Beer Recommender System

Jason Yingling

December 14, 2017

1 Introduction

Have you ever wondered, “how does Netflix know which movies I would like?”. What about Amazon ‘people who bought this also bought this’. Or “How does Facebook automatically suggest us to tag friends in pictures”? The answer is Recommendation Systems. In this paper, I construct a recommender system using real world beer recommendations from Beeradvocate .

The enormous growth of data has resulted in the creation of new interdisciplinary research to help analyze the data. Recommender systems is one of the emerging topics in industry and academia.

As consumers, we are facing more and more new products and brands. According to Schwartz, facing more choices creates a psychological burden. More choices means more time and effort to choose and increased odds that you’ll make the wrong choice and regret it. Recommender systems help alleviate the psychological burden of information overload. As stated by Gedimas and Alexander, (Gediminas and Alexander, 2004) ”recommender system helps to deal with information overload and provide personalized recommendations, content and service”.

Recommender systems can be divided into two categories, Content-based and Collaborative. Content-based approaches work by directly building profiles for users and products, often through explicit questionnaires and expert evaluation (Hu et al., 2008). An example of such a system is the internet radio Pandora, based on the Music Genome Project, where each song is characterized by an extensive feature set created by a music expert. The features, such as a song’s genre, can then be matched with users’ preferences in order to make recommendations. While this method can be successful, it can be costly to attempt to obtain an exhaustive factor profile. In addition, these profiles hold innate creator bias. If five groups were to construct factors that describe movies, it would be likely that no two models are exactly the same.

In contrast, collaborative filtering relies only on past user behavior to make recommendations. One of the main appeals of collaborative filtering is that it is domain free, yet can address data aspects which are often elusive and difficult to profile with content-based systems (Koren et al., 2009).

There are two approaches to collaborative filtering: neighborhood methods and latent factor models. Neighborhood methods focus on inferring relationships between products and/or users, and using those relationships to make recommendations (Koren et al., 2009). Let's say Bob and Alice have fairly similar interests. Alice rated Star Wars highly but Bob hasn't seen Star Wars. It is reasonable to assume that Bob will also like Star Wars because they have similar interests.

Latent factor models, on the other hand, attempt to characterize users and products by feature vectors automatically inferred from data (Koren et al., 2009). Such feature vectors describe users and products along multiple dimensions. These features can be similar to the explicit profiling in content-based systems, although the actual interpretation of features is often impossible (Koren et al., 2009). As the Netflix Million Dollar Challenge demonstrated, one of the the most effective way to build such systems is through matrix factorization. The winner used an ensemble approach that used SVD models as it's primary base. Matrix factorization algorithms utilize prior item feedback given by users to automatically build user and product profiles. A product can then be recommended to a user if the user's profile closely matches that of the product.

2 Dataset

I use a beer ratings dataset of beer reviews from Beeradvocate, provided by Stanford Network Analysis Program. The data (171.83 MB) spans a period of more than 10 years, consisting of 1.5 Million beer reviews. Each review includes ratings from five "aspects": appearance, aroma, palate, taste, and overall impression. Reviews include product and user information, followed by each of these five ratings.

Since, processing of such a large dataset is highly memory and CPU-intensive, we used validation set as our main data. For the first model, we use only 10 percent of the data. On the second model, and take 50 percent of that dataset to train and evaluate the model. The final model we will use 90%. Further, we will filter out beer that have less than 50 reviews. Removing beer that have low number of user ratings will help the sparsity of the data.

The goal of this project is to create a "Top-N" recommendation for volunteers at the "End of Semester" party for the Mathematics Department, Fall 2017. I provided 12 different beers for volunteers to test and rate. I also requested each volunteer to add other beer that they may have tried previously.

3 SVD

Singular Value Decomposition (SVD) is a well-known matrix factorization technique that factors an m n matrix R into three matrices as the following:

$$R = USV^T$$

. Where, U and V are two orthogonal matrices of size mr and nr respectively, where r is the rank of matrix R. S is a diagonal matrix of size rr having all singular values of matrix R as its diagonal entries (Koren et al., 2009). The matrices obtained by performing SVD are particularly useful for our application because of the property that SVD provides the best lower rank approximations of the original matrix R. We use SVD in recommender systems to perform two different tasks: First, we use SVD to capture latent relationships between customers and products that allow us to compute the predicted likeliness of a certain product by a customer. Second, we use SVD to produce a low-dimensional representation of the original customer-product space and then compute neighborhood in the reduced space. We then used that to generate a list of top-N product recommendations for customers.

