Clustering of Last.fm Music Artists

Jordan Rosenblum, Justin Law & Erin Grand

Data Science Institute, Columbia University, New York, NY 10027

May 4, 2015

ABSTRACT

This is the abstract.

Contents

1. Introduction

2. Data Munging

2.1. Matrix Factorization

The goal of matrix factorization is to use collaborative methods to build a recommendation system for users based on user ratings of objects (in this case songs). This will allow us to recommend certain songs to users based on their listening history and without the need for using content based approaches. Since our data set contains number of plays for a given user and song (rather than rating), we normalized plays for every song on a scale between 0 and 1 and used this as a proxy for rating.

We then constructed training and testing matrices, of which both have N_1 users (rows denoted by u_i) and N_2 songs (columns denoted by v_j). Of course the matrices will be very

sparse, containing zeros in all entries except for those in which a user (rows of matrix) has listened to a song (columns of matrix). The goal is to factor the training matrix into the product of two matrixes, U and V. The matrix U will be $N_1 \times d$ and the matrix V will be $d \times N_2$. We want to learn a low-rank factorization (i.e. we choose d) so as to restrict the patterns we see in the rows and columns of our original matrix (e.g. we think a priori that if a user likes 1 top 100 song, the user may also like other top 100 songs). There is subjectivity in picking d but 20 is a common place to start. In the factorized matrices, the predicted rating will be $\hat{M}_{ij} = u_i^T v_j$

Using a coordinate ascent algorithm over 100 iterations, each row (u_i) and column (v_j) of the training matrix is then updated (see equations 1 and 2) in order to maximize the log joint likelihood (see equation 3).

$$u_i = \left(\lambda \sigma^2 I + \sum_{j \in \Omega_{w_i}} v_j v_j^T \right)^{-1} \left(\sum_{j \in \Omega_{w_i}} M_{ij} v_j \right) \tag{1}$$

$$v_j = \left(\lambda \sigma^2 I + \sum_{i \in \Omega_{v_j}} u_i u_i^T \right)^{-1} \left(\sum_{j \in \Omega_{v_j}} M_{ij} u_i \right) \tag{2}$$

$$\mathcal{L} = \sum_{(i,j) \in \Omega} \frac{1}{2\sigma^2} ||M_{ij} - u_i^T v_j||^2 - \sum_{i=1}^{N_1} \frac{\lambda}{2} ||u_i^2|| - \sum_{i=1}^{N_2} \frac{\lambda}{2} ||v_j^2|| + \text{constant} \tag{3}$$

Note: We use a rank 20 matrix for factorization purposes, a $\lambda = 10$, and calculate the variance of the observations for our σ^2. Also, Ω is the set of all indices in the matrix which have an observation.

We keep track of the root mean square error (RMSE) versus the testing set (i.e. how close our prediction is as compared to the actual normalized play count in the testing set) and the log joint likelihood of the training set as a function of iteration (see Figures 1 and 2).

2.2. Kmeans Clustering

Next, we cluster each of the learned u_i using k-means (describe more here later) so that the cluster centroids can be interpreted as listener/personality types.

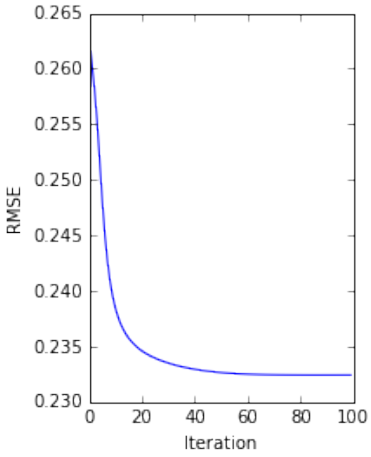

Fig. 1.— (how close our prediction is as compared to the actual normalized play count in the testing set

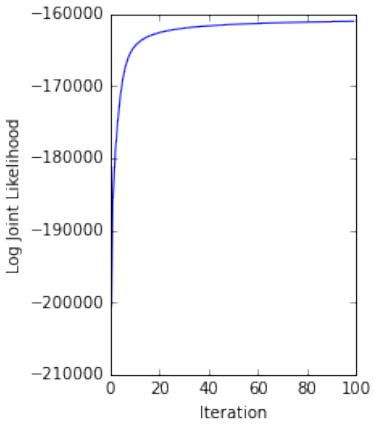

Fig. 2.— log joint likelihood of the training set as a function of iteration

3. Notes

Music Recommendation Dataset

Sparsity of $8.08199453451 \times 10^{-5}$ (needs updating)

The matrix was too big for our computers to handle the matrix factorization using the whole set of users and songs, so we subset the data.

The full matrix was originally 163206 songs x 110000 users.

Subset songs by how many times the song was listened to. i.e the popularity of the song = new number

Subset users by how many songs they've listened to (how many popular song?) = new number

SONG PLAY COUNT
mean = 28
max = 35432
min = 1
std = 215.826789

USER PLAY COUNT
mean = 42
max = 1305
min = 5
std = 53.31547

29 - users 34 - songs

How to split into training and test? Random selection by user or element? ELEMENT. Need matrixes to be the same size to compare them.

Interesting plots: - Cumulative Distribution of number of songs with a given play count

4. Appendix

Link to our github repository Link to the data set