Music Recommendation System

Team Members:

Yixuan Geng (PennKey: `gyixuan`; Email: `gyixuan@seas.upenn.edu`)
Lichang Xu (PennKey: `lichangx`; Email: `lichangx@seas.upenn.edu`)
Fadi Mikhail (PennKey: `Fadi`; Email: `Fadi.Mikhail@uphs.upenn.edu`)

Assigned Project Mentor:

Anant Maheshwari

Team Member Contributions:

Team Member	Contributions
Yixuan Geng	problem identification literature review problem formulation/definition write proposal and final report algorithm implementations (Topic Model) model evaluation, visualization prepare presentation
Lichang Xu	problem identification literature review problem formulation/definition write proposal and final report algorithm implementations (KNN, Random, and SVD) web interface implementation model evaluation, visualization
Fadi Mikhail	attended only one meeting did nothing else

Code Submission:

[Code can also be accessible via GitHub at `https://github.com/LichangLovesCS/CIS520-Music-Recommender`]

1 Abstract

In this project we have explored several machine learning algorithms such as EM, KNN, and SVD to implement a music recommender system that will recommend a list of songs to the user based on his listening history. We came up with metrics to evaluate the performance of our models and built a web interface (instructions to run this can be found in the README.md at our Git repo) to illustrate the outcomes of two implementations, KNN and SVD. Finally results were discussed together with potential future improvements.

2 Introduction

Recommender system, a subclass of information/data filtering system, aims to predict a user's rating or preference to selected items. With the advancement of data mining techniques, recommender systems have become increasingly popular in industries such as movies, music, academic research, search engine queries, social networks, online shopping, etc.. They are so ubiquitous that people sometimes do not realize their lives have been affected by such state-of-the-art technology. For example, when one purchases on popular E-commerce platforms like Amazon and Ebay, his or her shopping behaviors are constantly evaluated by the recommender system which will later provide items that might be interesting to the customer. A practical implementation of such system will help us better understand how it is applied in real world scenarios. Therefore, in this project we have designed a recommender model using several common machine learning algorithms that given the list of songs a user has listened in the past, predicts and recommends songs and/or artists that are similar to the taste of such user (just like a real music app).

3 Related Work

As mentioned previously, recommendation system is not a completely novel topic. In general there are two types of recommender system, one using collaborative filtering approach and the other using content-based filtering approach [2]. Several great attempts exist in music industry that employ either one or a hybrid of these two methods. For example, Last.fm observes what bands and individual tracks the user has listened to on a regular basis and compares those against the listening behavior of other users (collaborative filtering), whereas Pandora uses the properties of a song/artist to learn user behaviors and refines the recommendations based on users feedback (content-based filtering). Inspired by last.fm and interesting recommender mechanism based on it [1] and due to time constraint, we would mainly explore the collaborative filtering methodology in this project.

4 Data Set

After exploring several public datasets, we have decided to use the Last.fm data that is from the Music Technology Group at the Universitat Pompeu Fabra in Barcelona, Spain. The data were scraped by scar Celma using the Last.fm API, and they are available free of charge for non-commercial use. The dataset is avaiable at `http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html`. Detail of the dataset is included in the README.txt file.

5 Problem Formulation

(i) Problem Formulation
We are formulating this problem as an unsupervised machine learning problem where we need to design and fit an unsupervised machine learning model to the user listening history data and use the model to predict ranking of all songs based on a particular users preference and select the highest ranking songs to recommend to the user.

(ii) Data Processing
Notice that there are a few entries in the original raw dataset that miss fields like artist name or track name, so we need to clean the data by dropping those entries and selecting only relevant fields (i.e., artist name and/or track name). Python's pandas module was chosen and the data processing details are in prepare.py.

(iii) Evaluation Metric
A good recommendation system would tend to give a user a recommended play-list which consists of songs that the user would otherwise encounter in the future and fall in love with anyways. In comparison, a bad recommendation system would give a user a play-list that consists of songs that the user would never want to listen to. So a good evaluation metric of the music generation system is the rank of a user's future songs in our recommended list, assuming that the recommendation list is ordered by relevance and predicted listening frequency of the user for the song. A good recommendation system would correctly identify the songs that user might listen to frequently in the future and therefore place the song high in the recommendation list. As a result most of the songs in the user's actual future play-list would rank high up in the recommendation list. For a bad recommendation system, the opposite happens. The future play-list would rank low in the recommendation list.