SVD for recommendation data:

$$R = Q \cdot P^T$$

here $Q = U$ and $P^T = SV^T$. In order to predict each user rating, $\hat{r}_{ri}$

$$\hat{r} = r_{ui} = q_i^T p_u$$

We must note that recommendation data is sparse, or full of zero entries. The sparsity of recomender data is logical because most users will not recommend every item that is in a database. Therefore, we must modify SVD to ignore the missing data.

q_i and p_u can be found by an iterative process such that square error difference between their dot product and known user rating is:

$$min \sum_{(u,i)\in K} (r_{ui} - q_i^T \cdot p_u)^2$$

However, in order to help handle overfitting, we add the magnitudes squared for user and item:

$$min \sum_{r_{ui}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(||q_i||^2 + ||p_u||^2\right)$$

Where λ is a $L2$ regularization term. The importance of this term can be seen when analyzing the extreme case of a user only giving one item a low recommendation. The minimization will effectively provide low recommendations for every item thereafter, which we know is not logical.

We also need to include bias terms for both the user and item. Examples of biases would be: Alice rates no movies higher than 2 out of 5, or Star Wars Battlefront II receiving low ratings for cultural protest, rather than game quality. Adding in the bias terms, we now have:

$$min \sum_{r_{ui}\in R_{train}} (r_{ui} - b_u - b_i - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2\right)$$

The minimization is performed by stochastic gradient descent (Schwartz et al., 2002):

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$
$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$$
$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u)$$
$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i)$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$. The learning rate γ and regularization term λ are coefficients that can be modified to training the model.

The standard metric for analyzing the accuracy of a model is root mean squared error (RMSE).

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(r_j - \hat{r_j})^2}$$

RMSE measures the distance between predicted preferences and true preferences over items. One of the benefits of SVD is that RMSE is created in the creation of the recommender model (Koren et al. 2009). This begs to question how does minimizing error help customers? Koren argued that even a 1% decrease in RMSE causes a significant change in "Top-N" recommendation systems (Koren, 2007).

3.1 Method

The models will be trained by the full dataset and will provide a biased RMSE since the test will come from the full partition of the data corresponding to the model. Due to memory limitations, this project will be unable to provide cross-validation testing.

The predictions will be constructed by a "anti" test set. This set will be constructed by filling in every item that each user has not rated with the mean rating value. We will then predict ratings for each user and item by using the trained model.

The anti test set provides the largest memory limitation. In Model 3, there are 840k reviews. However, the anti test set consists of 82M reviews, creating a large strain on the computer that is required to predict the model. The memory load is so much that I was unable to complete Model 3.

4

5 Results

5.1 Model 1

For the first model, we use only 10% of the data. The RMSE for Model 1 was .4776 on the biased trained data. For the unbiased data that Model 1 was not trained on, the RMSE was .6165.

The Top-5 items that Model 1 predicted for our volunteers:
Cameron

1. Rare Bourbon County Stout
2. Heady Topper
3. Bourbon County Brand Coffee Stout
4. Â§ucaba (Abacus)
5. Founders KBS (Kentucky Breakfast Stout)

Wes

1. Rare Bourbon County Stout
2. Founders CBS Imperial Stout
3. Beer Geek Brunch Weasel
4. Bourbon County Brand Coffee Stout
5. Heady Topper

Cade

1. Rare Bourbon County Stout
2. Heady Topper
3. Arrogant Bastard Ale
4. Beer Geek Brunch Weasel
5. Founders CBS (Canadian Breakfast Stout)

Jordan

1. Trappistes Rochefort 8
2. Exponential Hoppiness
3. Bourbon County Brand Coffee Stout
4. Trappistes Rochefort 10
5. Great Lakes Barrel-Aged Blackout Stout

