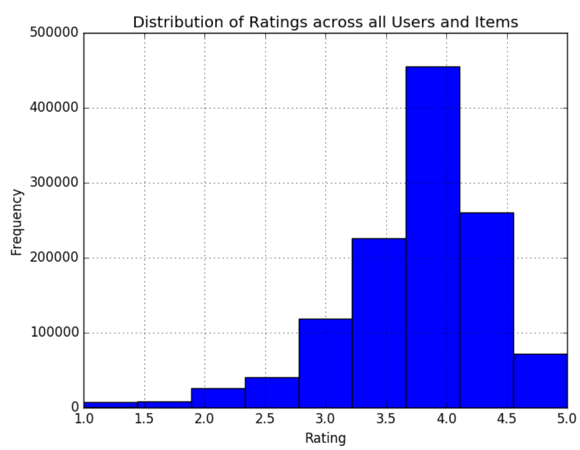


Figure 5: Distributions of Number of Ratings by Item and by User