Brewery Boys
Predicting Beer Ratings through Singular Value Decomposition and Collaborative Filtering

Matt McFarland and Theodore Owens

March 10, 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 rating baseline predictions. **Item-to-Item Collaborative Filtering** struggles to reach similar accuracy to the best average-baseline prediction measure.

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

Collaborative Filtering We use collaborative filtering to predict how a user will score a beer. Scores are predicted based on how the rest of the user population rated beers similiar to the beers in the target user's history. Collaborative Filtering more accurately predicts new user-item interactions by having a large library of historical ratings to use as similarity references.

Training Our prediction for unseen user-item interactions depends on identifying similiar items to the target item. To establish item similarities for Item-Based Collaborative Filtering, we constructed a similiarity correlation matrix between all pairings of items. In the Single Value Decomposition Analysis, the decomposition of the user-item matrix reduces the feature dimensions of the item space into most-similar basis vectors.

3 Data

Our dataset contains $1,586,599$ reviews concerning $m = 33,388$ users and $n = 65,680$ items. Each review contains a 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 item. To generate our user-by-item 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 uniformly separated out a percentage (15% - 20%) of all observed user-item interactions 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 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.

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

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**: use rating means rather than machine learning approaches to establish a baseline prediction 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 correlation matrix comparing all pairs 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.

We used another baseline predictor used by Simon Funk in the Netflix Prize [1]. First, we calculate the mean for each item (μ_{item}). *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. We then subtract the user bias from each row, reducing Y to residuals. We form the predicted rating $\hat{Y}^{ij}_{baseline}$ for user i on item j by calculating: $\hat{Y}^{ij}_{baseline} = \mu_{bias_i} + \mu_{item_j}$. The accuracy of these baseline predictors are included below.

Table 1: Results of Mean Predictions

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

6 Singular Value Decomposition

Theoretical Basis Single Value Decomposition factors an n by d matrix Y into approximation matrices $U * \Sigma * V^T$, where U is n by K and V is d by K. In our case, Y is a large, sparse matrix of user-item interactions, and we can simplify the features of Y by finding the primary latent features in U and V (where Σ is a diagonal matrix multiplied into U and V) for the K largest eigenvectors. These latent features expose the directions of greatest variation, allowing us to identify items and users that are most similar in our dataset and build predictions based off of those similarities.

Previous Work Single Value Decomposition has been used as a prediction mechanism for user-item interactions very successfully before, most famously in the famous Netflix Prize.[2] Complex approaches, such as adaptively altering the hyperparameters or incorporating time-sensitivity to a user's history, have been attempted to maximize the accuracy of predictions.[3][2] We will attempt to adapt and apply some basic, successful techniques to our dataset.

Algorithm

The goal of the SVD analysis is to decompose Y into a U and V matrix whose product well appoximates Y. To find a U and V matrix that approximates Y, we minimized the difference between the observed training ratings and the predicted ratings.

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

Because Y is incomplete, we cannot solve for the closed form solution and thus must use gradient descent to minimize the error function by iteratively updating U and V. (S is the set of observed user-item iteractions, and I is the indicator of seen, training ratings for Y.)

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

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

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

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

To maximize the potential predictive power of this approach, we fit U and V to predict the residuals of the training data after subtracting the $Y_{baseline}$ prediction. We build a prediction for user i and item j by adding the baseline predictor and the residual prediction: $\hat{Y}^{ij} = Y^{ij}_{baseline} + U_i V_j$. We evaluated the testing error based on the difference between the review baseline residual and the predicted residual.

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 largest learning rate 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

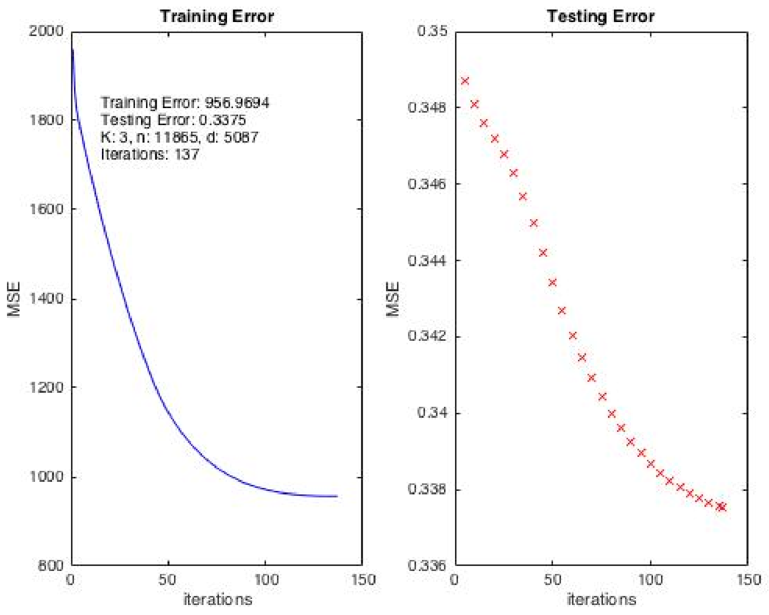
The Avg MSE is the averaged MSE of the three validation tests for the given K. In the validation test, 20% of the training data was randomly partitioned into a validation set. SVD trained on the remaining portion of the training set, and validation error was calculated with the found U and V on the validation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