Addison

1. Rare Bourbon County Stout
2. Isabelle Proximus
3. Weihenstephaner Hefeweissbier

4. Drie Fonteinen Oude Kriek Van Schaerbeekse Krieken
5. Founders KBS (Kentucky Breakfast Stout)

Samantha

1. Heady Topper
2. Bourbon County Brand Coffee Stout
3. Founders CBS (Canadian Breakfast Stout)
4. Parabola
5. Â§ucaba (Abacus)

Katie

1. Dark Horizon 1st Edition
2. Firestone Anniversary Ale
3. Founders Breakfast Stout
4. Trappistes Rochefort 10
5. Bourbon County Brand Coffee Stout

Missing Name 1

1. Rare Bourbon County Stout
2. Samuel Adams Utopias
3. Founders CBS (Canadian Breakfast Stout)
4. Heady Topper
5. Bourbon County Brand Coffee Stout

Missing Name 2

1. Founders CBS (Canadian Breakfast Stout)
2. Bourbon County Brand Coffee Stout
3. Cantillon Gueuze 100% Lambic
4. Exponential Hoppiness
5. Firestone Anniversary Ale

5.2 Model 2

For the first model, we use only 50% of the data. The RMSE for Model 1 was .4776 on the biased trained data. For the unbiased data that Model 1 was not trained on, the RMSE was .5665.

The Top-5 items that Model 1 predicted for our volunteers:

Cameron

1. Cantillon Gueuze 100
2. Kuhnhenn Bourbon Barrel Fourth Dementia
3. Drie Fonteinen Hommage
4. Matt Hair of the Dog
5. Founders CBS (Canadian Breakfast Stout)

Wes

1. Rare Bourbon County Stout
2. Bourbon County Brand Barleywine Ale
3. Cantillon Gueuze 100% Lambic
4. Wooden Hell
5. Kaggen! Stormaktsporter

Cade

1. Bourbon County Brand Barleywine Ale
2. Wooden Hell
3. Drie Fonteinen Oude Geuze
4. Founders Breakfast Stout
5. Founders CBS (Canadian Breakfast Stout)

Jordan

1. Wooden Hell
2. Founders CBS
3. Firestone Anniversary Ale
4. Heady Topper
5. Cantillon Lou Pepe - Kriek

Addison

1. Founders KBS (Kentucky Breakfast Stout)
2. Heady Topper
3. Firestone - Parabola
4. Kuhnhenn Bourbon Barrel Fourth Dementia
5. Black Damnation IV - Coffee Club

Samantha

1. Wooden Hell
2. Kuhnhenn Bourbon Barrel Fourth Dementia
3. FCantillon Gueuze 100% Lambic
4. Veritas 004
5. Kaggen! Stormaktsporter

Katie

1. Veritas 004
2. Wooden Hell
3. Firestone Anniversary Ale
4. Founders CBS (Canadian Breakfast Stout)
5. Melange No. 3

Missing Name 1

1. Veritas 004
2. AleSmith Speedway Stout - Barrel Aged
3. Black Tuesday
4. Adam From The Wood
5. Cantillon Blåbær Lambik

Missing Name 2

1. Firestone Anniversary Ale
2. Bourbon County Brand Barleywine Ale
3. Wooden Hell
4. Bell's Black Note Stout
5. Cantillon Blåbær Lambik

6 Issues Faced

6.1

One of the major challenges in working with the dataset is memory constraints. The data cannot be stored as a dense matrix due to its huge size. We have to make use of sparse matrix representations in order for the program to work without memory issues. Further, intermediate results such as the user-user similarity matrix cannot be computed and stored due to the huge memory footprint. We had to think of ways to compute the similarity values as and when needed. Further, each model needed substantial run-time.

6.2

Another major issue with the model was selection bias. Because each of our volunteers tried the same beer, there is strong likelihood that the volunteers were put into their own class. Therefore the results may be skewed simply based off of the beer being recommended. One method to help alleviate this issue is to allow volunteers to rate their own beer without any provided beer. This way there will be minor bias introduced into the model by the creator.

