Brewery Boys

Predicting Beer Ratings through Singular Value Decomposition and Collaborative Filtering

Matt McFarland and Theodore Owens

March 9, 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**.

Our baseline establishes... We find...

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 $\mathbf{X}$, an $m \times n$ matrix of m users and their ratings of n items. This matrix is sparse, as most users have only rated a small subset of items. We have no information about the nature of the users (e.g. age, gender) or the nature of the 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 Given the lack of feature information (categorization of items, demographics of users, etc.), we cannot rely on supervised learning techniques. We thus turn to unsupervised methods and collaborative filtering, in particular. Collaborative Filtering techniques attempt to establish similarities in user preferences for certain items based on observed user-item interactions. Predicted ratings rise when a user favorably rates items similar to the predicted item.

Training This process first requires establishing which items are similar to each other. Intuitively, if a user rates two items in the same fashion (both poorly or both well), then

they are similar. If the user rates them oppositely, they are dissimilar. In our SVD analysis, we establish a set of latent features describing items that are similar in some respect, as determined by user preferences. For Item-Based Collaborative Filtering, we generate a correlation matrix that describes the similarity between all pairings of items.

Prediction Intuitively, if a user favorably rates item A, which is quite similar to an unrated item B, we expect the user to favorably rate item B. In the prediction stage, we make use of the user's provided ratings to make inferences about an unkown rating. In SVD analysis, we can predict the score of an unreviewed item by finding the product of that user's latent feature vector multiplied by that item's latent feature vector.

3 Data

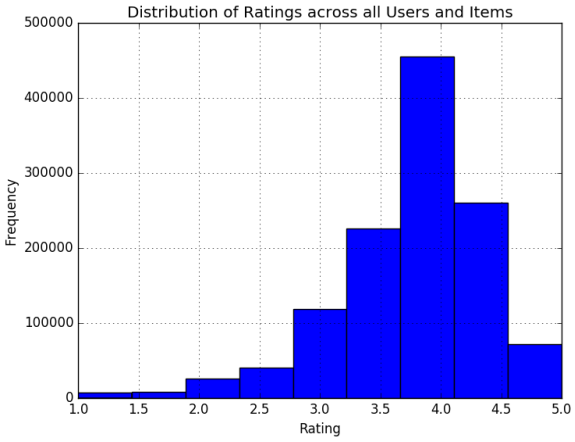
Our dataset contains $1,586,599$ reviews concerning $m = 33,388$ users and $n = 65,680$ items. Each review contains a rating on a scale from 0 to 5 by intervals of $\frac{1}{2}$. It also provides user rating along several other metrics (palate, taste, appearance, aroma) as well as some information about the item itself (brewery, style, alcohol by volume).

Figure 1: 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%

We limit our analysis to use the primary rating.

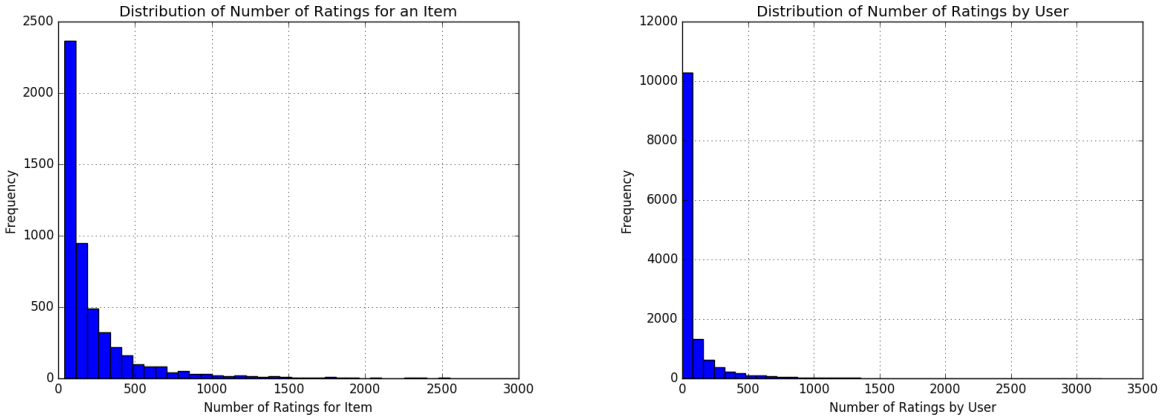
Figure 2: Rating Distribution



In *Figure 2*, we observe that a majority of the ratings fall into the 3.5, 4, and 4.5 categories.

Not all items receive the same number of reviews, nor do all reviewers rate the same number of items. In *Figure 3*, we find that our dataset is highly skewed, with many items and users possessing quite few reviews.

Figure 3: Distributions of Number of Ratings by Item and by User



Each review contains a rating on a scale from 0 to 5 by intervals of $\frac{1}{2}$. It also provides user rating along several other metrics (palate, taste, appearance, aroma) as well as some information about the item itself (brewery, style, alcohol by volume). We limit our analysis to use the primary rating ("overall review").

