

Practical 3: Preferences in Streaming Music

Baojia Tong (baojia.tong@cern.ch)
Yuliya Dovzhenko (dovzhenko@g.harvard.edu)
Alan Legollec (alanlegollec@g.harvard.edu)

Public Repository: <https://github.com/tongbaojia/pubcs181>

April 7, 2017

1 Data

1.1 Data Information

First, some basic checks on the dataset is done, as shown in Figs. 1.

1.2 Additional Features

The artist information is gathered from musicbrainzngs API, with informations like tags, ratings, gender, and number of recordings. The user information is given, and further sorted into categories based on their sex, age and country.

1.3 Data Preprocessing for Singular Value Decomposition

In order to perform matrix operations on our data, we had to translate it from dictionary form to a sparse matrix. We used `lil_matrix` format from `scipy.sparse`, because it is intended for incremental construction of the matrix, one value at a time. We then called the singular value decomposition function intended for sparse matrices. We constructed our predictions in `lil_matrix` format as well. With these

To avoid unrealistic features in the data, we first filled in missing datapoints (the ones to be predicted) with median number of listens for the user. We then normalized each user's row to the total number of listens. We explored filling in with means, filling in with zeros, not normalizing,

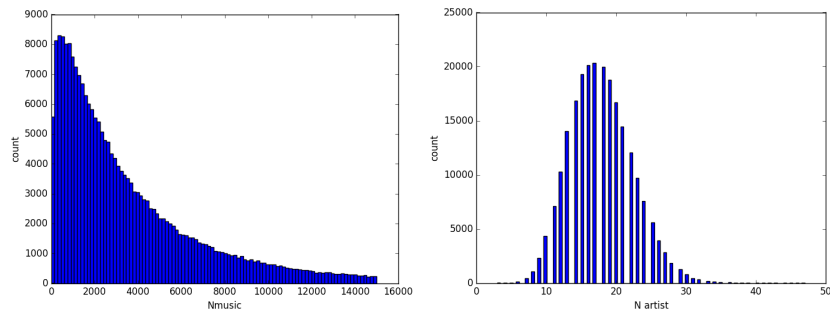


Figure 1: Left: Total number of listens for each user; Right: for each user, the number of artists he/she listens to.

Model	Acc.
BASELINE	137.79146
MODIFIED MEDIAN	137.77158

Table 1: Summary of our best prediction accuracies on the kaggle test dataset.

and normalizing to the median. We found the best performance when filling in with the median, and normalizing to the total number of listens. To undo the normalization after predictions have been made, we kept track of total listens for each user.

To obtain plots in Figs. 3-4, we selected 5% of our data at random to use as a test set for cross-validation.

2 Results

We explored several different methods and submitted the most promising predictions on kaggle. Kaggle scores are summarized in table 1.

2.1 Singular Value Decomposition

For this section, we read a high-level summary of the use SVD in the netflix competition[1], as well as the wikipedia page about singular value decomposition.

Singular value decomposition is an approximate diagonalization procedure, applicable for non-square matrices. It yields approximate eigenvalues (singular values) and approximate left and right eigenvectors. We have to distinguish between left and right eigenvectors, because they represent distinct things and have different dimensions.

$$X = USV \tag{1}$$

Here X is the matrix whose rows represent users and whose columns represent artists. Rows of matrix V are orthonormal eigenvectors in the artist space, i.e. vectors in 2000-dimensional space corresponding to the directions of the largest variability in the data. Each user is then represented by a row of U , which contains weights associated with every eigenvector in V . Intuitively, each user is a weighted combination of several music “trends”, where the trends are represented by vectors in V . A schematic representation of singular value decomposition is shown in Fig. 2.

The number of singular values, k , is a parameter we pass to the SVD function. Once a diagonalization is obtained, there are many possible ways to proceed. Because of time constraints, we limited ourselves to two simple approaches:

1. First, we predict total number of listens of user i to artist j according to the following equation:

$$n_{ij} = U[i, :]SV[:, j] \tag{2}$$

where $U[i, :]$ denotes i 'th row of matrix U . We calculated mean error of the prediction as function of k , the number of singular values. Our results are plotted in Fig. 3. We saw the error in our prediction decrease steadily as function of the number of singular values, reaching

$$\begin{array}{c} \text{artists} \\ \rightarrow \end{array} \begin{array}{c} \left(\begin{array}{c} \text{users} \\ \downarrow \end{array} \right) X \end{array} = \begin{array}{c} \left(\begin{array}{c} \text{users} \\ \downarrow \end{array} \right) U \end{array} \begin{array}{c} \left(\begin{array}{c} S \end{array} \right) \left(\begin{array}{c} \text{artists} \\ \rightarrow \end{array} \right) V \end{array}$$

Singular Values (diagonal)

Figure 2: Illustration of Singular Value Decomposition.

