Music Taste Prediction

Mukesh Patel School of Technology Management Engineering, NMIMS

Utsav Shah D036 Arnav Saxena D059 Raunaq Singh D073

April 6, 2016

Contents

- Introduction
- Similarity Functions
- Mean Absolute Error
- Initial Approach
- Normalization L1 Norm
- Normalization Min-Max Formula
- Normalization Train and Test
- Conclusion

Introduction

- In this project, a song recommendation system has been implemented, experimented, and designed.
- This recommendation system builds up a users profile based on his/her past records, and compares it with some reference characteristics.
- Then, it seeks to predict songs that a user would not have listened before.

Introduction(contd)

- One of the most promising technologies is collaborative filtering.
- Collaborative filtering works by building a database of preferences for items by users.
- Methods used for collaborative filtering :
  - User-user Collaborative Filtering
  - Item-item Collaborative Filtering
- In this project, item-item collaborative filtering has been implemented.

The User-Song Matrix

User-Song Matrix [1M x 200]

USERS/SONGS	SONG-1	SONG-2	SONG-3	.	.	SONG-200
USER-1						
USER-2						
USER-3						
.						
.						
.						
.						
.						
USER-1M						

Figure: **The User-Song Matrix**

Similarity Functions

- Weighted Average: This approach computes the prediction on an item i for a user u by computing the sum of the play counts given by the user on similar items to i. Each play count is weighted by the corresponding similarity $s_{i,j}$ between items i and j.

$$\boxed{P_{u,i} = \frac{\sum_{\text{all similar items, N}} (S_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}} (|S_{i,N}|)}}$$

Similarity Functions (contd)

- Cosine-based Similarity: Two items are considered as two vectors in a dimensional space. The similarity between them is measured by computing the cosine of the angle between these two vectors.

$$\boxed{sim(i,j) = \frac{\sum_{u \in U}((R_{u,i})(R_{u,j}))}{\sqrt[2]{\sum_{u \in U}(R_{u,i})^2}\sqrt[2]{\sum_{u \in U}(R_{u,j})^2}}}$$

- Adjusted Cosine Similarity: Here, corresponding user average is subtracted from each co-rated pair.

$$\boxed{sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \vec{R_u})(R_{u,j} - \vec{R_u})}{\sqrt[2]{\sum_{u \in U}(R_{u,i} - \vec{R_u})^2}\sqrt[2]{\sum_{u \in U}(R_{u,j} - \vec{R_u})^2}}}$$

Mean Absolute Error

- Measure of deviation of recommendations from their true user specified values
- Given formula:

$$\boxed{\mathbf{MAE} = \frac{\sum_{i-1}^{N} |p_i - q_i|}{N}}$$

- where
  - p_i : Obtained Value
  - q_i : Predicted Value
  - N : Total Number of Songs

Initial Approach

In this matrix- rows signify users and columns signify songs.

- Each element UxS of the matrix is the play count for the song S by user U.
- So each row is analogous to a user with the values in the row signifying the play count for songs where song numbers correspond to column numbers.
- A single column [**1M users x 1 song**] will be referred to as a song and each row [**1 user x 200 songs**] will be referred to as a user.

Initial Approach (contd)

- Application of cosine similarity algorithm to find the 10 most similar songs for each of the 20 test songs and took a weighted average of the 10 most similar songs for each of the 20 test songs and compute a predicted version of the test dataset matrix.

```
In [84]: mean_absolute_error(test_data, np.rint(predicted_data))
Out[84]: 0.097200299999999989
```

Figure: **1st MAE Result**

Normalization L1 Norm

- In this attempt, the values of the play counts are normalized for each user before splitting the matrix and applying the rest of the cosine similarity and weighted average procedure.
- Normalizing the matrix improves results because play counts of a song do not have rigid boundaries and different users may have different average play count numbers.
- In order to address the above issue, we decided to normalize the dataset between 0 and 1.

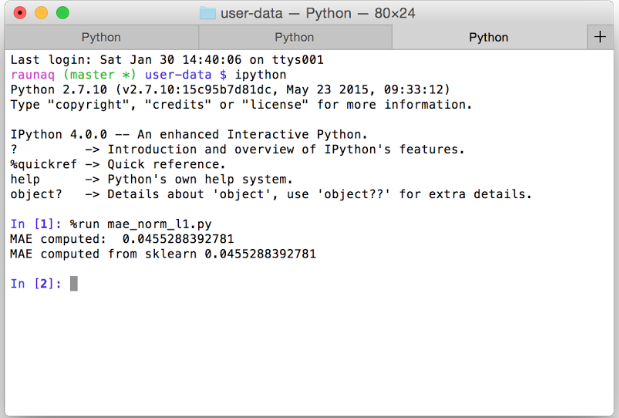
Normalization L1 Norm (contd)

- In this attempt the normalization formula, each play count is normalized between 0 and 1:

$$\boxed{\textbf{L1 Norm} = \frac{\textbf{User's Play Count for Particular Song}}{\textbf{User's Cumulative Play Count}}}$$

Normalization L1 Norm (contd)

- On using this formula, each play count is between 0 and 1 such that the cumulative play count will be 1.



Normalization Min-Max Formula

- On using this formula, every play count lies between 0 and 1 but the sum of the row is not necessarily 1.
- For the sake of convenience, this formula is referred to as min-max normalization.

$$\boxed{\textbf{Min-Max Norm} = \frac{\textbf{Play Count - Minimum Play Count}}{\textbf{Maximum Play Count - Minimum Play Count}}}$$

Normalization Min-Max Formula (contd)

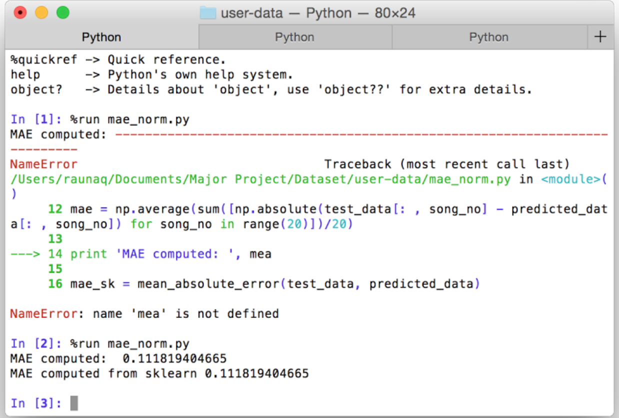


Figure: **3rd MAE Result**

Normalization Train and Test

- The matrix is split before it is normalized and then normalize the training matrix (1M users x 180 songs) and the test matrix (1M users x 20 songs) separately.
- The min-max normalization formula is used for normalizing .

Normalization Train and Test (contd)

The best MAE is obtained

```
In [2]: predicted_data
Out[2]:
array([[ 0.        ,  0.02121257,  0.01557442, ...,  0.11627749,
         0.11777711,  0.1216589 ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       ...,
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.09601624,  0.10276128,  0.        , ...,  0.        ,
         0.        ,  0.        ]])

In [3]: np.average(sum([np.absolute(test_data[: , song_no] - predicted_data[: , song_no]) for song_no in range(20)])/20)
Out[3]: 0.02911209606380636

In [4]:
```

Figure: **4th MAE Result**

Conclusion

This presentation is concluded as follows:

- MAE has reduced gradually step by step.
- Further on, adjusted cosine similarity will be implemented.