Clustering of Last.fm Music

Jordan Rosenblum, Justin Law & Erin Grand

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

May 6, 2015

ABSTRACT

This is the abstract.

Contents

1. Introduction

In the online world, recommendation systems are used in shopping, entertainment and even dating. Because of their popularity and usefulness, there are lots of incentives to implement a good recommendation system, but a lack of large datasets to learn the best algorithms from. In the music industry, companies use recommendation systems to provide a better music streaming radio service to their listeners. With the music industry shifting away from a more traditional distribution model of physical CDs or records, having an efficient and well performing recommendation system is very important.

In building a recommendation system, there are many problems to tackle. This is way some methods that take more information about the user and the song into account. There is no set similarity measure between two songs, as evident by the fact that songs by the same artist are often very different. As such, there are many different types of algorithms to explore.

For our project, we chose to explore various song recommendation systems algorithms such as matrix factorization, non-negative matrix factorization as well as user based and item based similarity measures.

1.1. Data

The main source of data for this project was the Million Song Data Set (MSD) (**?**). We used the sample data set from the Kaggle competition as well as metadata from MSD.

The full data set had 163206 songs 110000 users, and the number of play counts The scale of the data was too big for our computers to explore, so we subset the data down to a size easily handled by an average laptop.

We choose to subset the data to popular songs and the most active users. We chose a subset of 10% the size of the original matrix.

len(value) ¿ 27 users len(value) ¿ 22 songs

1.1.1. Data Statistics

- Sparsity -

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

USER PLAY COUNT

mean = 42
max = 1305
min = 5
std = 53.31547

1.2. Data Analysis

We did a random selection by data element to split between test and train.

1.3. Testing

We tested our algorithms against each other using a mean average precision score for each method. The mean average precision (MAP) algorithm came directly from the Kaggle MSD challenge (kag 2012). The score was used to rank the competition entries.

The MAP scores were computed by calculating the precision between a list of recommendations for a user and that user's test set. The MAP score is then the average precision value over all the users in the test set.

The bench mark we used to recommend to the top 500 songs to every user.

2. Algorithms

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 allowed 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 were 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 (equations 1 and 2) in order to maximize the log joint likelihood (equation 3).

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

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

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

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 **??** and **??**).

2.2. User Based and Item Based Collaborative Filtering

A simpler set of methods for determining which songs to recommend to individual users is by using user- or item/song- based similarity (both also considered collaborative filtering). These only consider whether or not a user listened to a song and does not take into account the play count. Similarity scores between users (i.e. user-based) and songs (i.e. item-based) were calculated using cosine similarity. Intuitively, they work by recommending users songs that other similar users like (user-based) or by recommending songs which are similar to the songs the user has already listened to (item-based).

2.2.1. User-Based Collaborative Filtering

For user-based recommendation, we first calculated the similarity score between every pair of users, u and v, using the equation below:

$$sim(u,v) = \frac{\#\ \text{common items}(u,v)}{\#\ \text{items}(u)^{1/2} \times \#\ \text{items}(v)^{1/2}} \tag{4}$$

Then, for each user, we looked at each song in the dataset and found all other users who have listened to that song. Next, we added up the similarity score of each of those users, v, with the original user, u, and got the weight for a particular song, i:

$$w_i = \sum_{v \epsilon V} sim(u,v)$$

That sum is a proxy for how likely the user is to like that particular song. We went through that process for all songs and then recommended the user the top 500 songs which had the highest scores. This method led to a MAP value of 0.0383.

2.2.2. Item-Based Collaborative Filtering

For item-based recommendation, which turned out to be our best algorithm in terms of a MAP value, we first calculated the similarity score between every pair of songs, i and j, using the equation below and saved the most similar songs to each song in the process:

$$sim(i,j) = \frac{\#\ \text{common users}(i,j)}{\#\ \text{items}(i)^{1/2} \times \#\ \text{items}(j)^{1/2}} \tag{5}$$

Then, for each user, we found all the songs listened to. Next, we got the most similar songs to each of those songs listened to by the user. If the same similar songs came up multiple times, we added the weights together. In other words, for each song, b, that was found to be similar to one of the songs a that the user listened to, we calculated the score for that song, b:

$$w_b = \sum_{a \epsilon A} sim(a,b)$$

Lastly, we sorted the songs by scores and recommended the user the top 500 songs which had the highest scores. This method led to a MAP value of 0.0503.

3. Conclusion

4. Notes

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

29 - users 34 - songs 60 %

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

To Reference a paper:

The Million Song Dataset Challenge: (McFee et al. 2012)
Million Song Dataset Recommendation Project Report: (Li et al. 2012)
codebook-based scalable music tagging with poisson matrix factorization: (Liang et al. 2014)
Matrix Factorization Techniques for Recommender Systems: (Koren et al. 2009)
A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge: (Aiolli 2013)

REFERENCES

(2012). Million song dataset challenge.

Aiolli, F. (2013). A preliminary study on a recommender system for the million songs dataset challenge. `http://www.ke.tu-darmstadt.de/events/PL-12/papers/08-aiolli.pdf`.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. `http://www2.research.att.com/~volinsky/papers/ieeecomputer.pdf`.

Li, Y., Gupta, R., Nagasaki, Y., and Zhang, T. (2012). Million song dataset recommendation project report. `www-personal.umich.edu/~yjli/content/projectreport.pdf`.

Liang, D., Paisley, J., Ellis, D. P., et al. (2014). Codebook-based scalable music tagging with poisson matrix factorization. `http://www.columbia.edu/~jwp2128/Papers/LiangPaisleyEllis2014.pdf`.

McFee, B., Bertin-Mahieux, T., Ellis, D. P., and Lanckriet, G. R. (2012). The million song dataset challenge. `https://www.ee.columbia.edu/~dpwe/pubs/McFeeBEL12-MSDC.pdf`.

This preprint was prepared with the AAS LaTeX macros v5.2.

5. Appendix

- Link to our github repository: `https://github.com/eringrand/musicanalysis`
- The MSD Kaggle challenge: `https://www.kaggle.com/c/msdchallenge`