Without "actual" future play-list of the user, since they are in the future, we decided to split the dataset into training set and test set, and then train the generative model on training set. For evaluation, we will treat the test set as "future play-list". For each user-song pair in the test set, we will find out the percentile ranking of that song in the recommendation list for that user. The median of all those percentile ranks of test-set songs will be a good evaluation metric of the recommendation system.

6 Algorithms

(i) Topic Model (EM)
Model Specification:
We assume that each user has a different preference for genre, and each music belongs to a set of

genre. The number of genre is set to be 12 which is the number of the major music genres. The model we designed is a generative model where for each observed (user, song) pair, the data is generated via the following process:
(latent variable) $Genre \sim Multinomial(\beta_{U_i})$, where β_{U_i} describes the preference of user U_i on each different genre.
$Song \sim Multinomial(\gamma_{G_i})$, where γ_{G_i} describes the popularity of each song in genre G_i. The parameters all have non-informative dirichlet prior.

$$\begin{aligned} P(\theta, G|y) \propto \prod_{i=1}^{m} [P_{Multinomial}(G_i; \beta_{U_i}) \cdot P_{Multinomial}(S_i; \gamma_{G_i})] \\ \cdot \prod_{U_i} P_{Dir}(\beta_{U_i}; \alpha_U) \cdot \prod_{G_i} P_{Dir}(\gamma_{G_i}; \alpha_G) \end{aligned} \tag{1}$$

we used E-M algorithm to find the MAP estimate of the parameters.

(i) E-step:
First we need to know the probability distribution of the latent variable G for each data point:
For data point i:

$$\begin{aligned} P(G_i|\theta) &\propto P(G_i|\beta_{U_i}) \cdot P(Si|\gamma_{G_i}) \\ &\propto P_{Multinomial}(G_i; \beta_{U_i}) \cdot P_{Multinomial}(S_i; \gamma_{G_i}) \\ P(G_i|\theta) &= \frac{P_{Multinomial}(G_i; \beta_{U_i}) \cdot P_{Multinomial}(S_i; \gamma_{G_i})}{\sum_{g \in Genre} P_{Multinomial}(g; \beta_{U_i}) \cdot P_{Multinomial}(S_i; \gamma_g)} \end{aligned} \tag{2}$$

So the expected complete data log likelihood is given by:

$$\begin{aligned} Q(y|\theta) &= E_G[log(P(y, G|\theta))] \\ &= E_G[log(\prod_{i=1}^{m} [P_{Multinomial}(G_i; \beta_{U_i}) \cdot P_{Multinomial}(S_i; \gamma_{G_i})])] \end{aligned} \tag{3}$$

(ii) M-step:
It is hard to find a θ to exactly maximize the above complete data log likelihood. we found the following approximation that guarantees convergence.

For each user U_i,

$$\begin{aligned} P(\beta_k|G, \gamma) &\propto \prod_{i:U_i=k} P(G_i|\beta_{U_i}) \cdot P(\beta_k|\alpha_U) \\ &\propto \prod_{i:U_i=k} P_{Multinomial}(G_i; \beta_{U_i}) \cdot P_{Dir}(\beta_k; \alpha_U) \end{aligned} \tag{4}$$

Since we know that Dirichlet distribution and multinomial distribution are conjugate, so

$$P(\beta_k|G,\gamma) \sim Dirichlet(\alpha_{U,k}) \tag{5}$$

where
$\alpha_{U,k}$ is a vector such that $\alpha_{U,k}(g) = 1 + \sum_{i:U_i=k} 1(G_i = g)$, $\forall g \in Genre$.

We set β_k to $\alpha_{U,k}$ as an approximation to the optimal β_k in the M-step.

Now similarly for γ:

$$\begin{aligned} P(\gamma_k|G,\beta) &\propto \prod_{i:G_i=k} P(S_i|\gamma_k) \cdot P(\gamma_k|\alpha_G) \\ &\propto \prod_{i:G_i=k} P_{Multinomial}(S_i|\gamma_k) \cdot P_{Dir}(\gamma_k|\alpha_G) \end{aligned} \tag{6}$$

So $P(\gamma_k|G,\beta) \sim Dirichlet(\alpha_{G,k})$, where
$\alpha_{G,k}$ is a vector such that $\alpha_{G,k}(s) = 1 + \sum_{i:G_i=k} 1(y_i = s)$, $\forall s \in Song$.

