Brewery Boys

Predicting Beer Ratings through Singular Value Decomposition and Collaborative Filtering

Matt McFarland and Theodore Owens

March 8, 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 ratings. 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.

3 Data

Our dataset contains $1,586,599$ reviews concerning $m = 33,388$ users and $n = 65,680$ items. Not all items receive the same number of reviews, nor do all reviewers rate the same number of items. We find that our dataset is highly skewed, with many items and users possessing quite few reviews.

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.

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 a certain number of items and all items that have not received a certain number of 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.

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

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

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.

(a) Summary Statistics

5 Methods

We use three methods to make predictions:

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

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 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 derive the global mean, μ_{global}, across all ratings. From our summary statistics, we see that this figure is around 3.82. After subtracting μ_{global} from all ratings in $\mathbf{X}$, we calculate the residual mean for each item, μ_{item}. This gives a sense of the quality and appeal of the item. Lastly, after subtracting both μ_{global} and 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. Both μ_{user}

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

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

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

(a) Results

7 Singular Value Decomposition

(a) Algorithm

(b) Hyper-Parameter Tuning

(c) Results

8 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:

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

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

9 Comparison of Methods

10 Next Steps