7 Future Work

The model presented suffers from several drawbacks that I would like to pursue further. The SVD method suffers from the ”cold-start” scenario. No matter how refined the model is, a new user will have inaccurate predictions. I would like to look into making hybrid models that use implicit characteristics in the future. I would also like to see how socio-economic demographics impact beer ratings. Therefore, future work may consist of creating an ensemble model.

Another avenue of potential research is to create models based on price ranges. Given a user's interested price range, I would like to be able to provide adequate recommendations. In the current form of the model, cost is not a factor for the recommendation. This could make the recommendation potentially useless if the item is outside the range of willingness to purchase, even if the the item is precisely what the user would enjoy the most.

References

[1] Adomavicius, Gediminas and Tuzhilin, Alexander, Recommendation Technologies: Survey of Current Methods and Possible Extensions (2004). Information Systems Working Papers Series, Vol. , pp. -, 2004.

[2] Chih-Chao Ma: “A Guide to Singular Value Decomposition for Collaborative Filtering“ In: csientuedutw (2008)

[3] Koren(2007). "How useful is a lower RMSE?". Netflix Prize Forum. http://www.netflixprize.com/community/viewtopic.php?id=828

[4] Koren et al.: "Collaborative filtering with temporal dynamics" In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (2009), pp. 447-456.

[5] Simon Funk on the Stochastic Gradient Descent algorithm: http://sifter.org/ simon/ journal/20061211.html (Dec 2006)

[6] Schwartz, B., Ward, A., Monterosso, J., Lyubomirsky, S., White, K., Lehman, D. R. (2002). Maximizing versus satisficing: Happiness is a matter of choice. Journal of Personality and Social Psychology, 83(5), 1178-1197

[7] Volinsky et al.: "Matrix Factorization Techniques for Recommender Systems" In: IEEE Computer, Vol. 42 (2009) , pp. 30-37

[8] Y.F. Hu, Y. Koren, C. Volinsky: "Collaborative Filtering for Implicit Feedback Datasets" In: Proc. IEEE Int'l Conf. Data Mining (2008), pp. 263-372

[9] Y.Koren et al.: "Collaborative filtering with temporal dynamics" In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (2009), pp. 447-456.

8 Python Code

```python
import numpy as np
import pandas as pd
import random
from surprise import SVD
from surprise import Dataset
from surprise import evaluate, print_perf
from collections import defaultdict
from surprise import SVD
from surprise import dataset
from surprise import accuracy
from surprise import prediction_algorithms
from surprise import dump


df = pd.read_csv('beer_reviews1.csv')

# Change all of the user names to integers
df['review_profilename'] = pd.factorize(df['review_profilename'])[0] + 1

# Filter out beer that has not been reviewed 50 times.
```

```
df = df.groupby('beer_beerid').filter(lambda x: len(x) >= 50)

reader = dataset.Reader(rating_scale=(1, 5))


data = Dataset.load_from_df(df[['review_profilename', 'beer_beerid', 'review_tas
raw_ratings = data.raw_ratings

# Split data
threshold = int(.90 * len(raw_ratings))
A_raw_ratings = raw_ratings[:threshold]
B_raw_ratings = raw_ratings[threshold:]

data.raw_ratings = A_raw_ratings  # data is now the set A

def get_top_n(predictions, n=10):

    # First map the predictions to each user.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))
    # Then sort the predictions for each user and retrieve the k highest ones.
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]
    return top_n


# retrain on the whole set A
algo = SVD()
trainset = data.build_full_trainset()
algo.train(trainset)

# Compute biased accuracy on A
testset = data.construct_testset(A_raw_ratings)  # testset is now the set A
predictions = algo.test(trainset.build_testset())
print('Biased accuracy on A,', end='   ')
accuracy.rmse(predictions)

# predict r for all pairs (user, item) that are NOT in the training set
# by setting the pairs that were to 0 and the pairs that were not in the
# training set to mean of all ratings.
testset = trainset.build_anti_testset()
predictions = algo.test(testset)

top_n = get_top_n(predictions, n=10)
```

```
# Compute unbiased accuracy on B
testset = data.construct_testset(B_raw_ratings)  # testset is now the set B
predictions = algo.test(testset)
print('Unbiased accuracy on B,', end=' ')
accuracy.rmse(predictions)
```