4 Pre-Processing

The dataset is arranged into a list of ratings, where each rating represents one user's rating of one item. To generate our user-by-item matrix $\mathbf{X}$, we must rearrange these reviews and extract the primary rating.

We additionally remove all users who have not rated at least 5 items and all items that have not received at least 50 ratings. The denser the network of users and items, the better we can establish similarities amongst preferences and items. Removing obscure items and reviewers whose tastes are relatively unknown reduces the noise without hindering prediction quality. This leaves us with approximately $5,000$ items and $13,000$ users, where the resulting matrix is 1.8% filled.

Additionally, we form a modified user-by-item matrix $\bar{\mathbf{X}}$, where:

$$\bar{\mathbf{X}}_{ij} = \mathbf{X}_{ij} - \mu_{user_i} - \mu_{item_j}$$

We derive the μ terms in our baseline algorithm. Intuitively, $\bar{\mathbf{X}}$ represents the residual of a rating that is not explained by the item's average (its "quality") and the user's rating tendencies.

Table 1: 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

(a) Summary Statistics

5 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 latent 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

6 Error Measurement

In accordance with the literature standard, we evaluate the performance of our algorithms with the Mean Squared Error (MSE) measurement, defined as follows:

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

7 Baseline

To evaluate the success of our approaches, we first establish a simplified prediction algorithm. For many datasets, this naive approach yields decent results and can be hard to improve upon.

Our baseline algorithm first calculates a series of means. First, we calculate the residual mean for each item, μ_{item}. This gives a sense of the quality and appeal of the item. *After* subtracting both μ_{item} (for each item) from $\mathbf{X}$, we calculate the residual mean for each user, μ_{user}. This tells us whether a user is a generous or harsh rater.

We form the predicted rating for user i on item j by calculating:

$$\hat{y}_{ij} = \mu_{user_i} + \mu_{item_j}$$

Put another way, we predict the item's average and adjust for bias introduced by the user's rating tendency (harsh or easy).

(a) Results

8 Singular Value Decomposition

(a) Background

Theoretical Basis Single Value Decomposition is a form of Principal Component Analysis, which factors a matrix Y into component matrices $U * \Sigma * V^T$. Principal Component Analysis alters the basis of Y so that, in the factorized form, the directions of greatest variance are exposed in decreasing order. In our dataset, Y represents the user-item interaction matrix, and when we decompose that matrix to constituent parts U and V, we will find the eigenvectors of Y that define the directions of greatest variance. Single Value Decomposition limits the factorization to the K most significant eigenvectors. Because Y is very large, but very sparse, we can simplify the data by factoring to U and V. Where U is a n by k matrix where each row represents a latent feature vector for a user, and V is a d by K matrix where each row represents the latent feature vector for an item. These latent feature vectors are meaningful because, given an incomplete Y matrix (where not all user-item interactions are accounted for), we can iteratively train U and V to approximate the data we do have for Y. New user-item interactions (ratings) are predicted with $\hat{Y}_{ij} = U_i V_j^T$.

Literature Review Single Value Decomposition used a technique to simplify a large, sparse user-interaction matrix has been used very successfully before, most famously in the famous Netflix Prize **cite Gower paper**. Studies on the application of SVD to decompose a matrix has revealed several variants on the basic algorithm to increase the accuracy of the predictions. Such methods include adaptively altering the learning rate, changing the iterative update method, and adding regularization **cite Chih-Chao Ma paper**. Other have also added different normalization constants to the SVD analysis to account for quality and user biases **cite Paterek paper**. Simon Funk, a famous contestant in the Netflix algorithm further tuned the bias weights based on the number of reviews an item or user had cite SF post. Gower explains that successful Netflix SVD algorithms also incorporated time-based information about the user to increase the prediction's accuracy **cite Gower paper**. The SVD method has been explored extensively, and we will attempt to replicate some of the most successful methods on our dataset.

(b) Algorithm

The goal of the SVD find a U and V matrix that well approximate Y. This is accomplished by minimizing the following difference function $E(U, V)$.

$$[\hat{U}, \hat{V}] = \min \|Y_{ij} - \hat{Y}_{ij}\|^2 = \min_{U,V} \|Y_{ij} - U_i V_j^T\|^2$$

Because Y is incomplete, we cannot solve for the closed form solution and thus must update U and V iteratively with a gradient descent. (S is the set of observed user-item iteractions, and I is the indicator of seen ratings for Y.)

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

In compact matrix form, this is $\frac{\partial E}{\partial U} = -2 * I(Y - UV^T)U$

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

In compact matrix form, this is $\frac{\partial E}{\partial V} = -2 * (I(Y - UV^T))^T V$

(c) Hyper-Parameter Tuning

Had to cross validate for regularization...

Tried different bias methods...

(d) Results

Hard to beat the mean?

9 Item-Based Collaborative Filtering

(a) 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).

(b) 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.

(c) Results

10 Comparison of Methods

11 Next Steps