We set γ_k to $\alpha_{G,k}$ as an approximation to the optimal γ_k in the M-step.

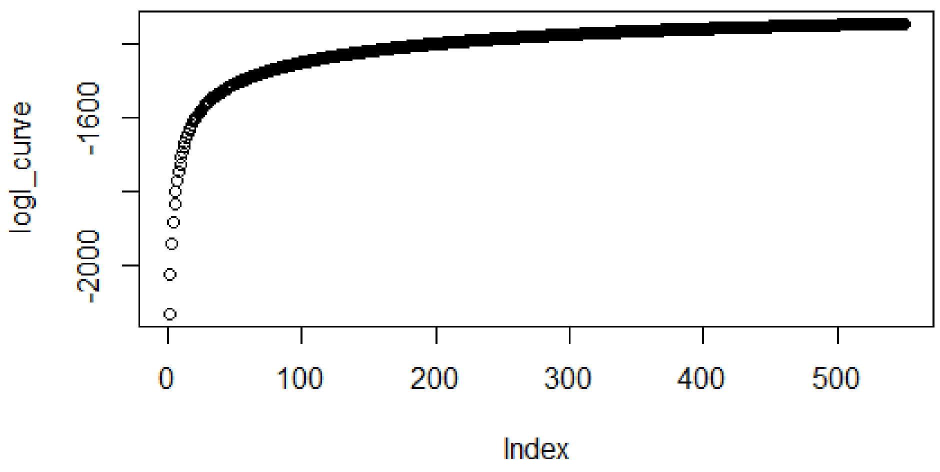
(ii) KNN Model
KNN and SVD models are implemented in Python. These algorithms are chosen because they are suitable to rank songs in the resulting recommendation list based on relevance. However, we also noticed a potential problem in the optimization – the matrix can be really sparse since we used a dictionary to record only non-zero positions and values. To start, we assigned each user, song, and artist to an id [0,1,2,3,4,...] to speed up the processing and to save memory, as seen in s.py. Then we created a matrix that contains information about the user and his listening history, with each value as user K listening to song N for M times. Then we need to have input from a user's listening history, and computed the euclidean distance between each song in the history and the new input, picking k users who are closest in distance. We then selected the songs of these K users (but not in the song history of the input user) as final recommendations.

(iii) SVD Model
To implement the SVD algorithm, we performed SVD operation over the aforementioned matrix that contains users and songs to obtain USV, where A is the original matrix and A = U*S*V' where V' is the transpose of V. Then we used the new user's vector to calculate the cosine similarity to get the most relevant listening histories from the users, and selected the songs that these users have listened to as final recommendations.

7 Experimental Design and Results

(i) Topic Model
Experiment Procedure

we split the training set into 300 training data and 164 test data. We wrote R code to alternate the E-step and M-step for 550 iterations on training set when the expected complete-data log likelihood converges. Here's the graph of the expected complete-data log likelihood over iterations:



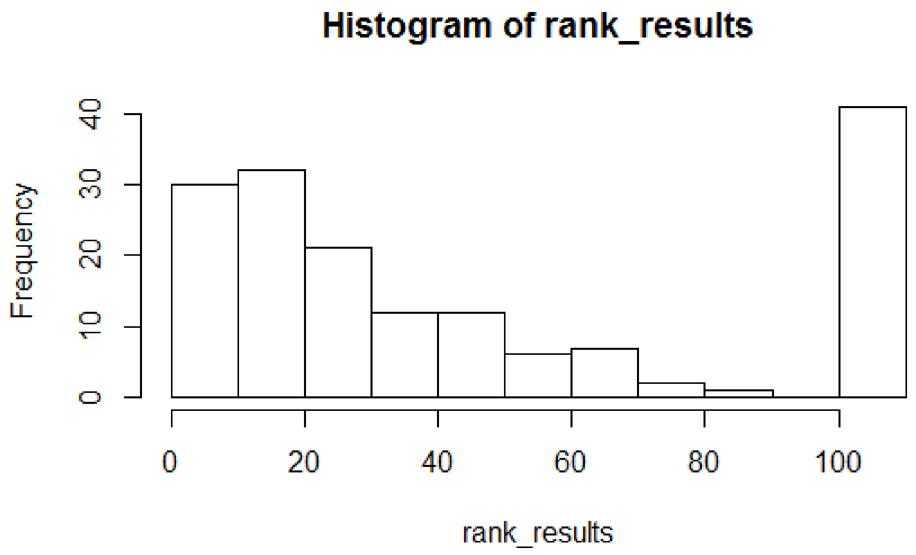
Experiment Results

Now we have our approximately best estimate of our parameters $\widehat{\beta}$ and $\widehat{\gamma}$. For each user, we use these parameters to generate a new play-list consists of 10,000 songs (songs are not unique since this is supposed to be a simulation of the user's listening pattern in the future) based on the user's $\widehat{\beta_{U_i}}$ and parameter $\widehat{\gamma}$ as recommendations:

Now each user has a 10,000 song play-list. We summarize the number of appearances of each song in this play-list and rank the songs by this number to get our final recommendation list. The songs rank high in this list are the ones that we recommend to the particular user the most.

Now for each user we find out the songs he or she listens to in the test set, and then rank those songs in our recommendation list. As discussed in Performance Evaluation metrics, the higher those songs rank in our recommendation list, the better our recommendation list is.

Here's a histogram of the percentile rank of all the songs that show up in the test set:



The median is at top 30% which shows that the model is performing and is at least better than random guessing. The spike at end of the histogram is due to all the songs that do not show up in the recommendation list. We place them at the bottom of the recommendation list for the particular user.

(ii) KNN Model
The KNN model is evaluated similarly to that of EM topic model. The median is at top 32% which indicates that the model slightly outperforms the topic model. However as we bench-marked the run time of KNN, we noticed that its run time is roughly 1.9 times more than that of topic model and 2.2 times more than that of SVD model. This is expected as KNN scales poorly with large datasets compared to the other two models.

Another evaluation metric used to measure the performance of KNN and SVD is to select a portion of users Un and then for every user's listening history Hn, divide it into Hn1 and Hn2. Then use Hn1 as input to obtain result Hn3 and compare Hn3 with Hn2. If there are any intersections, then record the score as 1, otherwise as 0. Based on this gain method, we also noticed that KNN performed better than random recommendation. However since the data is totally offline, such evaluation metric is not deterministic.

(iii) SVD Model
The SVD model is evaluated similarly to the previous models. The median is at top 39% which indicates that the model outperforms the previous models. Based on the gain metrics, SVD again outperformed KNN by approximately 15%.

8 Conclusion and Discussion

As mentioned by the design and result section, we have implemented three algorithms with EM topic model written in R, KNN and SVD models written in Python. In the topic model, from the histogram we can see that the model is able to pick up genre distribution over the songs and users' preference over genres. Our model is able to make good recommendations to the user about songs they've never heard before. Due to limitation of computational power, we had to massively reduce the size of our training set, which leads to potentially more over-fitting and thus many songs in the test set are overlooked by the model.

During our model evaluation process, we found EM model to have a median at top 30%, KNN model to have a median at top 32%, and SVD model to have a medain at top 39%, which implies the three models all outperformed the random one as shown in our web interface. However, they are just slightly better than random guessing. This result is expected as the evaluation and adjustment of recommendation system are never as straightforward as classic classification and regression problems. In fact, ”good” recommender systems are usually implemented as a hybrid of collaborative filtering and content-based filtering. Since our models are offline and do not have any user feedback, the performances are expected to be somewhat unsatisfactory. Nonetheless this introductory ML project teaches us an important lesson in real-life applications of ML techniques – not all algorithms are suitable for the same problem and an excellent model usually involves more than one algorithms and is closed-loop in nature. There are several possible future improvements, including but are not limited to, implementation of a hybrid model with user feedback that scales to massive datasets, more complex evaluation metrics that better capture the semantics of the recommendation, and incorporation of advanced ML techniques such as NLU/NLP.

Acknowledgments

Thanks the CIS520 course staff, especially Anant Maheshwari for his help.

References

[1] Nick Becker. Music recommendations with collaborative filtering and cosine distance. *https://beckernick.github.io/music$_r$ecommender/*, 2016.

[2] Koren et al. Review articles on recommender systems. *IEEE Computer*, 2009.