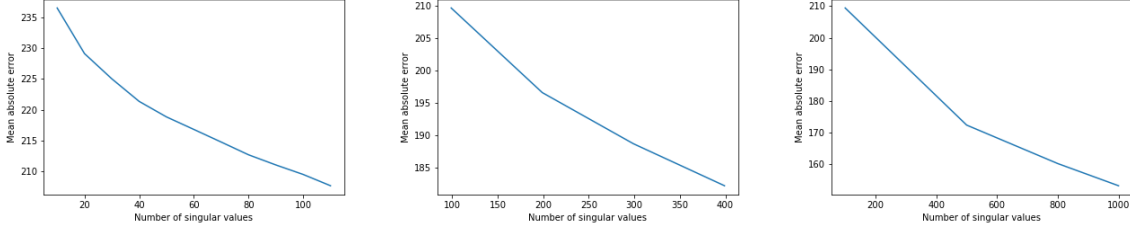


Figure 3: Mean error as function of k , the number of singular values used in SVD. The error is monotonically decreasing.

below 160 with $k = 800$. While the trend was promising, we ran into memory constraints before we could beat the baseline.

A serious concern is that, with k approaching half of the total number of artists, we are most likely overfitting, and in fact we are simply approaching the median values we initially used to fill in the missing values in the data. Intuitively, we would not expect to find as many trends as there are artists.

2. We attempted to improve on the basic SVD prediction by taking advantage of the reduced dimensionality of the data after decomposition to find g nearest neighbors of a given user. We used the KDTree library from the `scipy.spatial` module, which has efficient methods for finding a given number of closet vectors to a given vector. We used rows of U to compute user-to-user distance. We then used an average of the number of listens to the same artist by nearest neighbors to predict listens of a given user. Mean error as function of g , the number of neighbors, is plotted in Fig. 4. The averaging was done after normalizing each user's number of listens. While the error decreased with group size, the computational cost rose very quickly, making it difficult to find convergence in the error.

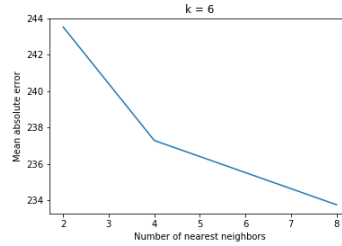


Figure 4: Mean error with fixed $k = 6$, and averaging number of listens from g nearest neighbors in order to make a prediction. g , the number of neighbors, is plotted on the axis.

2.2 User Based Models

We favored for each user, train a small Decision Tree/Random Forest based on the number of plays and the artists feature matrix, just based on the artist's information. The next method we tried is to generate features based on the user features. For the features, we converted them into integers and floats. Features was large, like 10-20 in dimensions, so we tried to use PCA and SelectKBest to reduce the dimensions.

Unfortunately, we ran into computation issues because of re-evaluating the features for each artists too many times. This was fixed only at the last minute and hindered us to further explore and fix potential bugs. Both of these model testing turn out to be extremely computationally expensive given the large dataset; and also, due to the small size of the training sample, which is around 20, the performance of these user based models tend to perform poorly on the test sample.

2.3 Modified Median

Given a lot of methods we tried failed, we went back to evaluate the validity of the median. It certainly helped in cases where there are outliers, but for a rather normal user-preference distribution, it is not as good as mean. Hence, we played a trick that if the user's training standard deviation is smaller than the median, we take the middle three values and re-evaluate the mean of these three medians. This helps to reduce the bias in a normal user and it helped us to beat the median score, although only by a marginal fraction. From this, we think we should change the modeling strategy to find for different groups of users, which estimator—mean or median—would be better. But we don't have enough time for that testing.

3 Discussion

User median is an unexpectedly good predictor, and was hard to beat.

We explored dimension reduction due to sparse nature of the data. We were expecting to find underlying patterns in the data, e.g. a certain subset of users liking a certain genre of music. We pursued SVD. We expected it to perform very well, because of its use in the Netflix prediction challenge. One notable difference is that all Netflix ratings are numbers between 1 and 5, while our number of listens vary considerably from user to user.

A key difficulty of this project is incomplete data. If a certain user has not listened to an artist, that could mean that the user is not interested in the artist. It could also mean that the user would like the artist, but hasn't listened to him or her yet. It was probably for this reason that working with pure listen counts may have been the wrong approach.

References

- [1] Stephen Gower. *Netflix Prize and SVD*. <http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf> (2014).