Music Recommendation System

Jason Lefever, Mitali Sanwal, Sirui Wang

Problem Description

Building a recommendation system is a common task that is faced by Amazon, Netflix, Spotify and Google. The underlying goal of the recommendation system is to recommend items a user will enjoy. This might involve providing personalized content for a user's specific tastes.

Music Recommendation

Content-based : Recommending based on the content of the music. This could either be the metadata or the audio signal itself. This often leads to predictable recommendations and has fallen out of favor.

Collaborative Filtering : Recommending based on the usage patterns of all users and items. Comes in two flavors: neighborhood and latent model

Popularity : A naive approach that makes the same recommendations to every user. Predicts using the number of times that song was listened to in general by all users.

Research Papers Recommendation

1. **Music Recommendation using Collaborative Filtering and Deep Learning**: Anand Neil Arnold, Vairamuthu S.
2. **Music Recommender System Based on Genre using Convolutional Recurrent Neural Networks**: Adiyansjaha, Alexander A SGunawan DerwinSuhartonoa
3. **Music Recommendation System Music + ML**: Yuhang Lin, Fayha Almutairy, Nisha Chaube
4. **Collaborative Filtering for Implicit Feedback Datasets**: Yifan Hu, Yehuda Koren, Chris Volinsky
5. **Deep content-based music recommendation**: Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen

Dataset

Source: Million Songs Dataset (Columbia University)

- The **Million Song Dataset** is a freely-available collection of audio features and metadata for a million contemporary popular music tracks.

- The **triplet_file** contains user_id, song_id and listen count.

- The **metadata_file** contains song_id, title, release_by and artist_name.

Dataset: Implicit vs Explicit Feedback

- Collaborative filtering of every kinds is based on user-item feedback
- Sometimes this is an explcit rating. Like when “like” a song on Spotify.
- But why limit yourself to just explicit feedback?
- If you look at the implicit listen counts for a user, you can get a picture of their music tastes without the user having tell you anything directly

  But! It is hard to tell if a user does not like something based on implicit feedback. If a user never listened to a song, it doesn’t mean they don’t like it. Maybe they just never had the opportunity to hear it?

Dataset: Interaction Matrix

We have n items (songs)

We have m users

We have a list of (user id, song id, listen count) triplets

We can rearrenge these triplets into an m×n “interaction” matrix

This is implicit feedback!

Each cell contains r_{ui}, the rating / listen count

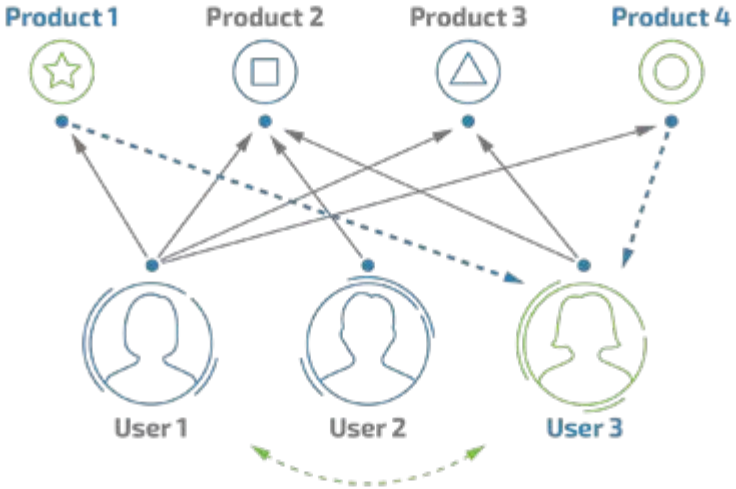
Most cells are zero

Neighborhood Approach

A neighborhood approach requires a similarity metric between either users or items. Depending on which we use, we call say it is user-based or item-based. From there, we directly compare items or users against one another. For user-based one, the intuition is that users who listen to the same songs in the past tend to have similar interests or tastes and will probably listen to the same songs in the future. Similarly, the item-based approach is backed by the idea that songs that are often listened by the same user tend to be similar and are more likely to be listened together in future by some other user

Neighborhood Approach

User-based Neighborhood CF:



Users who listen to the same songs in the past tend to have similar interests and will probably listen to the same songs in future.

Neighborhood CF is one of the most widely used algorithms in terms of recommendations, not only in music. Basically it has two categories, the one is user-based the other one if item-based. For user-based one, it is like, users who listen to the same songs in the past tend to have similar interests or tastes and will probably listen to the same songs in the future. If you look at the graph, user 1 and user 3, both of them listen to the song, lets say, song 2 and 3, so, when user 1 listens to the song 1, it will be recommended to the User3.