After determining the 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 Prediction Results



Item-Based Collaborative Filtering

Algorithm

Training For this method, we similarly begin from our residual feature matrix $\bar{\mathbf{X}}$. We generate an $n \times n$ correlation matrix $\mathbf{C}$. The entry $\mathbf{C}_{ij}$ describes the similarity of ratings between items i and j. If a particular user reviews both items i and j favorably, the similarity increases. If the user reviews both poorly, the similarity also increases. If the user rates one postively and the other negatively, the similarity decreases. We use the Pearson Product-Moment Correlation to determine these similarity scores, which fall in the range $[-1, 1]$.

Our prediction step requires knowing which items are "similar" to each other. We must discretize $\mathbf{C}$ such that correlation scores above a certain threshold s^* are considered "similar". By applying this threshold to all entries in $\mathbf{C}$, we generate an $n \times n$ matrix $\mathbf{S}$, where:

$$\mathbf{S}_{ij} = \begin{cases} 1 & \text{if items } i \text{ and } j \text{ are similar} \\ 0 & \text{otherwise} \end{cases}$$

Prediction Armed with the similarity matrix $\mathbf{S}$, we can make predictions. For a given user i, we wish to predict their rating on an item j that they have not yet rated, given their 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:

$$\hat{y}_{ij} = \mu_{user_i} + \mu_{item_j} + \frac{\sum_{s \in S} \bar{\mathbf{X}}_{\mathbf{is}}}{\sum_{s \in S} 1}$$

We predict our baseline plus an item-based collaborative filtering term. 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

This algorithm requires setting a threshold s^* to discretize whether or not items are similar to each other. We first explore how tuning s^* impacts our results in terms of Mean-Squared Error.

Item Based Collaborative Filtering Results

Comparison of Methods

Next Steps

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.

Appendices

Dataset Characterization

Figure 2: 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 3: 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 3: Rating Distribution

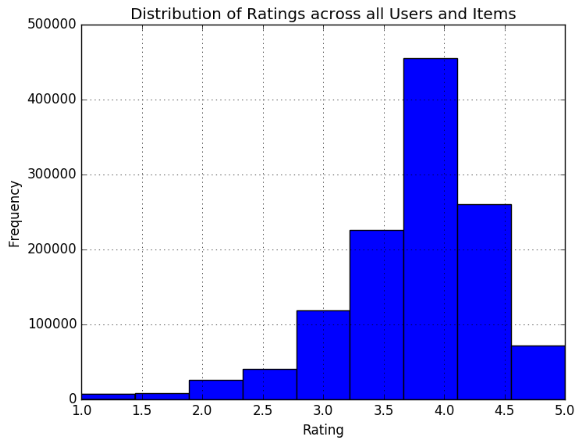
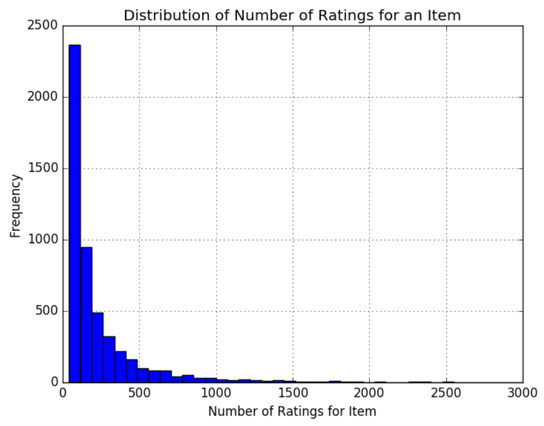
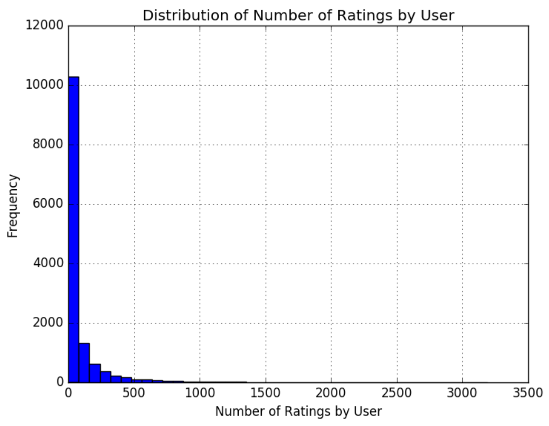


Figure 4: Distributions of Number of Ratings by Item and by User


Distribution of Number of Ratings for an Item
Frequency
Number of Ratings for Item


Distribution of Number of Ratings by User
Frequency
Number of Ratings by User