Practical 3: Preferences in Streaming Music

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

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

April 7, 2017

1 Data

1.1 Additional Features

1.2 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 optimal with incremental construction of the matrix, one value at a time. We then called the singular value decomposition intended for sparse matrices.

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, 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 obtains plots in Figs. 2-3, we selected 5% of our data at random to use as a test set for cross-validation.

2 Results

We explored four different methods and compared the results. Kaggle scores obtained by our methods 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[**?**], as well as the wikipedia page about singular value decomposition.

Model	Acc.
BASELINE	137.79146

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

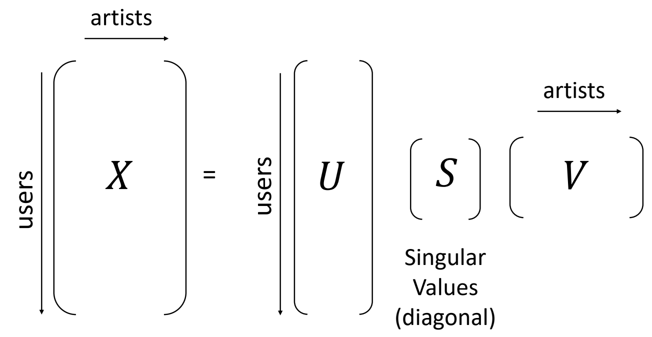

Figure 1: Illustration of Singular Value Decomposition.

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 V, which contains weights asssociated 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. **??**.

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. 2.We saw the error in our prediction decrease steadily as function of the number of singular values, reaching below 160 with $k = 800$. While the trend was promising, we ran into memory constraints before we could beat the baseline.

   Another concern is that, with k approaching half of the total number of artists, we may be in serious danger of overfitting, and in fact we are simply approaching the median values we initially used to fill in the missing values in the data.

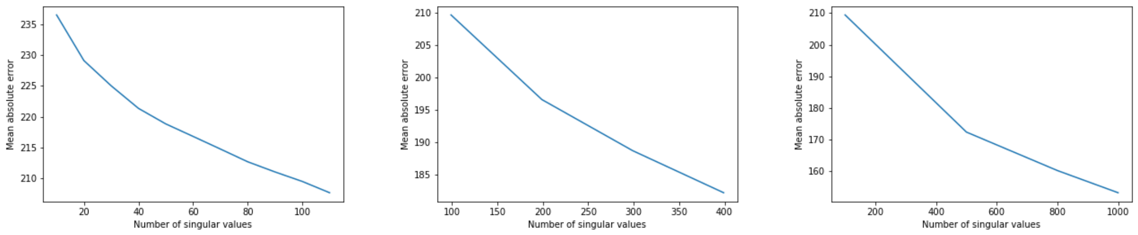

Figure 2: Mean error as function of k, the number of singular values used in SVD. The error is monotonically decreasing.

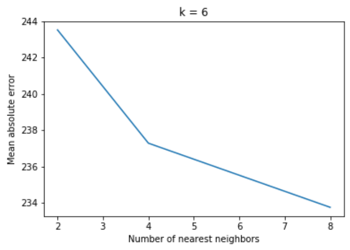

Figure 3: Mean error with fixed $k = 6$, and averaging number of listens from g nearest neighbors in order to make a prediction. g, the number of neighbors, is plotted on the axis.

2. 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. 3.

.

2.2 Method 2

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, which is a powerful technique for finding approximate left and right eigenvectors in the data. 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.