Neighborhood Approach

The user-based collaborative filtering algorithm generates the nearest neighbor recommendation group by clustering different users who have the similar interests. The recommendation song will be generated within these groups.

$$Cos(U_1, U_2) = \frac{\sum_{y \in R_{u1} \bigcap \in R_{u2}} R_{u1,y} * R_{u2,y}}{\sqrt{\sum_{y \in R_{U1}} R_{u1,y}^2} * \sqrt{\sum_{y \in R_{U2}} R_{u2,y}^2}}$$

$$Recommend(u, i) = \frac{\sum_{v \in U} Cos(u, v) * R_{v,i}}{\sqrt{\sum_{v \in U} |Cos(u, v)|}}$$

Latent Model Approach

Attempt to discover hidden or “latent” features for users and songs

Do this by decomposing the m×n interaction matrix into two matrices

X of size m×f for latent user features

Y of size n×f for latent item features

f is the number of latent features

Take the inner product, $X_u^T Y_{i\cdot}$, to estimate the unobserved rating user u gave to item i.

Latent Model Approach

How do we find X and Y? By minimizing this objective function.

$$\min_{x_\star, y_\star} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

This consists of a confidence-weighted mean squared error term and a lambda-weighted regularization term to curb overfitting.

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} \qquad c_{ui} = 1 + \alpha r_{ui}$$

Popularity Approach

Arranged the data in dataframe and counted the number of times each song was listened by all the users. Sorting this list in descending order gives us the most listened songs ie recommendation as per the song popularity.

Our Question

How do Neighborhood CF, Latent Factor CF, and Popularity models compare?

Results and Evaluation

How do we evaluate our recomendations?

Ideally we would give them to the users we trained our model on, and see if they listen to them or not.

We can’t do that, so we have to find another way

Instead we will “hide” 20% of positive preferences in our training set

Check if our models can recommend back the hidden items

Results and Evaluation

Still, this only tests for true positives and false negatives

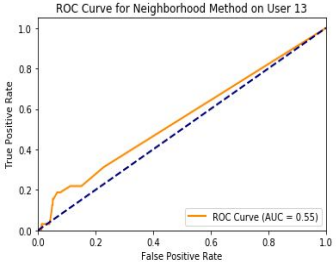
What does a false positive mean? Shouldn't we be recommending items they have not seen? Isn't that the whole point?

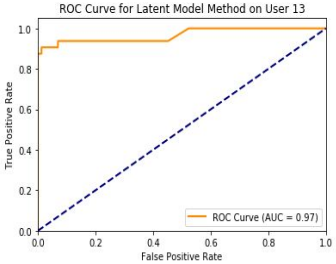
So we can do recall just fine, but precision is essentially undefined

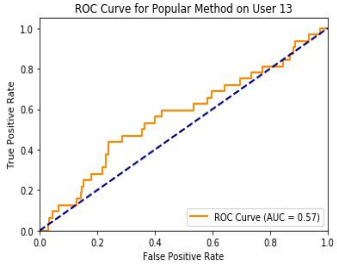
We will find the ROC curve for each user for a more holistic understanding

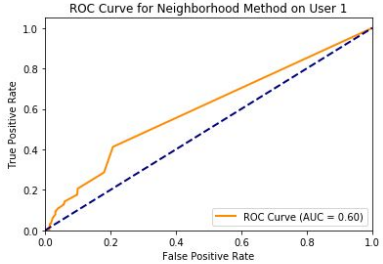
This is still limited but its the best we can do. Probably prone to overfitting.

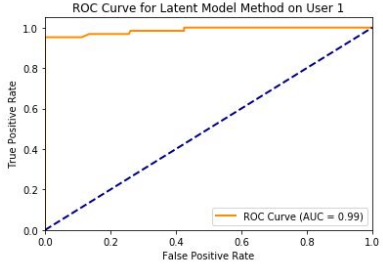
Results and Evaluation

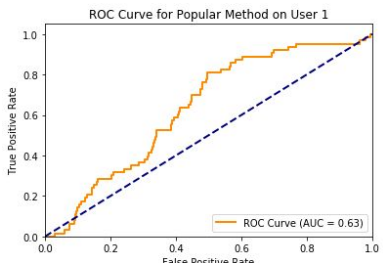












Results and Evaluation

```
Mean AUC (Neighborhood): 0.584695042271343
Mean AUC (Latent Model): 0.9681248924909588
Mean AUC (Popular):      0.6112951213138439
```

Qualitative Comparison

Do the recommenders recommend the same songs?

Why or why not?

Do the song choices seems reasonably relavent?

```
Songs recommended for user 26 by nn:
Sehr kosmisch
Somebody To Love
Just Dance
Fireflies
Teach Me How To Dougie
Nobody (Featuring Athena Cage) (LP Version)
```

```
Songs recommended for user 26 by latent model:
The Real Slim Shady
My Dad's Gone Crazy
Somebody To Love
Without Me
Speechless
Ghosts 'n' Stuff (Original Instrumental Mix)
```

```
Songs recommended for user 26 by popular:
Dress Me Like a Clown
Relax
I C Love Affair
You're The One
Undo
Fools
```

Future Work

Train a classifier to predict the genre of a song given its latent feature vector

Use a content-based method to predict a latent feature vector of a brand new song. This is an attempt to solve the cold start problem.

Algorithms for different aspects like user's mood, current time and day and so on.

Run the algorithms on a distributed system, like Hadoop or Condor, to parallelize the computation.

Evaluate against actaul users!